



# circuit cellar

## Tips for Dealing with High-Speed Signal Transmission Errors

SIGNAL TRANSMISSION & BIT ERROR RATE (BER)

$$BER = \frac{\# \text{ ERRORS}}{\text{TOTAL \# TRANSMISSIONS}}$$

TRANSMITTED

RECEIVED

11101000

11101000

00010100

00010100

01011110

01111110

11100000

11100000

11110011

11110011

11100110

11000110

101000

11101000

UNDETECTED  
ERRORS

Bit Error Rates Explained,  
BER Testers, BER vs. Noise  
Experimentation, & More

■ Q&A with an Application Engineer | Editors' Picks: Wireless Projects

■ Build an Electronics Testing Platform | DIY Five-Function Signal Analyzer

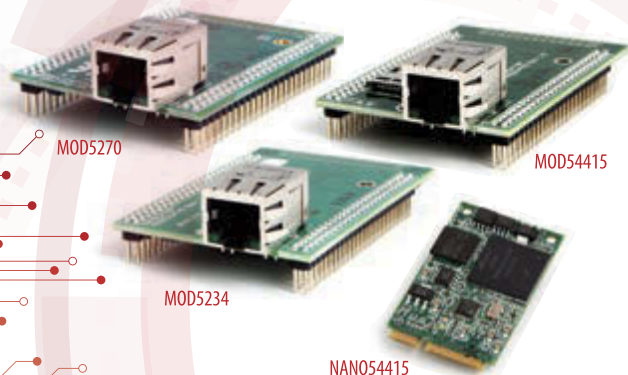
| Four Reverse Engineering Projects | Arduino-Based Tube Stereo Preamp

■ Practical EMC Requirements | ISE Project Management | Embedded Software

Development | Electroluminescence 101 ■ The Future of Embedded Linux



# Ethernet Core Modules with High-Performance Connectivity Options

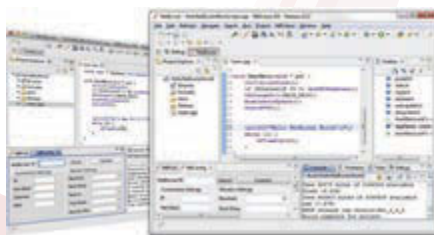


- **MOD5270**  
147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO · 3 UARTs · I<sup>2</sup>C · SPI
- **MOD5234**  
147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs · I<sup>2</sup>C · SPI · CAN · eTPU (for I/O handling, serial communications, motor/timing/engine control applications)
- **MOD54415**  
250 MHz processor with 32MB flash & 64MB RAM · 42 GPIO · 8 UARTs · 5 I<sup>2</sup>C · 3 SPI · 2 CAN · SSI · 8 ADC · 2 DAC · 8 PWM · 1-Wire® interface
- **NANO54415**  
250 MHz processor with 8MB flash & 64MB RAM · 30 GPIO · 8 UARTs · 4 I<sup>2</sup>C · 3 SPI · 2 CAN · SSI · 6 ADC · 2 DAC · 8 PWM · 1-Wire® interface

**Add Ethernet connectivity to an existing product, or use it as your product's core processor**



**The goal:** Control, configure, or monitor a device using Ethernet



**The method:** Create and deploy applications from your Mac or Windows PC. Get hands-on familiarity with the NetBurner platform by studying, building, and modifying source code examples.



**The result:** Access device from the Internet or a local area network (LAN)

**The NetBurner Ethernet Core Module** is a device containing everything needed for design engineers to add network control and to monitor a company's communications assets. For a very low price point, this module solves the problem of network-enabling devices with 10/100 Ethernet, including those requiring digital, analog and serial control.

MOD5270-100IR.....\$69 (qty. 100)	NNDK-MOD5270LC-KIT.....\$99
MOD5234-100IR.....\$99 (qty. 100)	NNDK-MOD5234LC-KIT.....\$249
MOD54415-100IR.....\$89 (qty. 100)	NNDK-MOD54415LC-KIT.....\$129
NANO54415-200IR....\$69 (qty. 100)	NNDK-NANO54415-KIT.....\$99

**NetBurner Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kits include all the hardware and software you need to build your embedded application.

➤ **For additional information please visit**  
<http://www.netburner.com/kits>

mouser.com

The Newest Products for Your Newest Designs®



**More** new products  
**More** new technologies  
**More** added every day



Authorized distributor of semiconductors  
and electronic components for design engineers.



**MOUSER**  
ELECTRONICS



Issue 295 February 2015 | ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by:

Circuit Cellar, Inc.  
111 Founders Plaza, Suite 300  
East Hartford, CT 06108

Periodical rates paid at East Hartford, CT, and additional offices.

One-year (12 issues) subscription rate US and possessions \$50, Canada \$65, Foreign/ ROW \$75. All subscription orders payable in US funds only via Visa, MasterCard, international postal money order, or check drawn on US bank.

#### SUBSCRIPTIONS

Circuit Cellar, P.O. Box 462256, Escondido, CA 92046

**E-mail:** circuitcellar@pcspublink.com

**Phone:** 800.269.6301

**Internet:** circuitcellar.com

**Address Changes/Problems:** circuitcellar@pcspublink.com

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

#### ADVERTISING

Strategic Media Marketing, Inc.  
2 Main Street, Gloucester, MA 01930 USA

**Phone:** 978.281.7708

**Fax:** 978.281.7706

**E-mail:** circuitcellar@smmarketing.us  
Advertising rates and terms available on request.

#### New Products:

New Products, Circuit Cellar, 111 Founders Plaza, Suite 300  
East Hartford, CT 06108, E-mail: newproducts@circuitcellar.com

#### HEAD OFFICE

Circuit Cellar, Inc. 111 Founders Plaza, Suite 300  
East Hartford, CT 06108  
Phone: 860.289.0800

#### COVER PHOTOGRAPHY

Chris Rakoczy, www.rakoczypphoto.com

#### COPYRIGHT NOTICE

Entire contents copyright © 2015 by Circuit Cellar, Inc. All rights reserved. Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited.

#### DISCLAIMER

Circuit Cellar® makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, Circuit Cellar® disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published by Circuit Cellar®.

The information provided by Circuit Cellar® is for educational purposes. Circuit Cellar® makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader's jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

© Circuit Cellar 2015 Printed in the United States

## 2015 UPDATES: CC NEWSLETTER & CALENDAR

The new year is in full swing. So I will take this opportunity to update you on a few important items.

### SUBMISSIONS & EDITORIAL CALENDAR

The free Circuit Cellar Newsletter is an excellent way to receive essential electrical engineering tips, embedded systems industry news, and exciting product deals via e-mail to your inbox on a regular basis. If you're looking for useful electrical engineering-related information, we've got you covered: embedded development, programmable logic, wireless communications, robotics, analog techniques, embedded programming, and more! Join the over 25,000 others who receive the Circuit Cellar Newsletter each week. For more information, go to <http://circuitcellar.com/circuit-cellar-newsletter-subscribe/>.

### SUBMISSIONS & EDITORIAL CALENDAR

Circuit Cellar publishes articles, books, and website content (workspaces, tutorials, project posts) by talented authors on embedded technology-related topics. When you publish with Circuit Cellar, you are reaching an educated professional audience of engineers, programmers, and academics around the globe.

We offer three main publishing opportunities:

- Article Publishing (2000-to-4000-word feature articles, product reviews, and columns in Circuit Cellar magazine)
- Book Publishing (full-length books on relevant engineering topics, lab manuals, DIY project books, etc.)
- Website Publishing (project write-ups, workspaces, and more at CircuitCellar.com)

Our editors carefully review all proposals and finished submissions before making final decisions about publication. Your submission should meet all the requirement specified in our Authors Guide, which you can access at <http://bit.ly/1Bde7VM>.

Here is our 2015 editorial calendar.

ISSUE	MONTH	PROPOSAL DEADLINE	SUBMISSION DEADLINE
294	January	September 1, 2014	October 1, 2014
295	February	October 1, 2014	November 1, 2014
296	March	November 1, 2014	December 1, 2014
297	April	December 1, 2014	January 1, 2015
298	May	January 1, 2015	February 1, 2015
299	June	February 1, 2015	March 1, 2015
300	July	March 1, 2015	April 1, 2015
301	August	April 1, 2015	May 1, 2015
302	September	May 1, 2015	June 1, 2015
303	October	June 1, 2015	July 1, 2015
304	November	July 1, 2015	August 1, 2015
305	December	August 1, 2015	September 1, 2015

**C. J. Abate**

[cabate@circuitcellar.com](mailto:cabate@circuitcellar.com)

## THE TEAM

### EDITOR-IN-CHIEF

C. J. Abate

### ART DIRECTOR

KC Prescott

### ADVERTISING COORDINATOR

Kim Hopkins

### PRESIDENT

Hugo Van haecke

### COLUMNISTS

Jeff Bachiochi (From the Bench), Ayse K. Coskun

(Green Computing), Bob Japenga (Embedded in Thin Slices), Robert Lacoste (The Darker Side), Ed Nisley (Above the Ground Plane), George Novacek (The Consummate Engineer), and Colin O'Flynn (Programmable Logic in Practice)

### FOUNDER

Steve Ciarcia

### PROJECT EDITORS

Chris Coulston, Ken Davidson, and David Tweed

### OFFICE ASSISTANT

Alice Roberts

## OUR NETWORK



## SUPPORTING COMPANIES

Accutrace	7	Lemos International	15
All Electronics Corp.	79	MaxBotix, Inc.	79
Custom Computer Services	79	Mouser Electronics	1
EMAC, Inc.	15	MyRO Electronic Control Devices, Inc.	79
HuMANDATA, Ltd.	79	NetBurner, Inc.	C2
IAR Systems	11	R.E. Simth, Inc.	45
Imagineering, Inc.	C4	Saelig Co., Inc.	27
Ironwood Electronics	79	Technologic Systems	17, 33
Jeffery Kerr, LLC	79	Triangle Research International, Inc.	79

## NOT A SUPPORTING COMPANY YET?

Contact Peter Wostrel (circuitcellar@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706) to reserve your own space for the next edition of our members' magazine.

## TRENDING INFO

You'll receive electrical engineering tips, interesting electronics projects, embedded systems industry news, and exclusive product deals via e-mail to your inbox on a regular basis. If you're looking for essential electrical engineering-related information, we've got you covered: microcontroller-based projects, embedded development, programmable logic, wireless communications, robotics, analog techniques, embedded programming, and more!

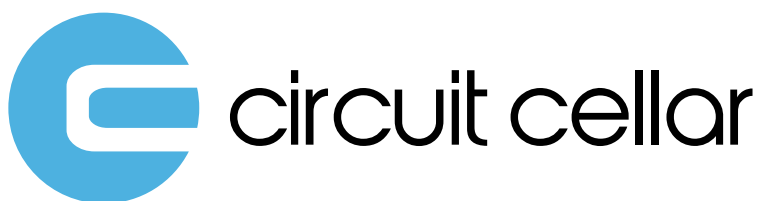
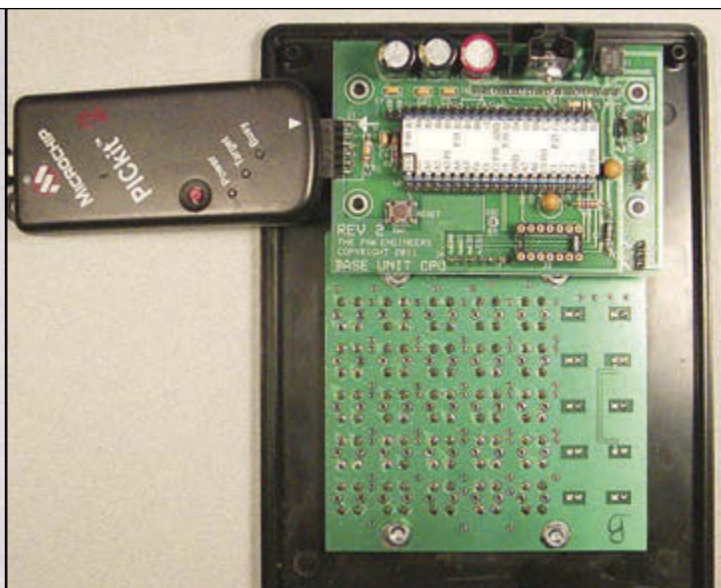
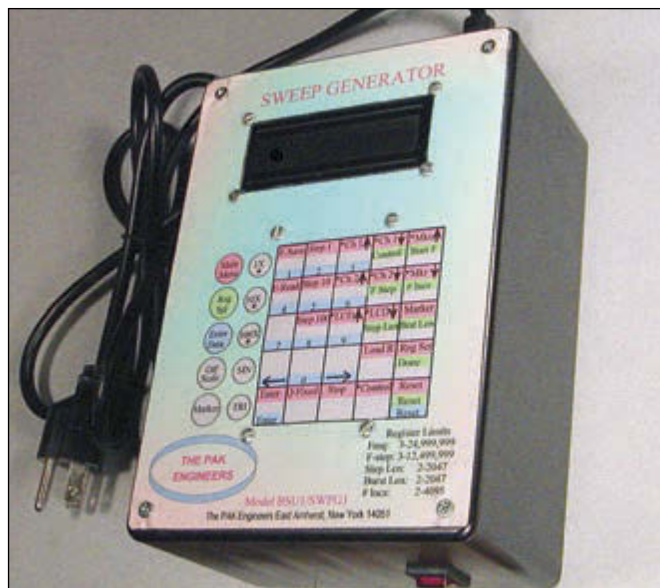
Subscribe now to stay up-to-date with our latest content, news, and offers!

**FREE at circuitcellar.com**



## CONTENTS

FEBRUARY 2015 • ISSUE 295

WIRELESS  
COMMUNICATIONS

## ELECTRONICS TESTING PLATFORM

## CC COMMUNITY

06 : QUESTIONS & ANSWERS  
Application Engineering

An Interview with Carmen Parisi

An applications engineer describes his work and shares some engineering tips

## 14 : EDITORS' PICKS

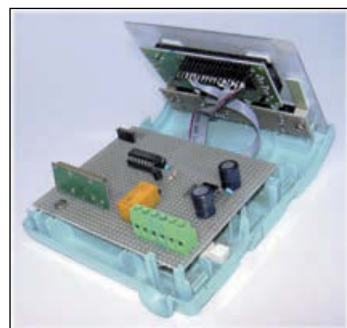
Several interesting wireless communications projects that will stimulate your creativity

## INDUSTRY &amp; ENTERPRISE

## 16 : PRODUCT NEWS

## 19 : CLIENT PROFILE

EarthLCD.com (Costa Mesa, CA)



## WIRELESS PROJECTS

## FEATURES

## 20 : Electronics Testing Platform (Part 1)

Base Unit Construction

By Gerard Fonte

Build a base unit for a DIY multifunction test instrument

## 28 : Five-Function, Network-Connected Signal Analyzer

By Neal Martini

A test tool featuring a digital scope, spectrum analyzer, signal generator, noise generator, &amp; filter response viewer

## 40 : Reverse Engineering Review

Insights from Four Projects

By Fergus Dixon

How to overcome several reverse engineering challenges

## 46 : Budgie

An Arduino-Based Tube Stereo Preamplifier

By Shannon Parks

How to upgrade a DIY preamplifier with an Arduino

## COLUMNS

52 : THE CONSUMMATE ENGINEER  
Essential Electromagnetic Compliance (Part 3)

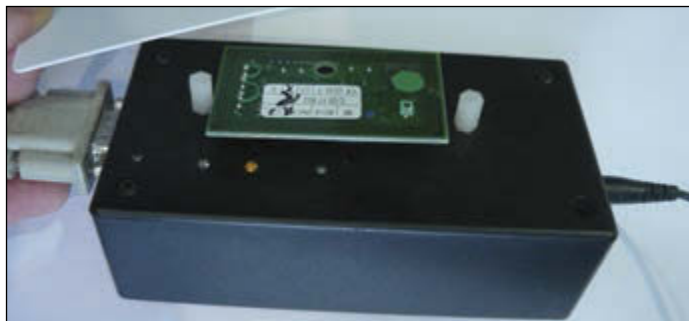
Practical EMC Requirements

By George Novacek

EMC requirements for electronic instruments &amp; systems



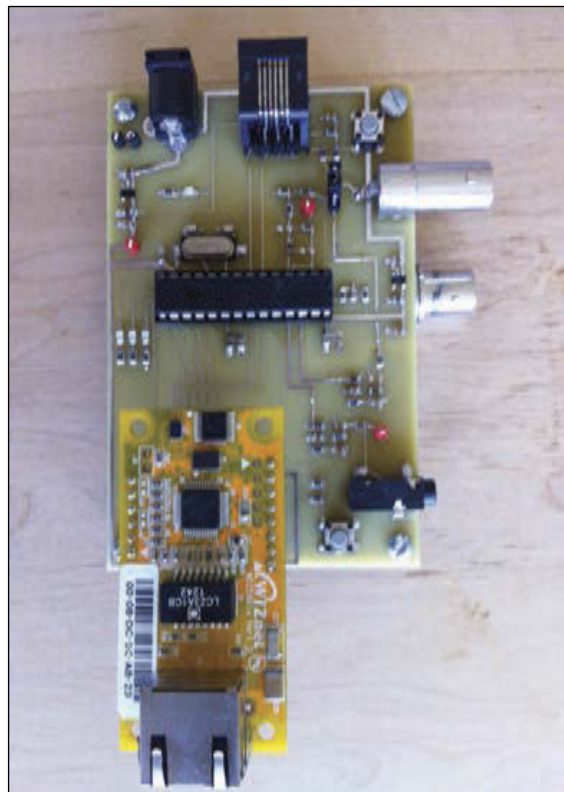
## CONTENTS



REVERSE ENGINEERING PROJECTS



BIT ERROR RATE (BER) TESTER



FIVE-FUNCTION, NETWORK-CONNECTED SIGNAL ANALYZER

### 55 : PROGRAMMABLE LOGIC IN PRACTICE

#### The DIY Approach to ISE Project Management

By Colin O'Flynn

Tips for managing an FPGA project that involves several automatically generated (COREGen) modules

### 60 : EMBEDDED IN THIN SLICES

#### Estimating Your Embedded Systems Project (Part 2)

Challenges Unique to Embedded Software Development

By Bob Japenga

An analysis of issues unique to the process of estimating an embedded systems project

### 64 : THE DARKER SIDE

#### Let's Count Errors

An Introduction to BER Testers

By Robert Lacoste

Details of how a bit error rate (BER) tester can help you properly measure error rates

### 70 : FROM THE BENCH

#### Solid-State Lighting (Part 1)

An Introduction to Electroluminescence

By Jeff Bachiochi

A thorough explanation of electroluminescence (EL) and tips for setting up an EL controller

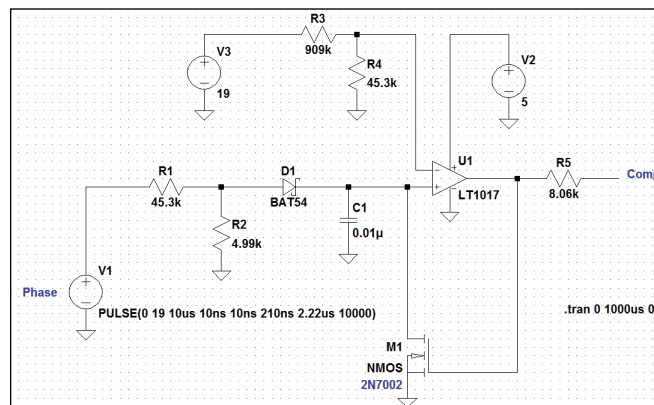
### 78 : CROSSWORD

## TECH THE FUTURE

### 80 : The Future of Embedded Linux

By David Lynch

The rate of embedded Linux adoption is growing and likely won't slow down anytime soon



PRANK CIRCUIT

## TESTS & CHALLENGES

### 77 : TEST YOUR EQ

## QUESTIONS & ANSWERS



# Application Engineering

## An Interview with Carmen Parisi

Carmen Parisi is an applications engineer who co-hosts an engineering podcast in his spare time. In this interview, he describes his work, shares some engineering tips, and tells us about a fun prank he played on an unsuspecting designer.

**CIRCUIT CELLAR:** Where are you located?

**CARMEN:** Currently, I'm living and working in Raleigh-Durham, NC, around the Research Triangle Park area between the two cities with my wife and new dog Sadie. Kelly and I moved down about three years ago from Buffalo, NY, and really like it here. There's a lot of tech companies and engineers around, tons of stuff to do, and great food and beer scenes. Plus, as a hearty Northerner, I get to laugh at the "cold" winters we experience. Come summer, though, I melt into a puddle on the pavement. Snow all the way for me, but Kelly disagrees.

**CIRCUIT CELLAR:** When did you decide to pursue electrical engineering and why?

**CARMEN:** Ever since I was a kid I had a fascination with tools and how things worked. I would always have a toy sword and various tools stuffed into my belt and would volunteer to help my dad around the house building a deck around the pool or fixing the fence.

Once I got into high school, I took a few basic engineering courses during which time I got bit by the engineering bug. The course that really "doomed" me to a life of electronics was a Robotics course taught by my favorite teacher C, as we called him. He put me through my paces learning how to solder, reading schematics, programming in BASIC, and robbing Fort Knox using a LEGO Mindstorms robot. C's class solidified my choice to go to college for engineering, and shortly thereafter, I picked electrical over mechanical for my major.

**CIRCUIT CELLAR:** When was the first time you used a microcontroller in a project?

**CARMEN:** If we're counting LEGO Mindstorms,

then the Robotics class in tenth grade with C where we had to build a robot to lift a golden brick and run away with it (thus "robbing Fort Knox"). I met all the individual milestones with my group for the project, but we couldn't get the whole thing working smoothly from beginning to end. I guess that was my first time learning how to successfully fail too which has turned out to be a very useful skill.

My first real microcontroller experience was the summer after sophomore year when I took a college course at a local community college offering a few classes to high school students interested in engineering. During that course I learned more basic circuit theory, got introduced briefly to SMT soldering, and built some robots using the Parallax BOE Bot. Looking back, I'd say this was the time my analog career kicked off as I slowly started to realize that I was more interested in the circuits themselves than the overall robot.

**CIRCUIT CELLAR:** Tell us about your university-level schooling.

**CARMEN:** I still consider myself a student in that I'm always looking to learn new things and grow as an engineer, but my formal schooling is over for the foreseeable future. In 2011, I completed a combined BS/MS degree in Electrical Engineering at the Rochester Institute of Technology in Rochester, NY. I initially started off interested in robotics but after working with a great analog designer on my first co-op at GE, I switched into the analog circuit and semiconductor track and never looked back.

**CIRCUIT CELLAR:** Can you tell us about your work in graduate school?

**CARMEN:** Sure thing. My graduate work was



PRINTED CIRCUIT BOARDS

THINK YOU CAN FIND PCB PRICES THAT BEAT OURS?

# WE DARE YOU.

If you do, than we  
will match the price  
AND give you \$100  
towards your  
next order!

**THERE ARE NO GAMES INVOLVED IN OUR PRICING**



## Our Capabilities:

- From same day quick turn prototype to production in under 10 days
- Full CAD and CAM review plus design rule check on ALL Gerber files
- Materials: Fr4, Rigid, Flex, Metal Core (Aluminum), Polyimide, Rogers, Isola, etc.
- HDI Capabilities: Blind/Buried Microvias, 10+N+10, Via-in-Pad Technology, Sequential Lamination, Any Layer, etc.
- Our HDI Advantage: Direct Laser Drilling, Plasma De-Smear Technology, Laser Microvia, Conductive Plate Shut.

Take the Accutrace Challenge and see WHY OUR PRICING CANNOT BE BEATEN

**Accutrace inc.**

[www.PCB4u.com](http://www.PCB4u.com) [sales@PCB4u.com](mailto:sales@PCB4u.com)

## QUESTIONS & ANSWERS

primarily with the Communications professor who needed a proof of concept built to test out a theory that looked plausible on paper. Prior to my joining the Comms Lab, my advisor and two past grad students had worked out a method of securing wireless channels using the randomness of the channel itself. There was an initial front end of sorts to test the idea out but I don't think it was ever tested.

I looked over the circuit design, decided to scrap it and start fresh, and immediately realized I had a big job ahead of me. Cue the analog professor becoming my co-advisor. Mixing circuits, active filters, phase detectors, ADCs, and communication theory swam through my head as I slowly cajoled the circuit to life. Two PCB revisions later the circuit worked in that it took the RF input signal and spat out some bits at the other end, but after my advisor applied his algorithm to the data,

an empty terminal bench. The next morning, groggy and sleep deprived, I suited up in the family restroom and flew out for six wonderful hours of technical interviews. I was absolutely wiped out by the end of the day but managed to survive the ordeal. The rest is history.

**CIRCUIT CELLAR:** Tell us about the work you are doing as an applications engineer for Intersil.

**CARMEN:** Well, for starters, being an apps engineer is exactly the rock n' roll lifestyle I'm sure all your readers expect it to be. I roll into the office every morning and have the roadies warm up my iron for me!

In reality though, I work on buck regulators for computing applications like notebooks, tablets, ultrabooks, with maybe a bit of desktop work from time to time. Most of the parts I work on are for the primary core voltage on Intel processors. Sometimes, should the part integrate multiple regulators, I'll work on a graphics rail or one of the other many voltage rails present on a motherboard. For each new processor tock (tick?

I always confuse the two), Intel releases a laundry list of specs that have to be met in order to provide power to their CPUs and my parts are designed to those specs.

When I work apps on a brand spanking new chip, I'll first work with the design engineers to run some feasibility studies and help define any new features for the IC. These tests range from tuning a similar part to the new Intel specs to see if the control scheme hits any corners or has stability issues to beating up some power FETs to determine if they can handle the new current requirements we have to meet. Once the chip tapes out, I'll start work on preliminary documentation—a rough datasheet draft or early reference design based on feasibility testing and simulations—for the field to use when working with customers. During this time, I also design the evaluation board I'll use to validate the part and send to customers for sampling.

The real meat and potatoes of my job is silicon validation. I've got an exhaustive spreadsheet of bench tests to do that functionally verify the IC over a wide range of corners. The first few weeks after silicon comes back I'm working full throttle, round the clock if need be, to make sure there are no show stopping bugs we need to address. I never see my office during validation. Instead I'm spending all my time in the lab hunched over the eval board or squinting at my scope. Things calm down slightly after the initial validation, but the work is still nowhere near

*When I work apps on a brand spanking new chip, I'll first work with the design engineers to run some feasibility studies and help define any new features for the IC.*

we weren't able to generate symmetric keys on different boards. Whether this was from an error in theory or with my board I never found out, as I ended the project there to focus on my full-time job leaving with a grad paper instead of a full thesis.

I still have all my old lab notebooks, schematics, and board layouts on my bookshelf at home. I think the files are sitting on a hard drive somewhere too. Looking at them now, I can spot a lot of little errors I'd like to fix due to my inexperience at the time and some maybe a few not so little errors too.

**CIRCUIT CELLAR:** What did you do after school?

**CARMEN:** After I left RIT, I moved down here to Raleigh-Durham to start my career as an Applications Engineer working on switching regulators with Intersil. Back in 2009 I had done a summer stint as an FAE at a small field office in Long Island with the company which got me interested in working in the semiconductor industry.

Life on the road as an FAE didn't appeal to me after spending my college years constantly moving around for co-ops, so my former boss set me up with an interview here at the RTP design center. On the way down for the interview, I got stuck in Dulles for the night thanks to some bad weather in Rochester causing me to miss my connection. I wound up getting a bare 3 hours of sleep that night on

## QUESTIONS & ANSWERS

done. Now I'm working with design and test engineers to debug any issues that crept up during validation and implement fixes. Ideally, a board-level change is found because PCB or apps level schematic changes are much easier and cheaper than silicon spins. In conjunction with this work, I'll also refine my reference designs and documentation as well as work with the field on initial customer designs by answering questions and checking over layouts and schematics to make sure everything's optimal for their builds.

Up until the part releases, I'll continue cycling through validation, debug, and customer support as needed, squeezing in documentation when I get a chance too. At any given time, I'm also supporting old parts still in production or, if I'm in a lull with my work, getting pulled onto other chips to help out other apps engineers in a jam.

The last part I released, and my first as the lead apps guy, was the ISL95813, a single phase regulator for Haswell and Broadwell systems. My next part is scheduled for release next year which I can't talk too much about, but it's really cool.

**CIRCUIT CELLAR:** During your time at Intersil, you must have learned some important lessons about professional engineering. Can you share one or two things you took from the experience?

**CARMEN:** Most importantly, good communication skills are key. A large chunk of my job is talking to other engineers and customers across the country and overseas. Their whole interaction with me is through the emails and reports I send out and I want to make sure they're top notch. You don't need to be a poet laureate by any means, but if you come across like a rock head, it will be much harder to get taken seriously and problems will drag out longer than necessary. Proofread your work; make sure you're getting your point across clearly; and tailor your email, report, PowerPoint, whatever, to your audience's level of technical expertise. Study up on how to make a slideshow that won't bore your audience or read a technical writing guide. It can't hurt.

Secondly, document, document, document—even if it's only for your own reference. And keep it somewhat organized so you can find what you need again without too much hassle. Yes, it can help CYA, but also I've saved myself a ton of time not redoing the same derivations or looking back at a difficult test setup I had documented in my notebooks. It's especially nice being able to pull up old data from past

parts to see why the heck we did what we did years later.

**CIRCUIT CELLAR:** Tell us about your most recent electrical engineering project. What did you build and why?

**CARMEN:** Well, I can't talk too much about work since all my projects at the moment are either customer related or under development, but suffice it to say I'm working on a lot of low power, multi-role chips.

Outside of work though for nearly two years now I've been co-hosting a podcast which keeps me plenty busy. The show's called *The Engineering Commons* (<http://theengineeringcommons.com>) and it gets released every other week by myself and three other engineers scattered across the US. It was originally started by Chris Gammell and

*The last part I released, and my first as the lead apps guy, was the ISL95813, a single phase regulator for Haswell and Broadwell systems.*

Jeff Shelton, but when Chris left the show for other projects back in 2013, I threw my hat into the ring when Jeff put the word out he was looking for new co-hosts. We discuss the engineering discipline as a whole rather than focus on any one field and some of our favorite topics include education, the value of co-ops, life in the workplace, and the stories of other engineers we bring on to interview.

The semiconductor field is pretty niche, and so through the show, I get exposed to all sorts of new ideas and philosophies, whether it's from researching a topic when coming up with show notes or hearing the stories of engineers and professors from across the globe. Some of my favorite episodes are the ones while interviewing a guest I barely have to say anything and not just because I hate hearing my voice when I re-listen to an episode! Hearing someone get really into a story and talk about their passion I can't help but get drawn in and become excited myself. All us engineers are alike; no matter the field once you get us going about that tricky bug we finally tracked down, the ridiculous meeting that happened the other day, or those ah-ha moments when a solution just clicks in your head we just can't help but gush and it makes for great content.

I've put out nearly 50 episodes with Jeff, Adam, and Brian, and I can't wait to do the next 50!

**CIRCUIT CELLAR:** On FakeEEQuips.com, you



## QUESTIONS & ANSWERS

write: "In my brief time as an engineer, I've come to realize that I'd much rather spend hours at a bench with an o'scope and an iron looking for the cause of an oscillation than slogging through a pile of code to find a missing semi-colon or parenthesis." Ok, but must you code at least some of the time, right?

**CARMEN:** Surprisingly, no. But that is changing somewhat as the nature of my job evolves with time. Don't get me wrong, I can code, but it's neither my forte nor can I say I like it all that much. I do use Excel, MathCad, and various simulators to do some basic calculations and plot data, but never firmware development or anything along those lines.

That being said, I am learning Python on my own time to do some longer term projects at work I cooked up for myself and I do toy around with at home with an Arduino. I guess I'm not a complete code curmudgeon, but the overwhelming majority of my time is spent on the bench counting nano-Coulombs of charge stored in FETs during their deadtime period or dancing on the edge of a transient response spec to see if just one more cap can come off from the output filter and I'd have it no other way.

**CIRCUIT CELLAR:** We read through your 2013 *EEWeb* blog post, "Capacitor Voltage Ratings in VCORE Applications." Can you provide some component picking tips here for our readers? Perhaps you've learned a few things since writing the post?

**CARMEN:** In regards to regulators, sure, I can

share some quick rules of thumb that have worked for me. Pick the biggest inductor, in terms of physical size, you can for an application. A physically big inductor means you'll have lower DCR (less DC loss, better efficiency) and better thermal performance. If you have a regulator with the power FETs integrated this counts double because with a good layout the inductor can help pull heat out of the IC package and keep the FETs happy while loaded. Picking the actual inductance value is worthy of an article in and of itself as you trade-off between several factors like current ripple, DCM output voltage ripple, and overshoot on load current release.

Another tip is don't cheap out on ceramic caps. All caps are not created equal, and that's actually what inspired my *EEWeb* article. I was seeing a difference in transient performance between two eval boards, one populated in house, one built up elsewhere, and I was in the lab late that night figuring out root cause. Turns out when we built up boards in our lab we used good-quality MLCCs giving better performance for the same number of caps on the output. In my inexperience I never called out a specific part number for the output ceramics in the BOM for the assembly house because I didn't think it mattered. Well, the board house just used whatever was on hand leading to my late night. Who has two thumbs and always specs his favorite caps now? Carmen Parisi, nice to meet you. For space-constrained applications cheap MLCCs could mean another four or five pieces are required to meet Intel specs due to their poor high frequency performance on repetitive load scenarios.

Sadly, as with all such rules, these ones apply until they don't at which point you're probably pulling your hair out trying to get a design to work before you realize it or have some requirement you can't negotiate in your favor. Not that I'd know from experience or anything (he mumbled looking at his shoes).

**CIRCUIT CELLAR:** Tell our readers about the prank circuit gag you pulled on the designer you worked with. And can you share an image of the prank circuit?

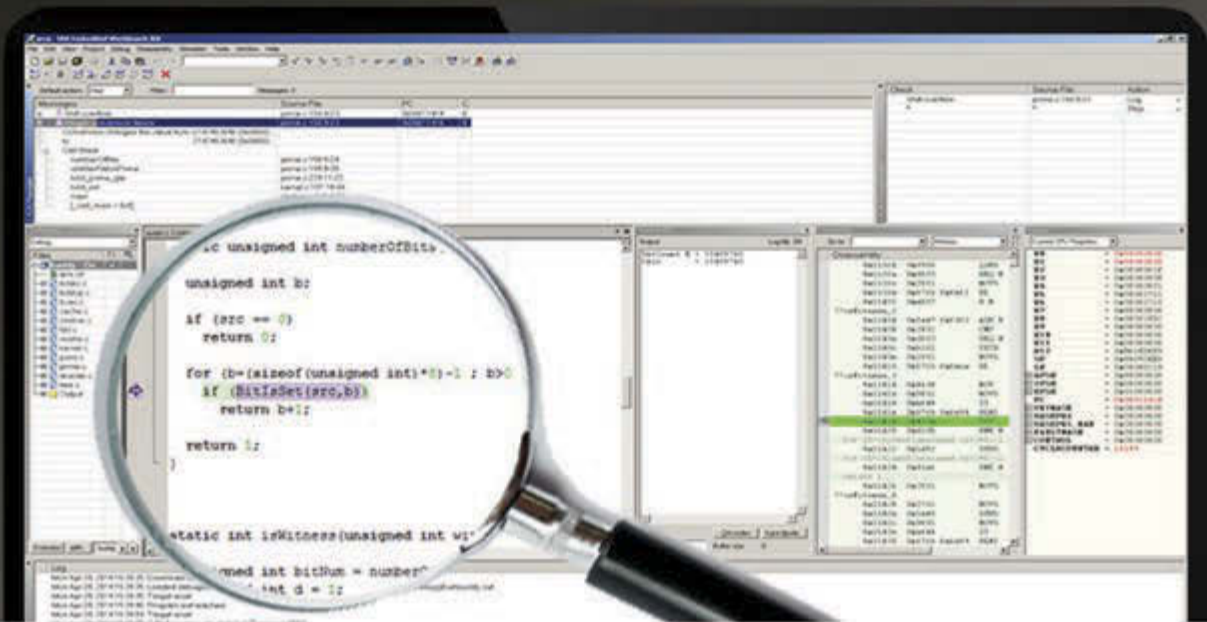
**CARMEN:** A good way through the 813 development I found some problems that ended up being non-issues because I misinterpreted a spec, had a test setup issue, or made a silly component choice in my design. The designer started ribbing me a bit by immediately calling everything a board issue from that point on. This kind of back and forth goes on all the time between apps and design and it's always good natured in tone. I didn't take it personally and took strides to be more thorough before ringing alarm bells going forward but I couldn't let him



Carmen's jury-rigged circuit

# EXPLORE C-RUN FOR ARM

## Runtime Analysis Simplified



C-RUN is a high-performance runtime analysis add-on product, fully integrated with world-leading C/C++ compiler and debugger tool suite IAR Embedded Workbench.

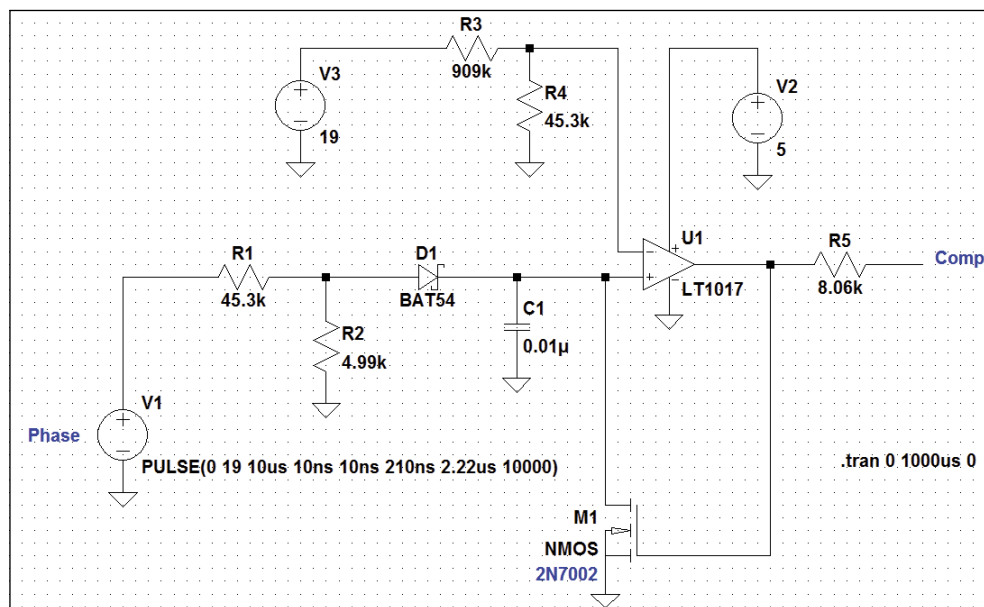
C-RUN performs runtime analysis by monitoring application execution directly within the development environment.

The tight integration with IAR Embedded Workbench improves development workflow and provides each developer with access to runtime analysis that is easy-to-use.

[www.iar.com/crun](http://www.iar.com/crun)



## QUESTIONS & ANSWERS



Carmen's prank circuit diagram

Since he was a thorough designer and liked to regularly pop into the apps lab I actually spent my morning running the tests he asked me to just to keep up the illusion something was wrong if he showed up.

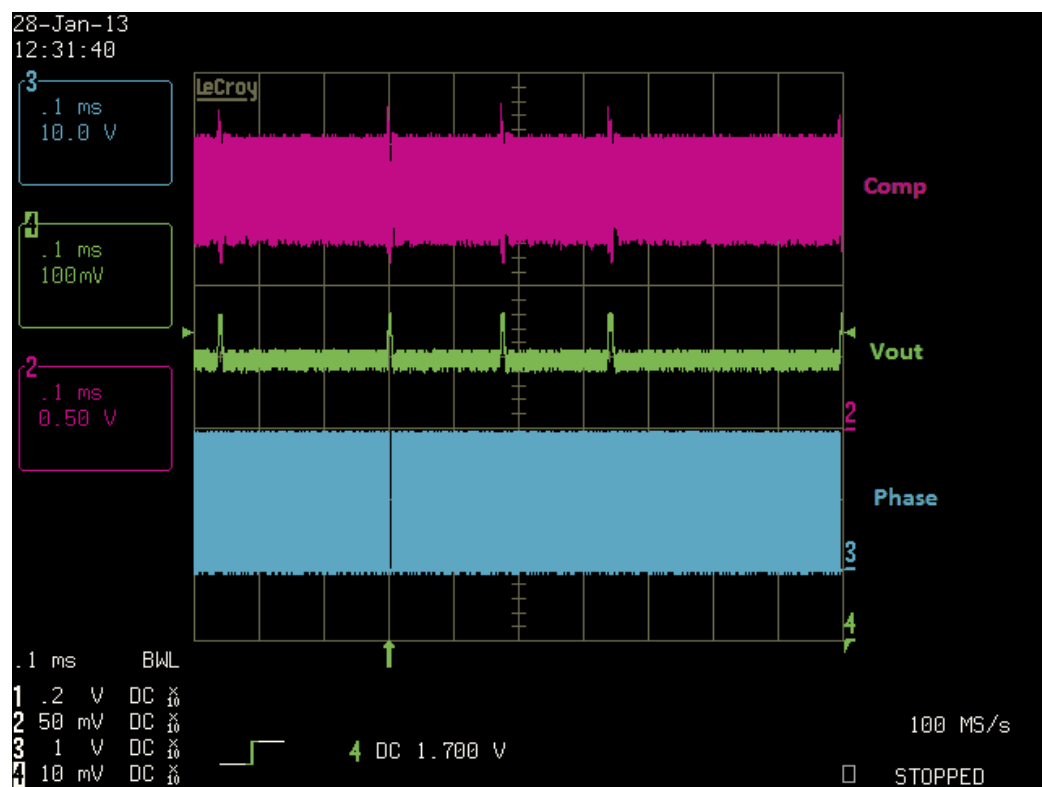
I kept him digging through the schematics trying to find his mistake until mid-afternoon before I brought him in the lab and slowly flipped the board over while telling him I found the error was caused by a parasitic circuit. At this point a couple other engineers who were in on the gag had found reasons to be in the lab for the reveal and we all had a good laugh. The designer took it pretty well, and I even bought him a beer for being a good sport.

get way Scot-free.

With my boss' permission I waited until a slow day came along and rigged up a little circuit to the bottom of the eval board that would overdrive the compensation node of our regulator, propagate through the control loop, and cause seemingly random spikes in the output voltage. I took some waveforms and sent them off to the designer explaining how I found an operational corner that affected regulation we needed to address.

**CIRCUIT CELLAR:** A lot engineering students read *Circuit Cellar*. Do you have any advice for them as they start looking for their first internships or jobs?

**CARMEN:** My boss at GE told me after I got hired for my first co-op what made me stand out at the Career Fair, and during the interview process, was how excited I was taking the plant tour, seeing all the labs, and learning about the job's day to day responsibilities. Looking



Bad circuit waveforms



## QUESTIONS & ANSWERS

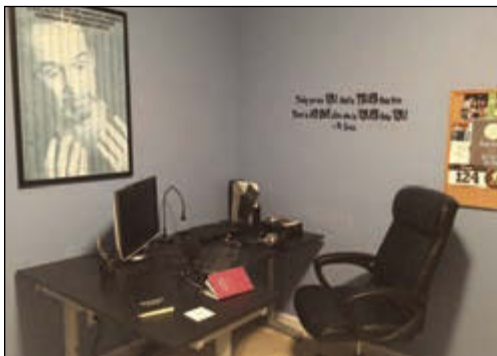
for your first co-op can be tough because you typically don't have any industry experience or upper level design courses to boost your resume. Maybe you don't even know which end of the soldering iron to hold so your passion is really the only card you have to play. Make sure it shines through! Now that I stand on the other side of the booth screening potential candidates I can say it truly does make all the difference. The students that stick in my mind are the ones who in the brief time we talk make me think they really care about the work they'll do and I'm much more likely to recommend them to the managers back at the office.

I talked earlier about the importance of communication and one thing that's helped keep my skills sharp is volunteering at the NC Mini Maker Faire Learn to Solder booth. You really learn how to distill an idea to its core concepts when there's a four year old wielding a hot iron and your fingers are on the front line. I'd definitely recommend getting involved in a similar role to learn how to think on your feet and teach other what you know.

Another piece of advice I have to offer from my experience is no one really minds how many questions you ask them so long as they don't feel like they're doing your work for you. Practically every time (though thankfully not too often) I've gone up to a coworker for help the first thing I'm usually asked is: "What have you done already?" If my answer was along the lines of "Not much," then I'm quickly shuffled out of their office mentally beating myself up because I didn't put in a solid effort. Instead, if I show up with calculations, test results, or anything to show I've tried first, my coworkers are usually much more inclined to help me out.

**CIRCUIT CELLAR: Tell us your thoughts on the future of electrical engineering? Is there a particular technology or area of EE that excites you?**

**CARMEN:** One thing that excites me personally is "The End of Moore's Law," as I've seen it called in various articles. If and when we hit the limit of what we're able to do with conventional transistors, what happens next? Will we have to focus on making firmware and software as efficient as possible or will some new, exotic technology emerge? How will it maintain backwards compatibility with the technology that exists today? Beats the hell out of me, but I can't wait to see it. All I know is I had enough trouble doing semiconductor math in grad school, I couldn't imagine trying to characterize and model one of these new 3-D FETs for the first time. Someone else can lead that revolution. Call me when the circuits need designing!



Carmen's workspace


Outside of technology, I love talking about engineering education, and it's a frequent topic we cover on *The Engineering Commons*. There's a lot of really smart people working to shake up the education system whether it's through expanding the Maker Movement, adding more practical hands-on classes to the curriculum, creating Renaissance Engineers, or doing away with degrees all together. I encourage everyone to listen to some of our episodes on the topic because it's truly fascinating stuff.

**CIRCUIT CELLAR: Tell us about your personal workspace.**

**CARMEN:** A quick aside about the office, the Dr. Seuss quote is a carryover from the previous owners whose son slept in the room.

In my office, where I record my podcast aside from a mic and headphones, there is a small power supply and hand-held DMM. Along with a small bin of scavenged parts that's the extent of my electronics gear. Hardly anything to brag about but I get by. At some nebulous point in the future I'm hoping to get an inexpensive scope too but who knows when that will be.

Other than that I'm also outfitting my garage with a workbench and tool chests so I can keep all my stuff in one place and organized. Finally being able to do basic car repairs is awesome now that I'm not in an apartment and can store things like a jack and more than a standard toolbox worth of gear. I'm fortunate enough to get to do a lot of circuit hacking and crazy one-off designs during my day job debugging silicon (see my recent *EDN* article for one example), so I find the car work and general house repairs a nice change of pace from electronics.

Having just moved into a house at the beginning of the year and so far all of our money has been sunk into making it look like real adults live inside instead of circuit cellar gear. I'm not complaining because I've been able to do a lot of tool shopping to pick up a few essentials and few "essentials" since I've never really gotten over my childhood hang up on tools and can always justify a purchase. 

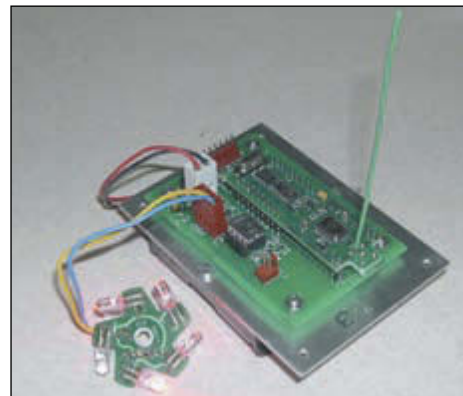
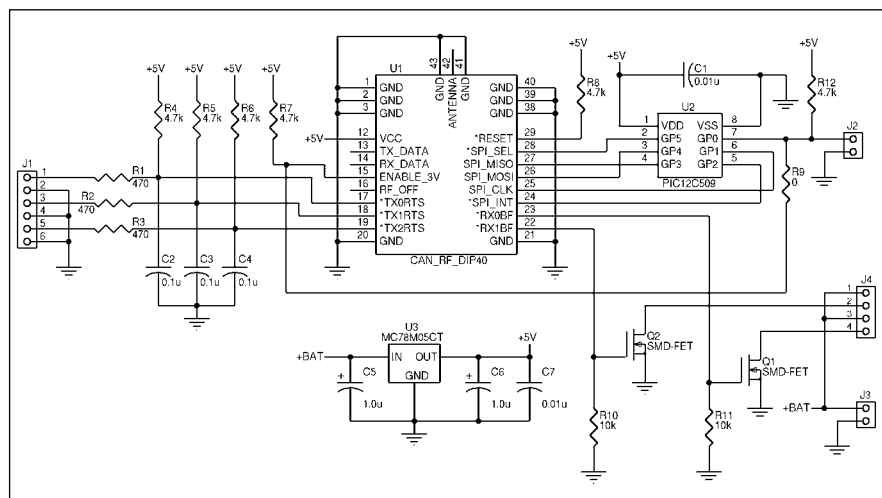
# Wireless Communications

## WIRELESS CAN YARD LAMP CONTROL

**John Dammeyer**

**Circuit Cellar 157, 2003**

In addition to providing a sense of security from intruders, sensor-controlled yard lamps make it safer for you to walk on your property after dusk. But they can pose problems, particularly when a flash triggered by a nocturnal animal awakens you from a dream. Other systems like solar-powered lights have problems, too. Wouldn't it be nice to have more control over the technologies you use? Check out the CANRF module John built for yard lamp control.



The module is mounted on a 2.625" × 3.75" metal plate. The battery holder for the four AA cells is screwed to the bottom, and the antenna is a quarterwavelength piece of solid 22-gauge wire.

The interface to the CANRF module is simple. The processor is marked as a PIC12C509, but you can use a PIC12C508 or PIC12CE674. You can install a 0-Ω R9 to allow for processor control of the power supply to cause the unit to power down and draw only microamps. Or, remove R9 and connect the A/D channel 0 of the PIC12CE674 to a temperature sensor via J2.

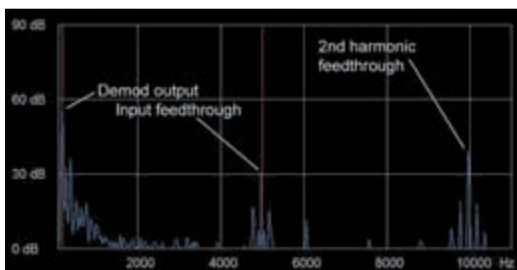
*These articles and others on topics relating to Wireless Communications are available in the CC Webshop. Go to [www.cc-webshop.com](http://www.cc-webshop.com).*

## MODULATION & DEMODULATION

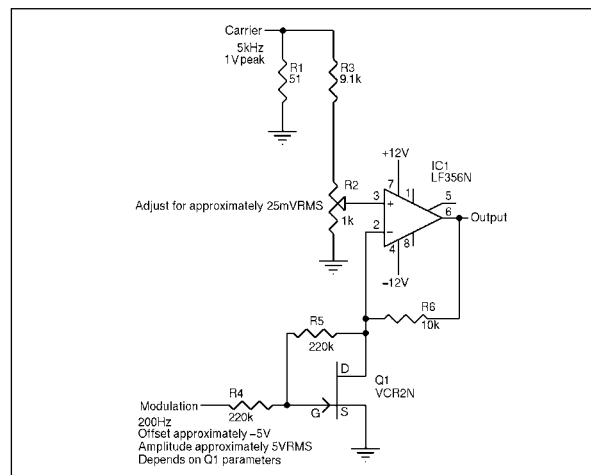
## Ed Nisley

**Circuit Cellar 159, 2003**

Our understanding of radio technology has come a long way since Hertz's nineteenth-century experimentations. In this column, Ed shows you what a simple AM modulator does to a carrier and an audio tone. In addition, he describes how you can demodulate the signal to recover the audio.



A simple RC filter attenuates the unwanted signals and leaves a reasonably clean output. The frequency separation is much higher for real RF, making the filter correspondingly more effective.



An n-channel JFET serves as a voltage-controlled resistor that makes the op-amp gain linearly proportional to the modulation input. The overall circuit is basically a two-quadrant multiplier that produces an AM signal at the carrier frequency.

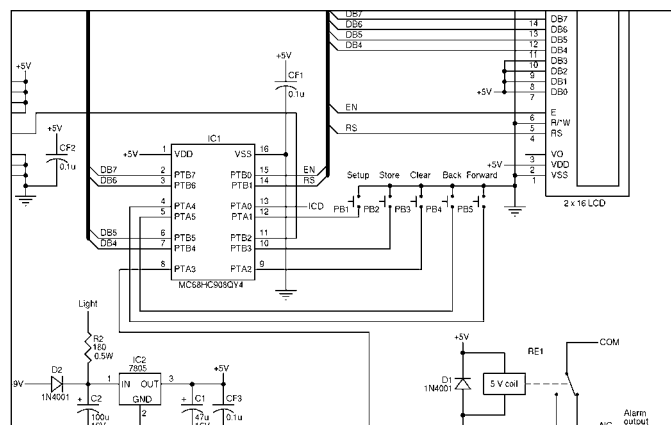
## CC WORLD

## WIRELESS MONITORING SYSTEM

Alberto Ricci Bitti

Circuit Cellar 167, 2004

In addition to providing a sense of security from intruders, sensor-controlled yard lamps make it safer for you to walk on your property after dusk. But they can pose problems, particularly when a flash triggered by a nocturnal animal awakens you from a dream. Other systems like solar-powered lights have problems, too. Wouldn't it be nice to have more control over the technologies you use? Check out the CANRF module John built for yard lamp control.

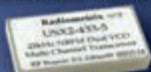


The receiver has few parts. Pull-ups, an oscillator, reset generation, and EEPROM are all included in the MCU. Connections to LCD pins 15 and 16 (backlight) and R2 can vary to suit your LCD's specifications. Older LCDs need a contrast control voltage to be set on pin 3. The relay can trigger a phone dialer when a trap triggers.

The receiver box is recycled from an old soldering station. The receiver is simple enough to be assembled on a prototype board. The display and keyboard are fixed to the front panel with thick double-adhesive tape.

## RF Specialists

"Making your RF ideas into profitable products."

FCC Part 90  
CompliantUSX2 - NBFM Multi-Channel UHF  
Transceiver with Programmable RF Power

## MURS (Multi-Use Radio Service)

SHX1 - Long Range, High Power  
MURS Band Transceiver

## ZigBee Pro

OEM Modules and USB ZigBee Sticks,  
Mesh Networks

## Industrial Bluetooth

OEM, Modules, Wireless Device Servers, RS-232  
Long range options. Low costUltra-Low Power Wi-Fi networking  
module and Eval boardThe AMW006 'Numbat' module is an ultra-low power  
Wi-Fi networking module with full regulatory certification.

## RF Design Services

Prepared to work with your in-house engineers,  
or support your RF project from initial design to implementation.

Industrial • Military • Space • Medical • Smart Grid Metering • SCADA • Lighting Control

LEMONS  
INTERNATIONALTel: 1.866.345.3667  
orders@lemosint.com  
www.lemosint.com

## Low Power ARM Module

## SoM-A5D36

- Atmel ARM Cortex A5 536Mhz Processor
- 4GB of eMMC Flash
- 512 MB of LP DDR2 RAM
- 16MB of Serial Data Flash
- 22 GPIO (3.3V) Lines
- 6x Serial Ports
- 24-bit LCD Controller
- Up to 720P Video
- Touch Controller
- External Address/Data Bus
- Internal Real time clock/calendar
- 4 PWM Channels, 5 Timer/Counters
- 10/100/1000 BaseT Ethernet
- 2x USB 2.0 High Speed Host ports
- 1x USB 2.0 High Speed Host/Device port
- 6 channels of 12 bit A/D (0 to 3.3V)
- 200 pin SODIMM form factor (2.66" x 2.375")



## Industrial Temperature



Designed and manufactured in the USA, the SoM-A5D36 is a System on Module (SoM) based on the Atmel ARM Cortex A5 ATSAMA5D36 processor. This low power, wide temperature ARM 536 MHz SoM utilizes 4GB of eMMC Flash, 16MB of serial data flash, and up to 512MB of LP DDR2 RAM. Like other modules in EMAC's SoM product line, the SoM-A5D36 is designed to plug into a custom or off-the-shelf carrier board containing all the connectors and any additional I/O components that may be required. Qty 1 pricing is \$155. Please contact EMAC for OEM & Distributor Pricing.

[http://www.emacinc.com/products/system\\_on\\_module/SoM-A5D36](http://www.emacinc.com/products/system_on_module/SoM-A5D36)
Since 1985  
OVER  
30  
YEARS OF  
SINGLE BOARD  
SOLUTIONSEMAC, inc.  
EQUIPMENT MONITOR AND CONTROLPhone: (618) 529-4525 • Fax: (618) 457-0110 • [www.emacinc.com](http://www.emacinc.com)



## PRODUCT NEWS

### DUAL ETHERNET MODULE OPERATES AS INDEPENDENT PORTS OR SWITCH

The NetBurner MOD54417 network core module provides 10/100 Ethernet connectivity with two Ethernet ports. The ports can operate independently, each with its own MAC address, or as an Ethernet switch, simplifying network infrastructure (i.e., daisy chaining) by enabling Ethernet devices to connect through it.

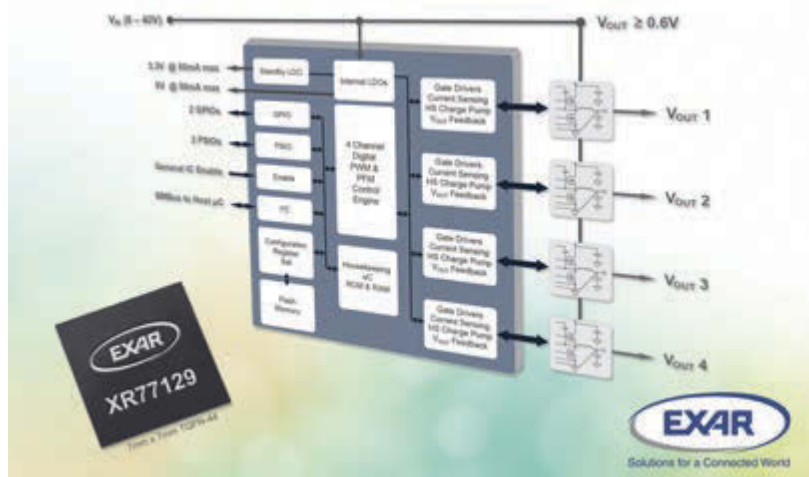
The module is industrial temperature rated ( $-40$  to  $+85^{\circ}\text{C}$ ) and also provides: 8 UARTs, 4 I<sup>2</sup>C, 2 CAN, 3 SPI, 1-Wire, a MicroSD flash card socket, 42 digital I/O, eight 12-bit analog-to-digital inputs, two 12-bit digital-to-analog outputs, and five PWM outputs. Wireless 802.11 b/g/n communication is available with the optional Wi-Fi add-on.

The NetBurner Network Development Kit (NNDK) provides a complete software and tools package including the Real-Time Operating System, full featured TCP/IP Stack, Web Server, DHCP Server, Eclipse development environment, C/C++ compiler and debugger. The NNDK is focused on ease of use and you will have your first custom program running within a few hours of receiving the kit. The price of the MOD54417 ranges \$94 to \$129.

**NetBurner**  
[www.netburner.com](http://www.netburner.com)



### QUAD OUTPUT PROGRAMMABLE UNIVERSAL PMIC



Exar Corp. recently announced the XR77129, a quad output programmable universal PMIC with an input operating voltage range of 6 to 40 V. Its patented control architecture is well suited for 40-V inputs using a 17-bit wide PID voltage mode VIN feed forward architecture. This controller offers a single input, quad output, step-down switching regulator controller with integrated gate drivers and dual LDO outputs. It can also monitor and dynamically control and configure the power system through an I<sup>2</sup>C interface. Five configurable GPIOs allow for fast system integration for fault reporting and status or for sequencing control.

The XR77129 can be configured to power nearly any FPGA, SoC, or DSP system with the use of Exar's PowerArchitect and programmed through an I<sup>2</sup>C-based SMBus compliant serial interface. PowerArchitect 5.2 has been upgraded to support the additional capabilities of the XR77129 including output voltage ranges beyond the native 0.6 to 5.5 V with the use of external feedback resistors. The XR77129 wide input

voltage range, low quiescent current of 450  $\mu\text{A}$  (standby) and 4 mA (operating) make it a logical choice for a wide range of systems, including 18 to 36 VDC, 24 VDC or rectified AC systems used in the industrial automation and embedded applications.

The XR77129 is available now in an RoHS-compliant, green/halogen-free, space-saving 7 mm  $\times$  7 mm TQFN. It costs \$9.95 in 1,000-piece quantities.

Summary of features:

- 6 to 40 V input voltage
- Quad channel step-down controller
- Digital PWM 105 kHz to 1.23 MHz operation
- SMBus-compliant I<sup>2</sup>C interface
- Supported by PowerArchitect 5.2 or later

**Exar**  
[www.exar.com](http://www.exar.com)

# SUPERIOR **EMBEDDED** SOLUTIONS



**DESIGN YOUR SOLUTION TODAY**  
**CALL 480-837-5200**

[www.embeddedARM.com](http://www.embeddedARM.com)

## **Touch Panel Computers** Panel Mount or Fully Enclosed



Series start at  
**\$369**  
Qty 100  
**\$409**  
Qty 1

### **Features can include:**

- 5, 7, and 10 Inch Touchscreens
- Fanless Operation from -20 °C to 70 °C
- Up to 1 GHz ARM CPU, 2 GB RAM, 4 GB eMMC Flash
- Ethernet, USB, DIO, CAN, RS-232, Modbus, SPI
- Optional Cellular, WiFi and XBee Radios
- Supports Android & Linux with Fast Boot Times

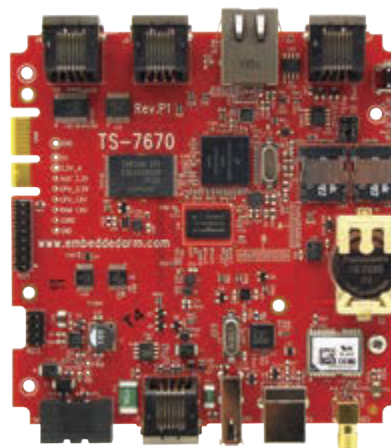


TS-TPC-8390-4800 Rear View



TS-TPC-8390-4800 Angled View

## **NEW! TS-7670 Industrial Computer** GPS Radio and Cellular Modem



Pricing starts at  
**\$168** Qty 100  
**\$129** Qty 1



low cost plastic  
enclosure available

### **Features:**

- 454MHz ARM CPU
- Up to 256MB RAM
- 2GB NAND Flash
- 2x mSD Card Socket
- 1x 10/100 Ethernet
- 1x USB Host
- 4x DIO, 2x CAN
- 2x COM, 1x RS-485
- 1x Battery Backed RTC
- 1x Temperature Sensor

### **Benefits:**

- Low power with 10mW sleep state
- -40 to +85C, 100% soldered-on components
- Easy development w/ Debian and Linux 2.6
- Boots quickly to your Embedded Application
- Guaranteed available until 2025

### **Coming Soon:**

**TS-7680: Like the TS-7670 w/ WiFi & Bluetooth**



We've never discontinued a product in 30 years



Embedded systems that are built to endure



Support every step of the way with open source vision



Unique embedded solutions add value for our customers

[www.embeddedARM.com](http://www.embeddedARM.com)

## PRODUCT NEWS

### NEW STM32 MICRONTROLLERS IN SMALL MEMORY SIZES

Nordic STMicroelectronics's new STM32F446 microcontrollers feature ARM Cortex-M4 based processing combined with 256- or 512-KB on-chip flash memory options. In addition to using STMicro's ART Accelerator, the microcontrollers feature smart architecture, advanced flash technology, and an embedded ARM Cortex-M4 core to achieve a performance of 225 DMIPS and 608 CoreMark at 180 MHz executing from embedded flash.

Key features include:

- At 180 MHz, the STM32F446 delivers 225 DMIPS/608 CoreMark performance executing from flash memory with 0-wait states. The DSP instructions and the floating-point unit expand the range of addressable applications.
- Using a 90-nm process, the current consumption in Run mode and executing from flash memory is as low as 200  $\mu$ A/MHz at 180 MHz. In Stop mode, the power consumption is 50  $\mu$ A typical.
- Two dedicated audio PLL, SPDIF input, three half-duplex I<sup>2</sup>S, and two serial audio interfaces (SAI) supporting full-duplex I<sup>2</sup>S as well as time division multiplex (TDM) mode.
- Up to 20 communication interfaces (including 4x USARTs plus 2x UARTs running at up to 11.25 Mbps, 4x SPI running at up to 45 Mbps, 3x I<sup>2</sup>C with a new optional digital filter capability, 2x CAN, SDIO, HDMI CEC and camera interface)
- Two 12-bit DACs, three 12-bit ADCs reaching 2.4 MSPS or 7.2 MSPS in interleaved mode up to 17 timers: 16- and 32-bit running at up to 180 MHz
- Easily extendable memory range using the flexible 90-MHz memory controller with a 32-bit parallel interface, and supporting Compact Flash, SRAM, PSRAM, NOR, NAND and SDRAM memories

- Cost-effective NOR flash extension with the 90-MHz Dual quadSPI interface supporting memory-mapped mode
- STM32F446 samples are now available for lead customers. Volume production is scheduled for Q1 2015 in packages from a tiny WLCSP81 measuring 3.728 x 3.85 mm to a 20 x 20 mm LQFP144 with 256- or 512-KB flash memory, all with 128-KB SRAM.

Pricing starts at \$3.75 for the STM32F446RC in a 64-pin LQFP64 package with 256-KB flash memory and 128-KB SRAM for orders of 10,000 units.

**STMicroelectronics**  
**www.st.com**



### NEW EZ APP LYNX LIBRARY FOR SMART BLUETOOTH SENSORS

CCS C-Aware IDE now includes the EZ App Lynx library. Quickly create a Bluetooth wireless sensor, or controller, that may be viewed or managed on a paired mobile device using the EZ App Lynx Android app.

The free EZ App Lynx Library was created to shorten the design time for smart Bluetooth app development. With EZ App Lynx, and no required hardware or software expertise, the library removes the barriers to entry for smartphone app developers who want to take advantage of a growing number of Bluetooth enabled smartphones and tablets. The new library allows for any GUI, on the App, to be created at run time from a PIC program. The library offers many useful sensor interface components, which allow for: Status Bars, Gas Gauges, Sliders, Buttons, Text Fields, and more.

**EZ App Lynx Library Features and Advantages:**

- No app design knowledge required
- Source code libraries included with all CCS C Compilers
- Included with maintenance update download
- EZ App Lynx App



- Available for Android in Google Play Store (iOS available soon)
- Build your own EZ App Lynx App in minutes with simple C library calls on the PIC
- Quick and easy prototyping

**CCS**  
**www.ccsinfo.com**



## CLIENT PROFILE

# EarthLCD.com

[www.earthlcd.com](http://www.earthlcd.com)

3184 Airway Ave., Suite J, Costa Mesa, CA 92626

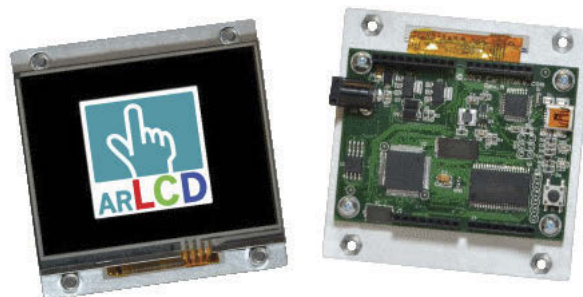
**CONTACT:** Randy Schafer (randy@earthlcd.com)

**FEATURED PRODUCTS:** The ezLCD-405 5.6" VGA Smart, Touchscreen can be used as an intelligent display or as a standalone device, easy-to-use, command driven programmable firmware environment, and easy-to-customize your firmware with our free tools. The ezLCD-405 is based on the STMicro STM32F429 ARM M4 microcontroller. Also an optional uLinux BSP is available. The arLCD by EarthMake.com is a Smart Touchscreen LCD combining an 3.5" Touchscreen, ezLCD GPU and the Arduino Uno designed for the maker market.

**WHY SHOULD CC READERS BE INTERESTED?** arLCD is a full smart ezLCD GPU with the Arduino Uno R3 on the same PCB in a thin, easy-to-integrate package. It can be used in many applications such as thermostat control, lighting controls, home security, audio control, water level gauge, robotics, and industrial automation. The arLCD combines the best of ezLCD-3xx and the Arduino UNO. The arLCD is not just an LCD and you should not confuse it with a snail's pace 2.8 LCD shield that uses almost all your I/O pins!

**SPECIAL OFFER:** Receive a 10% discount off your first EarthLCD product! Promo code: CIRCUITCELLAR

ezLCD + Arduino = Touchscreen Magic!



Circuit Cellar prides itself on presenting readers with information about innovative companies, organizations, products, and services relating to embedded technologies. This space is where Circuit Cellar enables clients to present readers useful information, special deals, and more.



Sign up today and SAVE 50% • Sign up today and SAVE 50% • Sign up today and SAVE 50%

## Now offering student SUBSCRIPTIONS!

When textbooks just aren't enough, supplement your study supplies with a subscription to *Circuit Cellar*. From programming to soldering, robotics to Internet and connectivity, *Circuit Cellar* delivers the critical analysis you require to thrive and excel in your electronics engineering courses.

Sign up today and  
Sign up today and SA

Sign up today and SAVE 50%



[www.circuitcellar.com/subscription](http://www.circuitcellar.com/subscription)

# Electronics Testing Platform (Part 1)

## Base Unit Construction

Gerard's affordable, multifunctional test instrument would be a handy tool for anyone focused on rapid product development. In this article, he explains how he engineered the base unit, which is a platform for the development of test instruments. Next month he details the construction of a plug-in sweep generator board.

By Gerard Fonte (US)

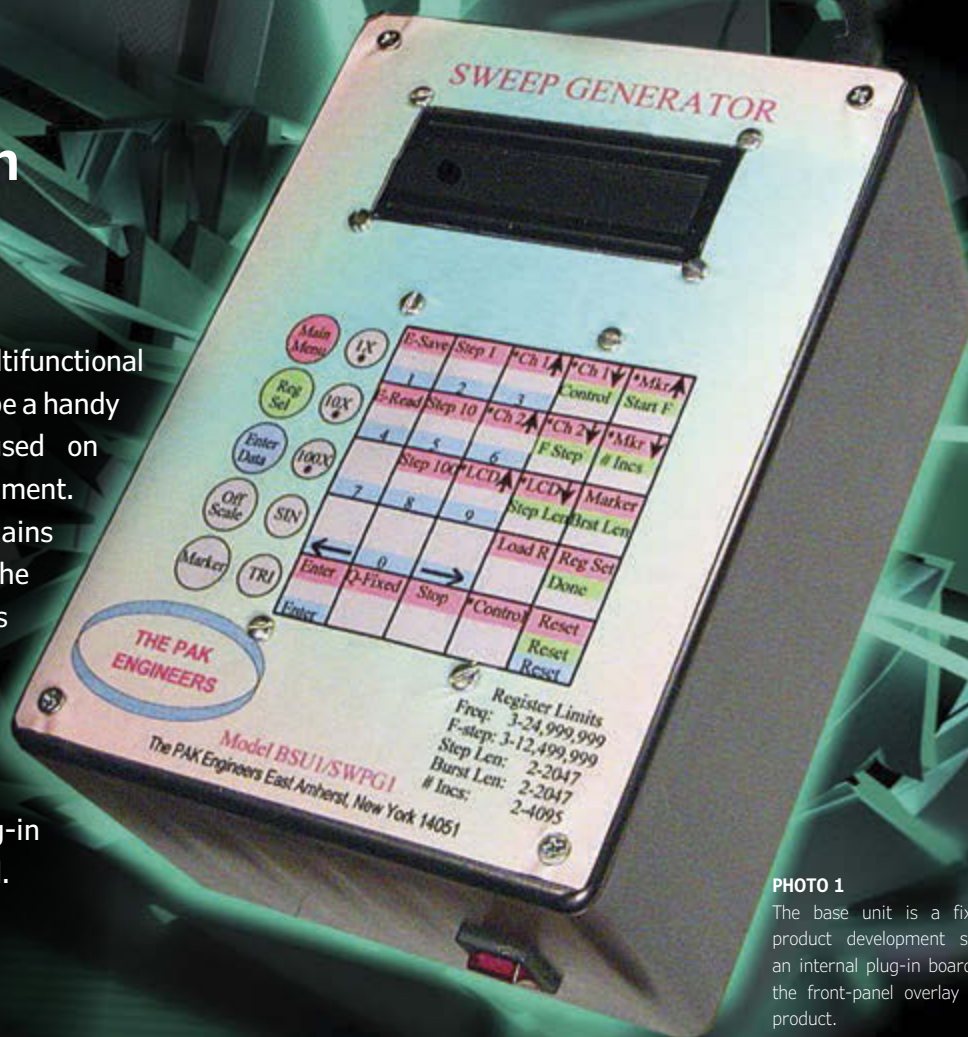


PHOTO 1

The base unit is a fixed-form-factor product development system. Adding an internal plug-in board and changing the front-panel overlay creates a new product.

This is a bit different from most *Circuit Cellar* projects. It's a professionally designed, open-source hardware and software product. I got tired of duplicating many facets of embedded computers in my instrumentation applications. So, I created a standard platform (base unit) that drives the typical input and output requirements of any instrument. This includes a standard  $2 \times 16$  LCD, 25 switches, and 10 LEDs (see **Photo 1**). Note that this is not a prototyping system. This is designed to go directly into production. The base unit hardware design and software are free for noncommercial and commercial purposes. (Use at your own risk!) All specialized plug-in boards that will follow are for noncommercial use only.

The object of the base unit is to provide a platform for the development of a series of very inexpensive, yet high-performance, professional test instruments. The first plug-in instrument is a sweep generator. It's digitally synthesized, sweeps from 3 Hz to 25 MHz (Nyquist limit), has 0.3- and 3-Hz steps with 50 PPM accuracy, differential outputs, operates in burst or continuous modes, low distortion sine wave (0.1%), triangle wave, square wave, linear or log sweep, fixed frequency and single-step modes, internal and external control, EEPROM save of up to eight set-ups, directly drives 51- $\Omega$  loads and 8- $\Omega$  speakers, and has a lot of other nice features. It also has a true marker/cursor (not marker generator) that precisely



identifies the frequency at any point in the sweep on your oscilloscope. But that's the teaser for the next installment. (Other upcoming plug-in test instruments include a micro-ohmmeter—yes, it measures millionths of ohms—a pico-ammeter, an AC mains analyzer, and others.)

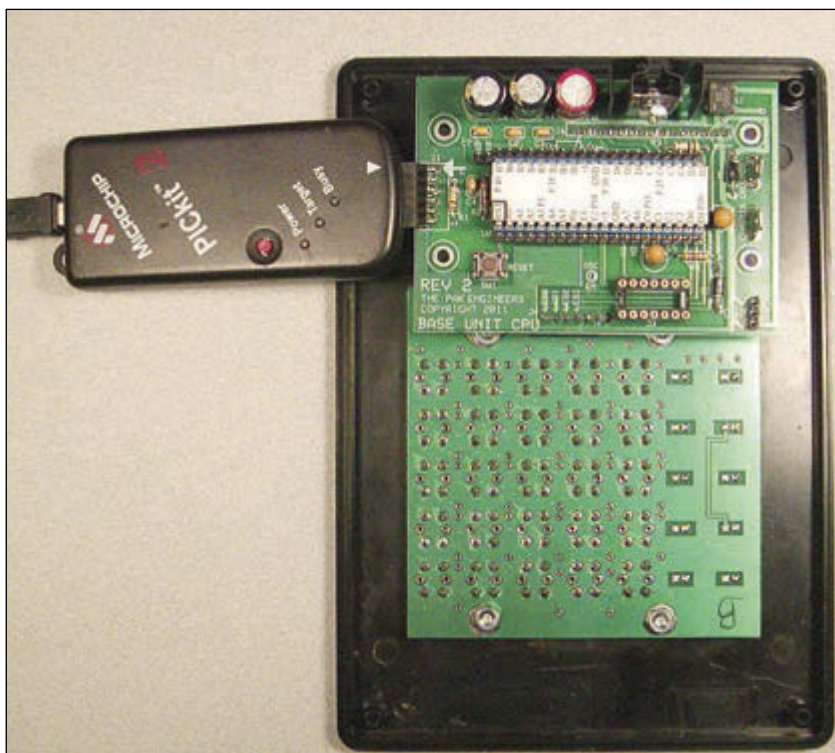
The base unit is also different from other development systems. It's designed around reusable hardware and software modules. This makes product development much faster. The 2 × 16 character LCD, 10 LEDs, and 25 switches are fixed in hardware. This means that the driver software for these is also fixed. There is absolutely no rewriting of code to change pin assignments and so on. Additionally, the hardware was designed to use the minimum number of microprocessor resources. The LCD uses seven I/O pins, the LEDs use four pins, and the switches use only one pin. That leaves 19 available I/O pins, as well as most of the special features.

Most products will not need all of these switches and LEDs. We'll see later that it's a simple matter to fabricate a professional-looking overlay to mask out these unneeded components. The software supports larger LCD modules, if needed.

## PIC MICROPROCESSOR

I chose a Microchip Technology PIC16F887 8-bit processor for a number of reasons. It has several nice features: 8-KB program space (flash memory), 368 bytes of RAM, 256 bytes of EEPROM, a 10-bit ADC, two pulse-width modulators (PWMs), three timer/counters, an internal oscillator, and more. It only costs about \$3.50. But the major reason for choosing the PIC16F887 was the tremendous amount of free support directly from Microchip. MPLAB from Microchip is a complete development system that has everything you need to get up and running, including: assembler, editor, debugger, programmer (software), project manager, and more. It's easy to use and very well documented. And it supports all devices from Microchip.

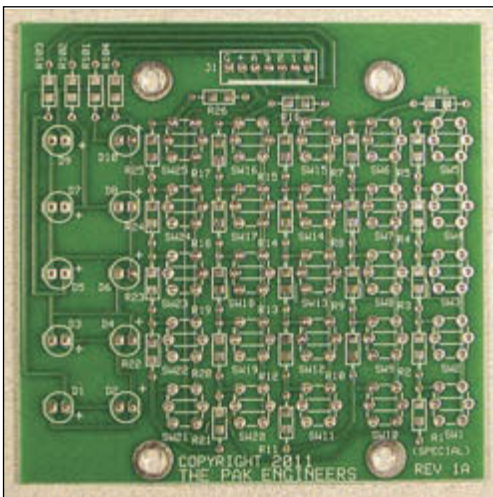
The only hardware you need is their programmer. The PICKit-2 costs about \$35 (PG464120) and is widely available (see **Photo 2**). This programs nearly all of their products directly. It also allows in-circuit, real time debugging. This is an absolutely great feature. You can put a break-point in your program and then run the hardware with real inputs and outputs. When you get to your break-point, the system stops and displays any variables you want. Then you can single-step through the code and watch the variables change. It's much, much easier, faster and more accurate than running a simulation.



**PHOTO 2**  
PICKit-2 programmer and debugger

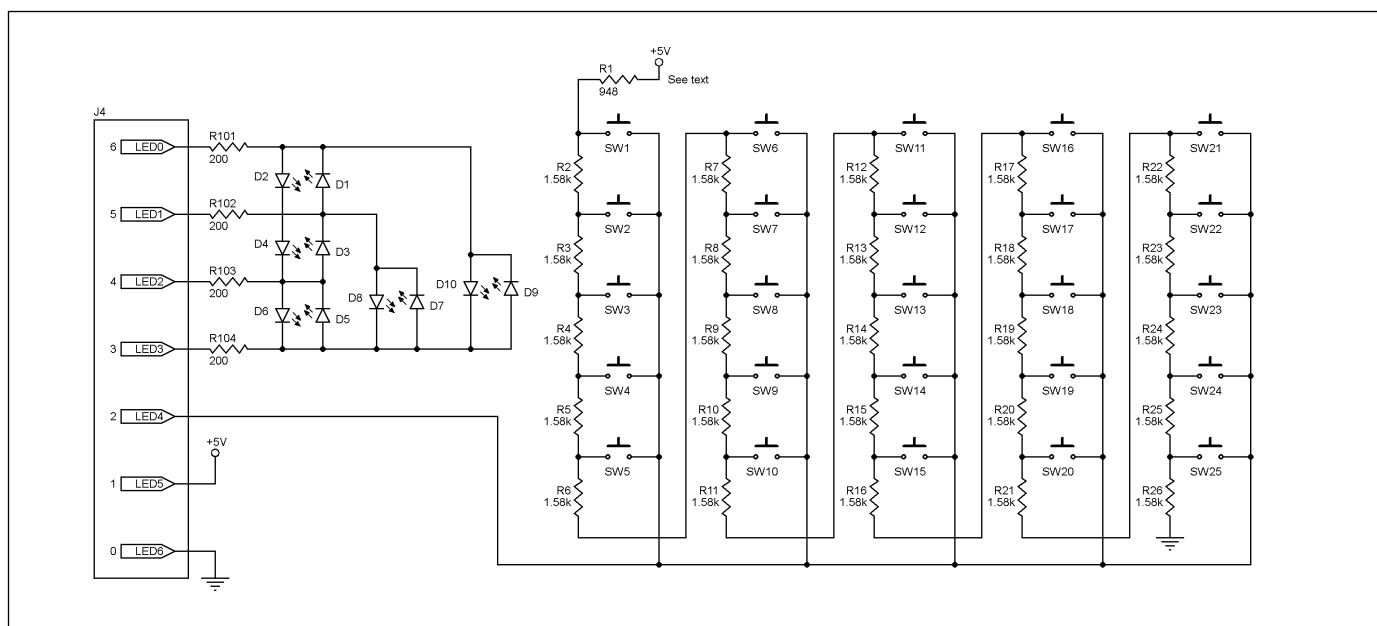
## I/O BOARD

The base unit is comprised of four components: an LCD, a microprocessor board, an I/O board, and a case/power supply. The I/O board was designed to be as flexible as possible. You can use SMT (1206 size) LEDs or through-hole LEDs. You can use SMT resistors (1206 size) or through-hole resistors. And finally, you have the choice of using 6-mm square (0.24") four-pin switches or 6-mm, 2-pin switches (see **Photo 3**).



**PHOTO 3**  
How to use one analog pin to read 25 switches and four pins to drive eight LEDs



**FIGURE 1**

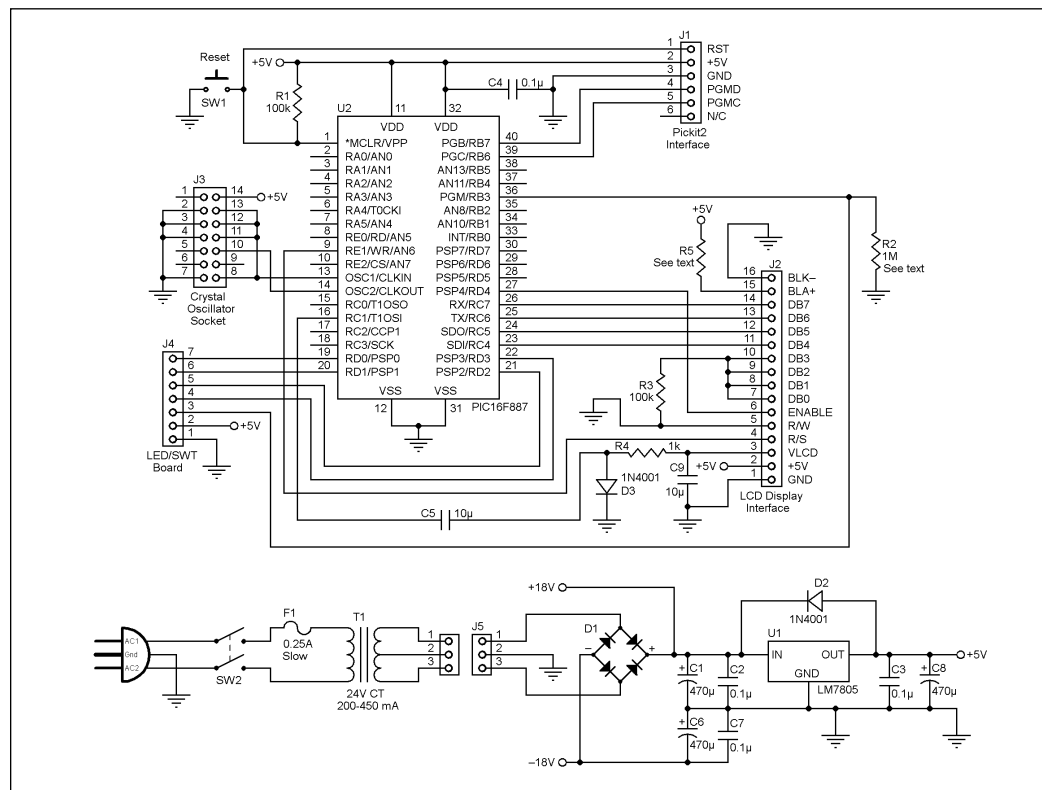
Twenty-five switches are read by a single pin.

I chose 25 switches because it seemed that the typical 4 × 4 matrix of 16 switches might be too small. It seems very possible that future modules would require a hexadecimal input, so a 16-switch keyboard wouldn't have anything left over for other functions. Normally a 5 × 5 matrix of switches would require 10 pins on the microprocessor. Since this is way too many pins to commit for just switches, I used a different approach.

**Figure 1** shows how these 25 switches are

read by a single pin. A simple resistive divider/ladder was used. It connects to an analog I/O pin of the microprocessor. Each switch closure will provide a particular voltage so it is pretty easy to figure out in software which switch was pressed.

There is a little trick used to make the software easier. Since we are only using the top 8 bits of the 10-bit ADC, the voltage range is 256 counts. It would be nice if each switch voltage was exactly 10 bits apart instead of

**FIGURE 2**

This is a schematic of the microprocessor board. R2 does double duty. C5 and C9 are nonpolarized electrolytics.

10.24 counts (i.e., 256 counts/25 switches). The trick is to add a 26th resistor that is 0.6x the other resistors. In this way there will be 250 counts for the 25 resistors and 6 counts for the 26th resistor for a total of 256 counts. Placing this 26th resistor at the top of the ladder “removes” 6 counts from full-scale. Therefore, the switch values will start at 0 (no switch pressed) and go by 10s to 250. It’s very convenient and it only costs one additional resistor.

The actual resistor values aren’t critical but should be in the 500  $\Omega$  to 2 k $\Omega$  range and have 1% tolerance. For best operation, the A/D expects to see less than 10-k $\Omega$  input impedance. Since both VCC and ground are defined as “zero” ohms impedance, the highest impedance will be at the center of the resistor ladder. This is about 12 k $\Omega$  for 1-k $\Omega$  resistors. But we aren’t concerned with absolute precision and we are only using 8 bits of a 10-bit ADC. So, using 2-k $\Omega$  resistors will work acceptably well. I used 1.58-k $\Omega$  resistors. This made the 26th resistor 948  $\Omega$  ( $0.6 \times 1.58$  k $\Omega$ ), but a 1-k $\Omega$  resistor worked fine. An additional 1-M $\Omega$  (R2) to ground is needed to drain off any floating charge on the input pin when no switch is pressed (see **Figure 2**).

There is the valid question of the approach’s reliability. Can we be sure that different manufactured units will be consistent? Or will there be a need to tweak the software and hardware? To answer the questions, I built eight I/O boards—six used through-hole resistors and two used SMT resistors. Two used “948” ohm resistors (750 and 200  $\Omega$  in series) for the 26th resistor (both through-hole) and six used 1-k $\Omega$ , 1% resistors. I ran a statistical analysis to calculate the mean and standard deviation of all the 25 possible switch-press resistances (without the 1-M $\Omega$  pull-down resistor). The worst-case switch (at the end of the resistor string) had a standard deviation of only 82  $\Omega$ . Since the difference between switch resistances is 1,580  $\Omega$ , this corresponds to a difference of over 19 standard deviation units or “19 sigmas” between switch values. The “one failure in a million” quality control procedure is only “6 sigmas.” The resistance ladder is very reliable.

But what about A/D conversion error? Is this a possible source of concern? The 10-bit ADC can resolve 1,024 steps. The difference between adjacent switches is 40 of these steps. If you sum all the worst possible errors in the ADC, they only come to six steps. The A/D conversion error is not a concern, either.

Ten LEDs are used on the I/O board. Again, using 10 independent pins for these

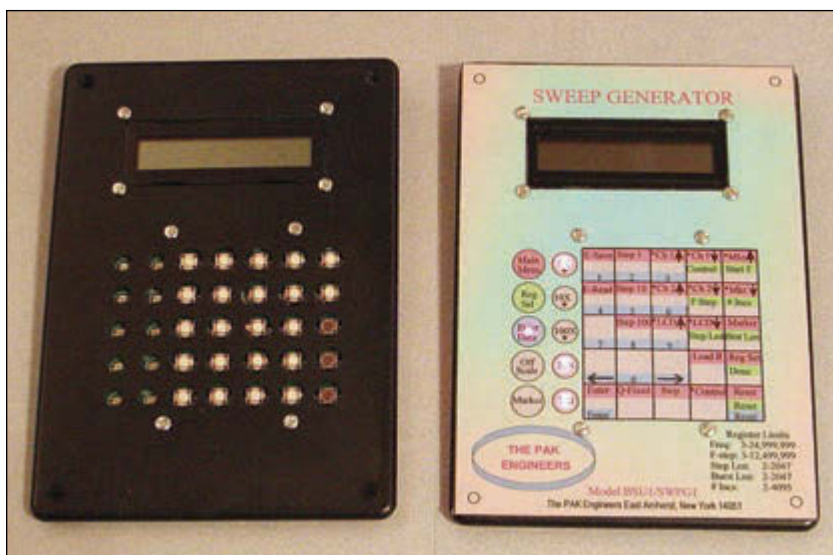
is too wasteful of microprocessor resources. I used a “Complementary Drive” approach from the Microchip’s application note TB029. Just four pins are needed to drive up to 12 LEDs. The key to understanding this circuit is to realize that only two pins are set as outputs at any time: one high and one low. The other two lines are set as inputs and are effectively disconnected from the circuit. By properly choosing which pin to drive high and which pin to drive low, any particular LED can be turned on. In general, multiple LEDs cannot be on at the same time; however, there are special cases. Having only one LED on at a time is not a particular problem. Many applications require multiplexing the LEDs to make it appear that several are on at the same time.

The four LED pins and the three pins for the switches (VCC, GND, and switch voltage) are brought out to a seven-pin header strip (J4). This plugs into the microprocessor board. The four LED lines are connected to the low-order Port D pins on the microprocessor. I chose these pins because they had no other function than digital I/O. This leaves the multipurpose pins available for future applications.

The choice for using bit 3 of Port B (which can be configured as an analog input) for the switch analog input is special. Port B bit 3 is also the “Low-Voltage Program” pin. It is not used by the regular PICKit-2 programmer. The default state for this pin is to activate the low-voltage option. So, if you don’t know (or remember) to properly configure this pin at start-up, you can enter the “Low Voltage Programming State” if this pin floats high (which is very common). The result is a microprocessor that appears completely

## ABOUT THE AUTHOR

Gerard Fonte is the principal engineer at The PAK Engineers and has nearly 30 years of hands-on experience that includes missile guidance systems, electronic warfare, gravity navigation, and projects that “don’t exist.” He has well over 100 publications. Gerard was awarded the Engineers’ Council Outstanding Engineering Merit Award in 2006 for his work on Egyptian pyramid construction. He holds a BA degree in Psychology and a MS degree in Natural Science.



**PHOTO 4**

Before and after adding the laminate overlay. Note that the high-intensity LEDs shine through.



PHOTO 5

The back of the LCD showing added pins (mates to J2). Note the nylon spacer nuts. The top of I/O board is shown and metal spacer-nuts are below board (not visible). I/O board connector mates to J4.

dead. My solution is to always pull this pin low to eliminate that possibility. Since I already have a pull-down resistor on that pin, it's very convenient to use it for the switch input.

By using inexpensive "tactile" switches and a little ingenuity, you can make a "membrane" switch panel. The basic method for this is shown in **Photo 4**. The front panel is then covered with a paper overlay which is

"laminated" on both sides with 2" wide, clear, cellophane tape used for packaging. (This is available at your local office-supply store.) You can be as creative as you like. Double-sided tape holds the membrane in place.

The top of the switch actuator should be even with the top of the plastic cover or perhaps slightly higher (according to your taste). In order to do this, the holes must be big enough to partially pass the switch body (unless you choose a switch with a long actuator). A small spacer (a 4-40 nut) will work for the switches shown (which are fairly standard). Different switches may require different spacers. Note that if you mount the through-hole resistors on the same side of the PCB as the switches you must be careful to make them lie perfectly flat. Otherwise, they can be too high and cause a problem with the spacing. If you use a PCB with plated through holes, you can mount the resistors on the bottom of the board and eliminate this concern.

The LEDs can be the standard T1 3/4 size, but they will protrude from the cover. You can also use SMT LEDs. If you go this route, use super-bright parts (600–1,000 Mcd). These are so bright that they will shine right through the paper and tape overlay.

## LCD

Often LEDs simply do not provide enough information. For that reason, an LCD is included. A commonly available 2 × 16 display was chosen (Hantronix HDM16216). It has the connector above the display. In addition, it is 3.15" × 1.42" (80 mm × 36 mm) and uses either a 4- or 8-bit interface. It turns out that these LCDs (incorporating an HD44780 controller) are incredibly generic. I referenced an AND LCD Products catalog from 1988 (an AND-491 display), a Microchip Application Note written in 1997 (using a Hitachi LM032L display), and a couple of recent magazine articles. All displays had identical pinouts and identical software commands. The only difference I was able to find was a small timing variation during the initialization of the LCD. The 4-bit interface is used here to save I/O pins for future modules.

Deciding which seven pins to connect to the LCD wasn't simple. I wanted to leave as many multipurpose microprocessor pins available for future use, especially the analog pins. I wired the R/W interface line directly to ground to save an I/O pin, so the LCD becomes write-only. I can't see any future need to read the data in the LCD memory. The high-order Port C pins had secondary uses for serial communications which do not seem that useful for test and measurement designs. The four LCD data connections went

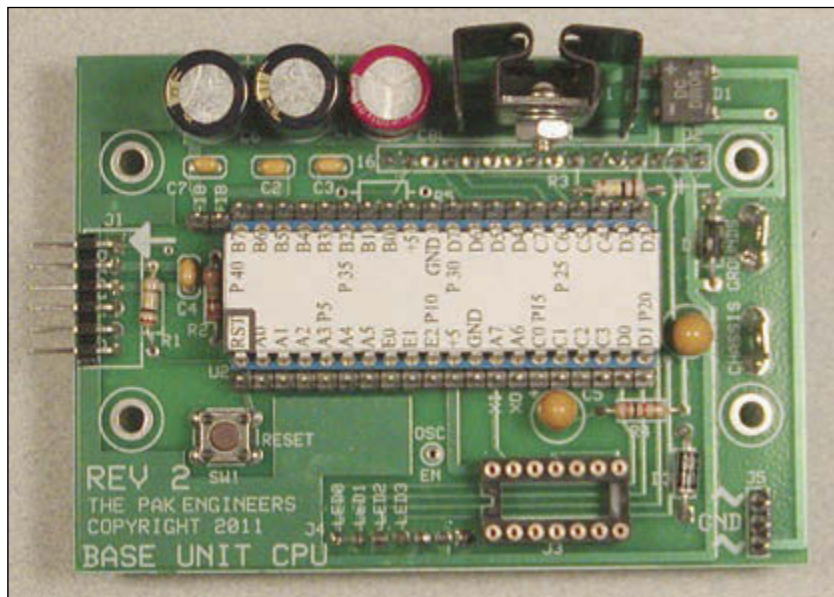


PHOTO 6

Top view of microprocessor board. Header pins around microprocessor mate to future plug-in modules. Mounting holes align with LCD screws but friction from J2 and J4 pins seems sufficient.

## SOURCES

HDM16216 LCD

Hantronix | [www.hantronix.com](http://www.hantronix.com)

PIC16F887 Microcontroller

Microchip | [www.microchip.com](http://www.microchip.com)



[circuitcellar.com/ccmaterials](http://circuitcellar.com/ccmaterials)

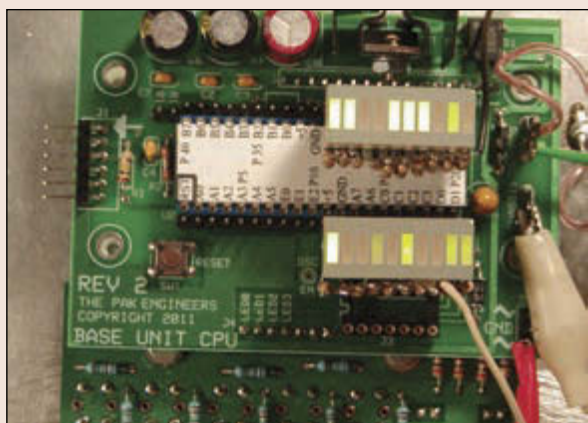


there. However, the low-order pins are used for the internal timers. These are useful, so I had to connect the remaining two LCD digital lines to different ports. The last free plain, digital I/O pin was Port D bit 4. I connected the LCD Enable line there. The other Port D

pins were also associated with the timers so I didn't want to use them. I ended up using Port E bit 1 for the R/S LCD line. It's a plain analog/digital line with no other special functions. This leaves all of the special-function analog pins available as well as all the timer pins.

## EMBEDDED DEVELOPMENT TIPS AND TECHNIQUES

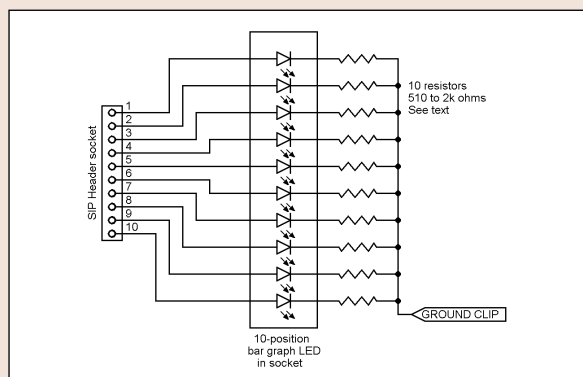
With all those pins, it's easy to lose track of which pin is which. The simple answer is to label them. Use your word-processor to create the label you want. Mine shows the port identifications, power, and every fifth pin number (see **sidebar Photo 1**). If you can't create a list to size, make it an image and shrink it as needed. Print it, cut it out, and attach it to the chip with double-sided tape. You will be amazed at how much time and grief it saves.



**SIDEBAR PHOTO 1**

LEDs and labels make things a lot easier. (Full-on green LEDs appear blue for some unknown reason.)

Using your oscilloscope or voltmeter to measure the output of a pin gets old real fast. Instead, build a simple 10-pin logic-state display (or two). All you need is a 10-segment bar graph LED, 20-pin socket, 10 resistors, a 10-pin SIP header, and a ground clip (see **sidebar Figure 1**). The resistors can be a SIP network instead of individual parts (but the most they come in is nine so you'll still need one discrete resistor). The value of the resistors

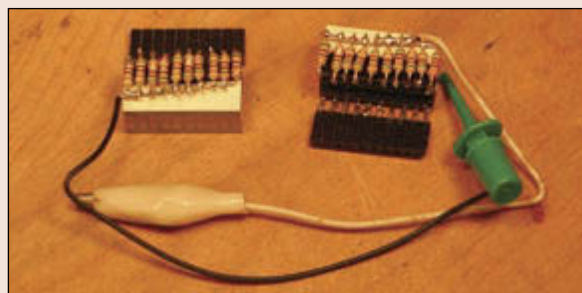


**SIDEBAR FIGURE 1**

The poor-man's logic-analyzer can't get much simpler.

depends on the efficiency of the display and the capability of your microcontroller. Typical values are 510 to 2,000  $\Omega$ . Mount them next to the LED (see **sidebar Photo 2**) so that they will not touch/short other components. This works especially well in single-step mode. (And, of course, this can certainly affect input pins. So be careful.)

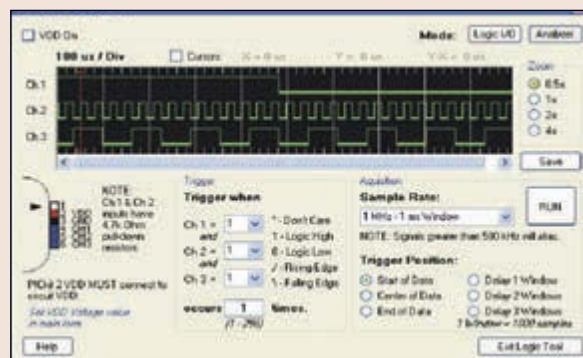
While the PICKit2 device (\$35) is an integrated part



**SIDEBAR PHOTO 2**

Here are two LED bar graph logic displays. The right one shows the SIP header connected to the LED bar graph socket and the positioning of the resistors better. (A socket is not an absolute requirement.)

of the free Microchip MPLAB Development System, the "PICKit2 Programmer" software is a separate free download application direct from Microchip that allows it to act as a simple three-pin logic analyzer. (It isn't included as part of Microchip's MPLAB.) Obviously, three pins aren't very much and it's not very sophisticated or fast. But for debugging microcontrollers, it's fast enough. It's perfect for serial data streams where you have Clock, Data, and Enable lines. And yes, it works for any TTL circuit or microcontroller. It's a completely stand-alone application. **Sidebar Photo 3** shows the output of a discrete TTL counting circuit.



**SIDEBAR PHOTO 3**

Separate software turns the PICKit2 programmer into a three-channel logic analyzer.

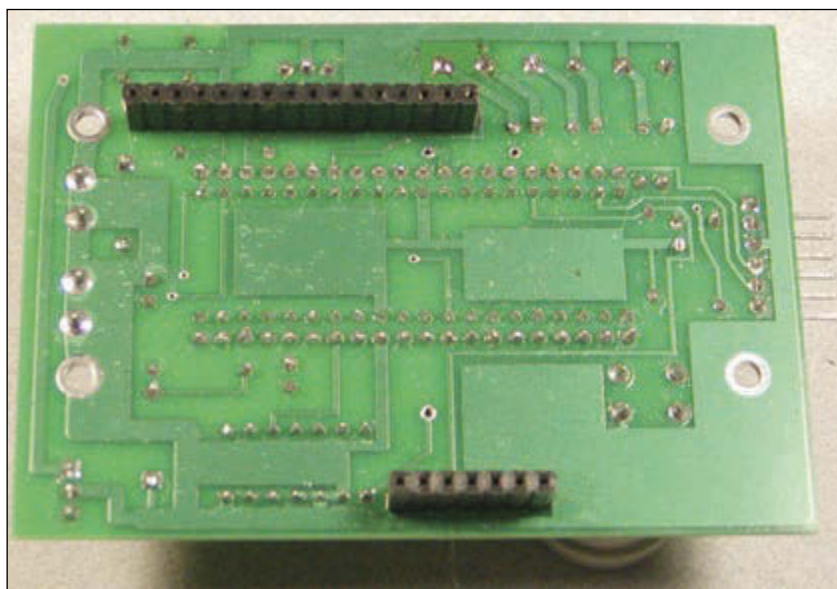


PHOTO 7

The bottom of the microprocessor board. Socket headers mate to J2 (I/O board) and J4 (LCD).

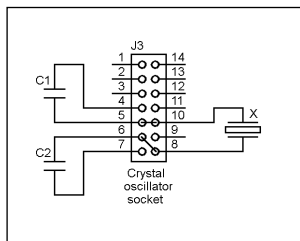


FIGURE 3

Crystal oscillator hook-up

I promptly drafted the CCP2 timer/PWM line of Port C to provide the analog voltage to the control the LCD contrast (LCD pin 3). Software sends out a PWM signal that is filtered into a DC value (see Figure 2). (The contrast is software controlled.) Once set up, this timer needs nothing more from the microprocessor and operates entirely on its own. I tested this with a variety of LCDs. Curiously, the yellow back-light version required a negative LCD bias voltage. Hence, a somewhat complicated circuit is needed to support all tested versions.

The value of back-light resistor (if needed) depends on the display. My yellow backlight

used 51  $\Omega$  while the white one needed 200  $\Omega$ . Verify what you need with the manufacturer's datasheet.

A 16-pin header strip is soldered to the back of the LCD circuit board which is mounted to the front panel with four screws and connects to J2. I had to open up the 2.5-mm mounting holes very slightly to pass 4-40 hardware (0.098" vs. 0.107"). In order to make the face of the display was flush with the cover, a 4-40 nylon nut/spacer is needed. I had to trim the sides of spacer nuts to fit. Additional nylon nuts were used to mount the circuit board because metal nuts would have touched board traces. **Photo 5** provides a view of this. Note that the display PCB and I/O PCB should be the same height from the front panel for proper connection to the microprocessor board (via J2 and J4).

## MICROPROCESSOR BOARD

The microprocessor board is not much more than a simple 5-V, DC power supply, the microprocessor, some connectors, and a few passive components. The power supply is a standard three-terminal type.

Header-strips are placed adjacent to the microprocessor. This is where the various future modules will connect. These pins are also very useful during testing and debugging. **Photo 6** shows the top physical layout.

**Photo 7** shows the bottom layout view. These two header-strip-sockets (J2 and J4) plug into the LCD and I/O board. Their placement is critical for proper connector mating. Drilling the mounting holes for the I/O board a little oversize helps in getting things aligned.

There is a six-pin, right-angle, header-strip is used to connect to the PICKit2 for programming and debugging (J1). A three-pin straight header-strip (J5) connects to the power. Friction from the connectors is usually sufficient to hold the microcontroller board in place. However, it can be mounted in place by increasing the length of the LCD mounting screws to 1" or longer. (Use spacers or extra nuts to set the proper height.)

A 14-pin socket is available for crystal or oscillator options. Normally, the internal microprocessor oscillator is used, but future projects may need the precision of a crystal. The socket is wired to accept four-, six-, eight-, and 14-pin DIP oscillators directly. Some oscillators have an "Enable" function that turns them on or off. This is brought out to a pad for possible future use. You can also use a bare crystal with capacitors. Just connect them as shown in **Figure 3** using a DIP-header. The capacitor values depend on the crystal. The RC oscillator option can also be implemented.

I also included a reset switch (SW1). Call



PHOTO 8

The bottom of the case holds AC components.



me old-fashioned but I like to be able to reset the microprocessor without having to remove power and wait for the capacitors to discharge. Being in control of the microprocessor is important to me.

Lastly, it is important to remember that future modules will plug into the header strips around the microprocessor. This means that all components (excluding the power supply) must be shorter than 0.400". This is only a concern for the PWM capacitors C5 and C9. Either use a short type or else lay them flat instead of mounting them straight up.

## AC POWER SUPPLY

The AC power supply is just a transformer switch and fuse (see **Photo 8**). Since future instruments will probably need op-amps and other non-five volt devices it's necessary to be flexible. A 24-V, center-tapped transformer can provide appropriate bipolar voltages. I used 0.25" quick connects, a power entry module, and a lighted DPST power switch. The transformer connects to the microprocessor board via a three-pin connector (yellow).


## FREE STUFF

All the driver software is available for free (for noncommercial and commercial

use) on the *Circuit Cellar* FTP site. There are lots of routines for driving the LCD, lighting the LEDs, and decoding the switches. It also includes all of the support routines (like timing and A/D code).

Additionally, there is a front panel diagram that provides the dimensions and placements for the holes on the front panel (non-commercial use only). The easiest way to construct the front panel is to print out the diagram as actual size. Then tape this to the front panel and drill/cut through this template. Double-sided tape works best.

## PLUG-IN BOARD

This is the first part for a number of future test instruments. The basic hardware and software is fixed, which makes product development very rapid. In the next installment, I'll describe a small and inexpensive plug-in board (2.5" × 3.8") and turn the base unit into a nice sweep generator. Of course, you can design your own plug-in boards for whatever you want. That's another nice feature. 

*Author's Note: A parts list is posted on the Circuit Cellar FTP site: [ftp://ftp.circuitcellar.com/pub/Circuit\\_Cellar/2014/](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2014/).*

**Verilog HDL**

**With the right tools**  
such as this book,  
**designing a microprocessor**  
**can be easy.**

Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one comprehensive guide to processor design in the real world.

Monte demonstrates how Verilog hardware description language (HDL) enables you to **depict, simulate, and synthesize an electronic design** so you can **reduce your workload and increase productivity.**

**Microprocessor Design Using Verilog HDL**

**cc-webshop.com**

**BEST SCOPES, BEST PRICES**

**PASSPORT-SIZE PC SCOPES \$162+**  
Great scopes for field use with laptops. Up to 200MHz bandwidth with 1GSa/s, high speed data streaming to 1MSa/s, built-in 1GSa/s AWG/wfm gen. **PS2200A series**

**30MHz SCOPE \$289**  
Remarkable 30MHz, 2-ch, 250MS/s sample rate scope. 8-in color TFT-LCD and AutoScale function. Includes FREE carry case and 3 year warranty! **SDS5032E**

**50MHz SCOPE \$399**  
50MHz, 4-ch scope at 2-ch price! Up to 1GSa/s rate and huge 12Mpts memory! Innovative "UltraVision" technology for real time wfm recording. FREE carry case! **DS1054Z**

**60MHz SCOPE \$349**  
Best selling 60MHz, 2-ch scope with 500MSa/s rate plus huge 10MSa memory! 8-in color TFT-LCD. Includes FREE carry case and 3 year warranty! **SDS6062V**

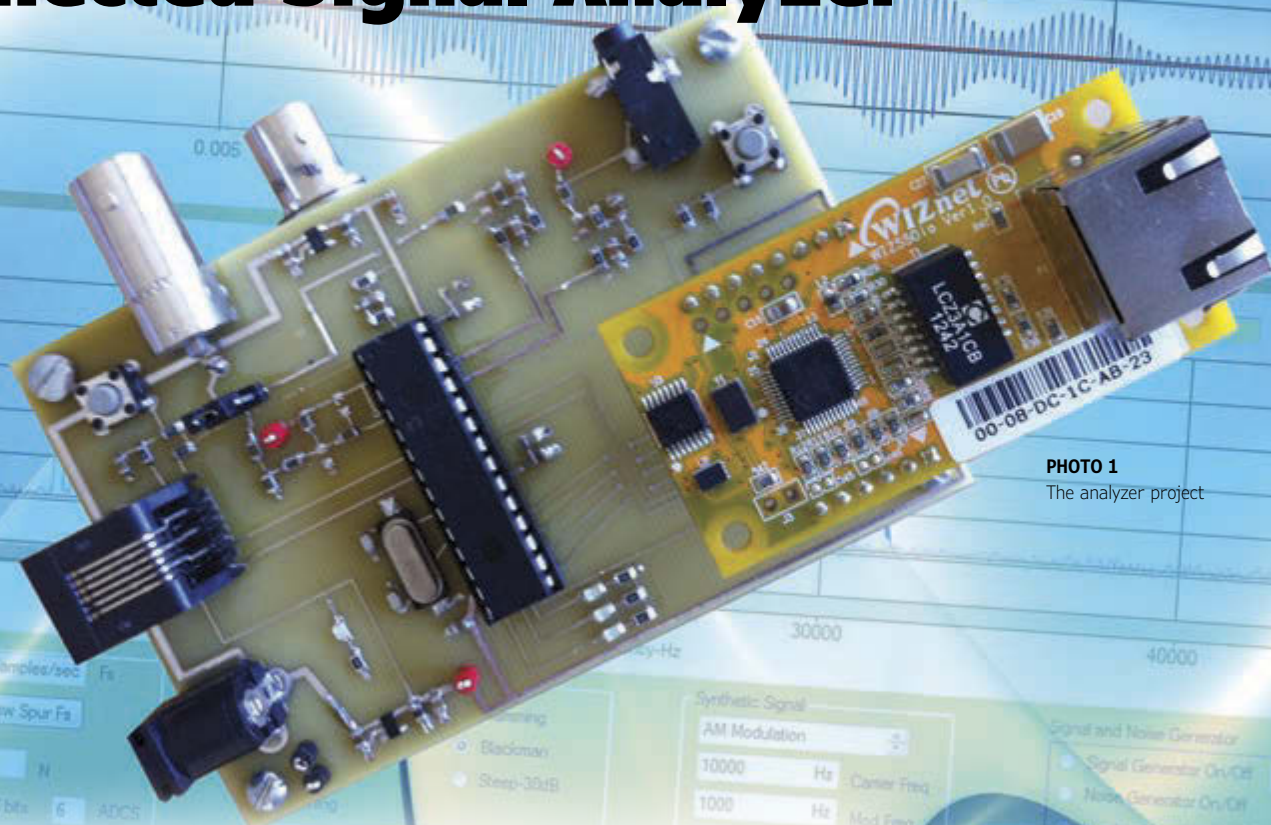
**70-300MHz SCOPES \$839+**  
Fast, versatile 2-ch 2GSa/s scopes with 8-in WVGA LCD, integrated generator, 14Mpt memory, very low noise floor. FREE carry case available! **DS2000A series**

**FREE TECH SUPPORT - GREAT CUSTOMER SERVICE** 



# Five-Function, Network-Connected Signal Analyzer

FEATURES



**PHOTO 1**

The analyzer project

This five-function, network-connected test tool would be an excellent addition to your workbench. The system includes a digital oscilloscope, spectrum analyzer, signal generator, noise generator, and filter response viewer.

*By Neal Martini (US)*

I wanted my students in a Communication Theory course I teach to get an appreciation for real-world signal sampling. So, initially, I set out to design a network-connected ADC system accessible to my students over a LAN. I intended to enable them to modify various sampling parameters and then observe the impact on data integrity. But, as the project progressed, my design evolved and I ended up with a much more capable system. The resulting five-function, network-connected test tool was my entry (project number WZ1281) to the WIZnet Connect the Magic 2014 Design Challenge (see **Photo 1**). The WIZnet WIZ550io Ethernet module-connect design functions include a digital oscilloscope, spectrum analyzer, signal generator, noise generator, and a filter response viewer (see **Figure 1**). Using the system's PCB running Microchip Technology dsPIC33 firmware and a PC application running the user interface

(UI)—both of which communicate over a network—you have a very powerful signal analysis tool (see **Photo 2**).

## SYSTEM CAPABILITIES

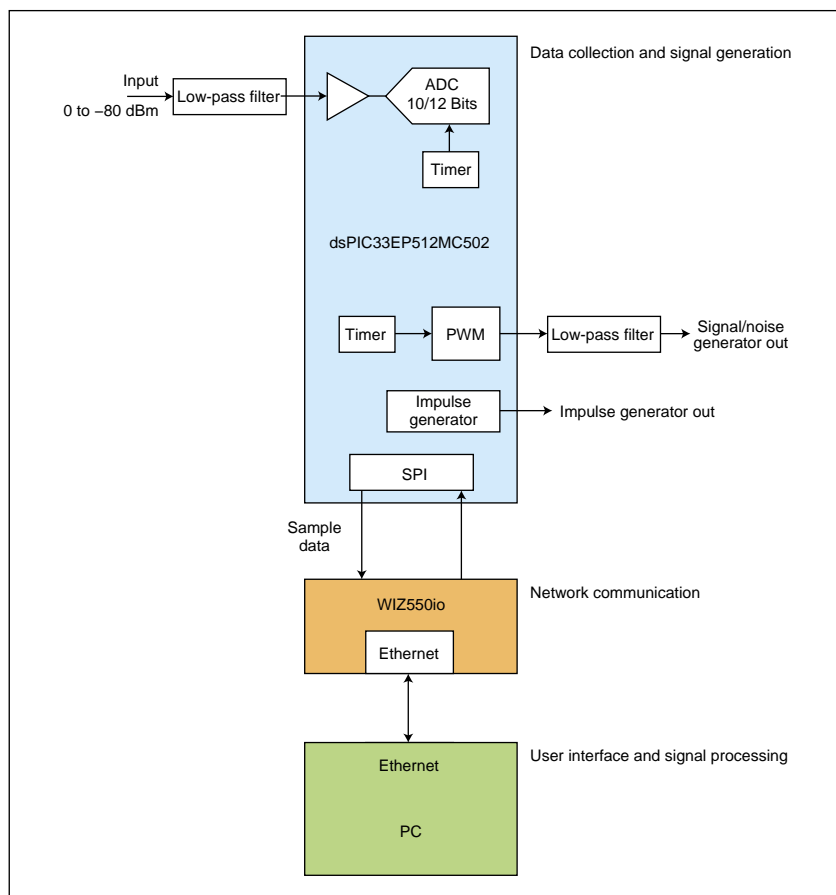
The test tool is capable of displaying a time plot of the input signal (digital oscilloscope) and the Fourier spectrum (spectrum analyzer) of that signal. Additionally the system has the ability to generate test signals (signal generator) or noise signals (noise generator) that can be fed into the system's input for analysis or used in an external user application. And finally, the system gives you the ability to deliver an impulse into an unknown filter and subsequently transform the impulse response to view the filter's transfer function (filter response viewer).

Figure 1 shows where the system's various functions are executed. The antialiasing filtering, signal sampling, signal generation,

noise generation, and impulse generation (used in filter response mode) are handled by the dsPIC33-based hardware PCB. The design is networked to a PC using a WIZ550io module. The PC application provides the main UI. In the UI, all critical parameters are passed to the PCB over the network. The PC also does the heavy lifting when it comes to spectrum analysis. Very high-speed fast Fourier transforms (FFT) are executed here. The UI also provides you with very sophisticated display capabilities. Additionally, the PC application enables you to generate synthetic signals and find the signals' spectrum. This standalone capability does not require the PCB to be connected. It provides an easy way for you to exercise the UI before the PCB is constructed.

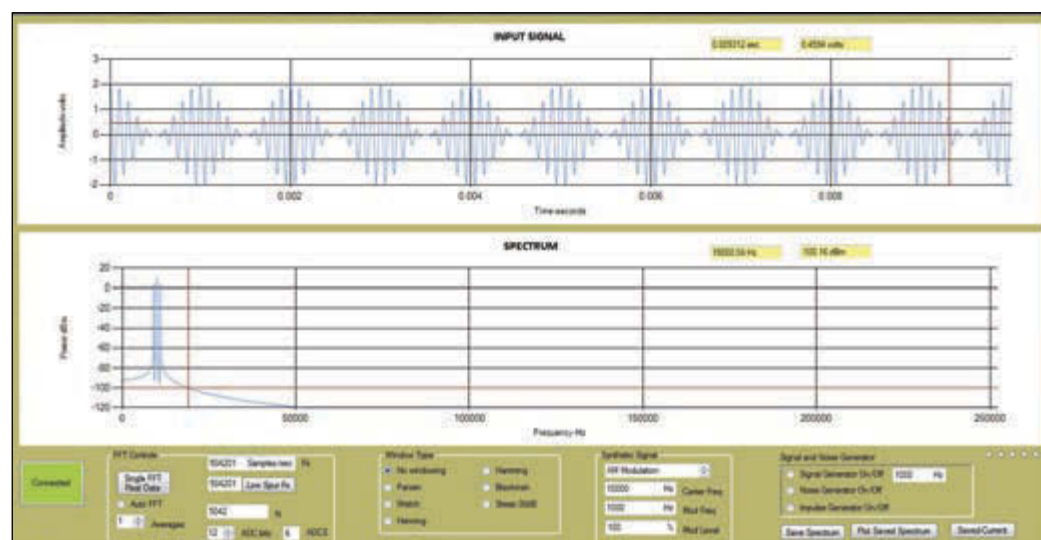
## HARDWARE

**Figure 2** depicts the main IC hardware components: a Microchip Technology dsPIC33EP512MC502, a WIZ550io module, and Analog Devices ADP151 low-dropout, low noise voltage regulators. I chose the dsPIC33EP512MC502 for a variety of reasons. First of all, I wanted to use a DIP package so that my students could build a PCB of their own if they chose to without having to solder a fine-pitch SMT device. Because high accuracy is desired for quality ADC sampling, I wanted an external 8-MHz crystal. Utilizing the dsPIC33's internal PLL, the microprocessor runs at 60 MIPS. This ensures we can keep up with the high-speed sampling and data transfers required to deliver excellent performance. The dsPIC33 also has about 50 KB of RAM, which allows large data blocks to be saved without the need for external memory. Also, since any sampling system needs an analog interface, the fact that this dsPIC33 has a built-in amplifier is ideal. Additionally, the dsPIC33



**FIGURE 1**

An overview of the five-function test tool

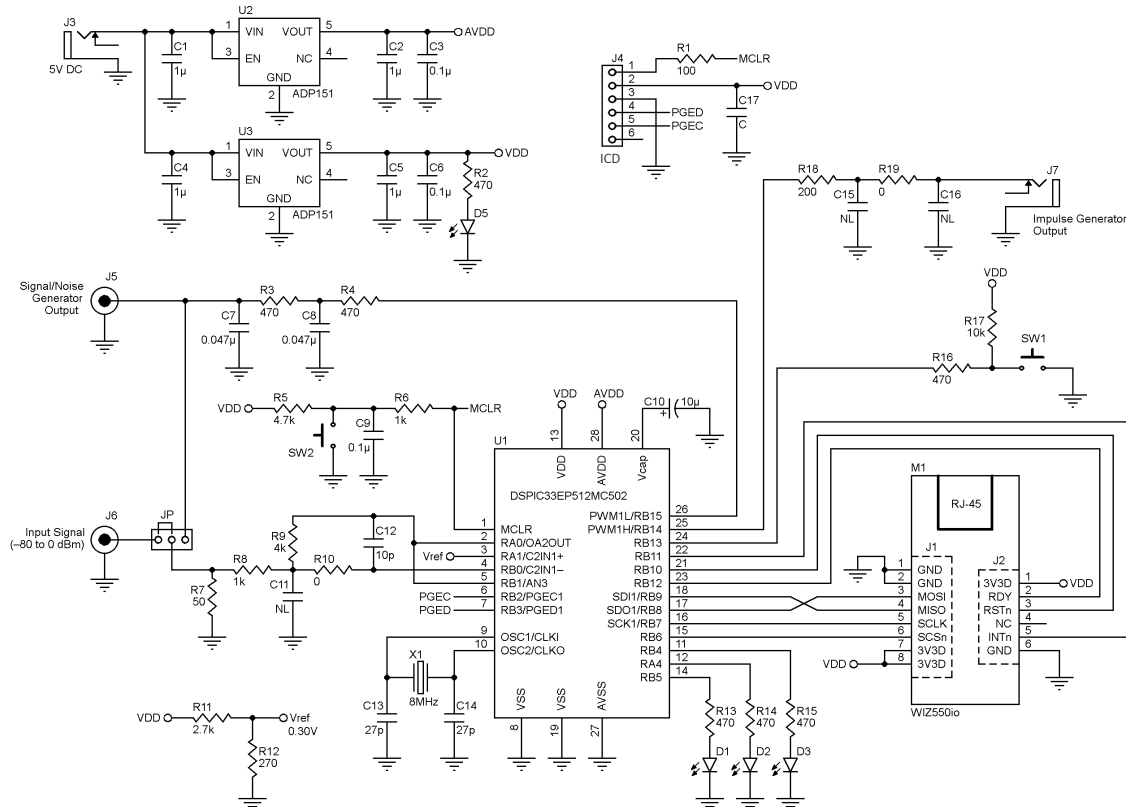


**PHOTO 2**

PC application user interface

**FIGURE 2**

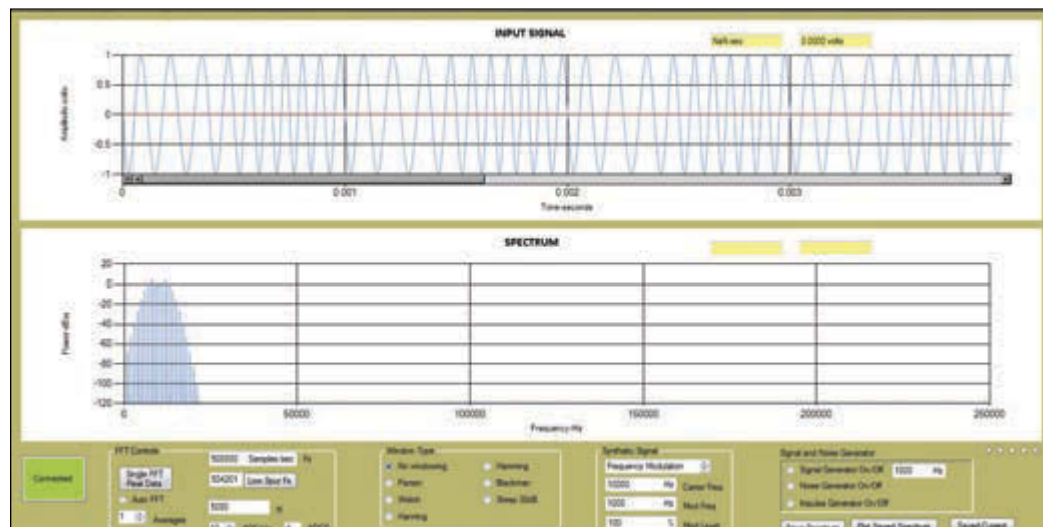
The PCB schematic



contains several high-quality peripherals needed for this application. The peripherals and their functions include: Timers 1 and 3 (precise control of sampling and DDS signal generation); high-speed PWM (DDS signal generation and noise generation); a SPI module (communication with the WIZ550io); ADC (high-speed, 10 bit/12 bit analog-to-digital conversion); and an operational amplifier (front end antialiasing filter and buffering).

The WIZ550io module manages the communication to and from the PC over the network. The control and data transfers are easily handled utilizing the dsPIC33's SPI communication port and a few control lines (RDY and RESET). Several passive components on the PCB are mainly used to accomplish antialiasing filtering and filtering for the PWM output. (I'll describe those shortly.)

You'll note that two separate 3.3-VDC regulators are included. This isolates the

**PHOTO 3**

The UI showing a synthetic FM signal being specified



analog ADC supply and minimizes digital noise crossover. I selected the ADP151 because it has a very low output noise ( $9 \mu\text{V}_{\text{RMS}}$ ) and a very low dropout voltage (140 mV). The low noise ensures a pure ADC supply. The ADP151 also requires very simple external filtering to get its job done.

Various connectors are included to provide programming capability (ICD), signal input, generator output, and impulse output. The WIZ550io module contains the Ethernet connector.

## SOFTWARE/FIRMWARE

The microprocessor firmware and the PC software are contained in two project files. The dsPIC33 firmware is written in C code using MPLABS X IDE (free from Microchip Technology). The project file is called WIZ1281\_Micrchip\_MPLX\_dsPIC33\_Project. The PC software is written in C# using Visual Studio 2010 Express IDE (free from Microsoft). The project file is called WIZ1281\_VisualStudio2010\_C#\_Project. In the following section, I break down where the various WIZ1281 functions are executed.

In the dsPIC33EP5012MC502 firmware, you have: timer-controlled analog-to-digital conversion; timer-controlled PWM signal generation; timer-controlled noise generation; impulse generation; and communication with WIZ550io module for Ethernet connection to the PC.

In the PC software, you have: UI for selecting all control variables (Fs, N, Averaging

## ABOUT THE AUTHOR

Neal Martini holds an MSEE from the University of Missouri, Rolla. He is retired after 24 years of working for Hewlett-Packard in the LaserJet and InkJet printing businesses. In addition to being involved with a variety of boards, Neal works independently in product development in several application areas.

n, ADC bit selection, FFT window type, etc.); plotting of input data, power spectrum and filter impulse response, and transfer function; synthetic signal generation to exercise the UI; and Ethernet communication with the PCB.

If you refer to the code posted on the Circuit Cellar FTP site, you'll see that I included a folder called WZ1281\_UI\_Application\_Installer. When the setup.exe in this folder is executed, the C# application WindowsFormsApplication1.exe is automatically installed on your PC. To run the application, you must have Microsoft .NET 4.0 (or later) installed on your PC. You can download it separately from Microsoft ([http://msdn.microsoft.com/en-us/library/5a4x27ek\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/5a4x27ek(v=vs.100).aspx)), or you can download Microsoft's Visual Studio 2010 Express ([www.visualstudio.com/en-us/downloads/download-visual-studio-vs#DownloadFamilies\\_2](http://www.visualstudio.com/en-us/downloads/download-visual-studio-vs#DownloadFamilies_2)) and the .NET framework will be automatically included.

You can operate the UI application without the PCB attached. I added a synthetic signal generation capability to the UI so you can specify parameters for various sample

```
// Declare W5500 driver SPI Functions
reg_wizchip_cs_cbfunc(wizchip_select, wizchip_deselect);
reg_wizchip_spi_cbfunc(wizchip_read, wizchip_write);

//Functions
void wizchip_select(void) {
    WIZCS = 0;}
void wizchip_deselect(void) {
    WIZCS = 1;}
void wizchip_write(uint8_t wb)
{
    uint8_t dummy;
    SPI1BUF = wb;    // write to buffer for TX
    while( !SPI1STATbits.SPIRBF );    // wait for TX complete
    dummy = SPI1BUF;
}
uint8_t wizchip_read()
{
    SPI1BUF = 0x00;    // write to buffer for TX
    while( !SPI1STATbits.SPIRBF );    // wait for TX complete
    return SPI1BUF;    // read the received values
}
```

### LISTING 1

Four functions control how the dsPIC33's SPI peripheral talks to the WIZ550io

```

var result = client.BeginConnect(IPAddress.Parse("192.168.0.123"), 4000, null, null);
result.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1)); //timeout if no PCB
if(client.Connected)
{
    clientStream = client.GetStream(); //get a client stream
}

```

**LISTING 2**

The TcpClient Class is used to establish the network link between the PC and the WIZ550io.

signals (AM, FM, and Square Wave) and to then generate the signal's spectrum. This enables you to interact with the UI and get a good idea of how it operates before a PCB is fabricated. **Photo 3** is a screenshot of the UI showing a synthetic FM signal being specified. The upper plot is the FM time domain signal and the lower plot is its spectrum.

**ETHERNET COMMUNICATION**

Let's briefly review how Ethernet communication is accomplished. On the microprocessor end of the communication, the ioLibrary\_BSD drivers are used to handle the WIZ550io communication. (The drivers are available at <http://wizwiki.net/wiki/doku.php?id=products:w5500:driver>.) In order to use these drivers, four functions had to be written to control how the dsPIC33's SPI peripheral talks to the WIZ550io. **Listing 1** shows the functions, including the declaration of these functions to the driver. Once this is done, communication with the WIZ550io and the PC is readily accomplished using the `recv(sn,buf,size)` and `send(sn,buf,size)` functions included in the ioLibrary\_BSD drivers.

A default IP address is assigned to the WIZ550io. But since I was attaching the WZ1281 to a classroom LAN, I needed to change the IP address. The classroom LAN had a TP-LINK router that has a default IP of 192.168.0.1. In order to connect the system

to the LAN, I needed to have the first three fields of the WIZ550io IP address match that of the router. Then the fourth field was arbitrarily assigned. I consequently assigned an IP of 192.168.0.123 to the WIZ550io. There are several alternatives I could've used here. I could've directly connected to a PC and then gone into the Windows Network Sharing options and assigned an IP. Alternately, I could've modified the firmware and software to dynamically assign an IP using the DHCP protocol in the router. I chose the simplest approach for this application. Incidentally, I also arbitrarily selected Port 4000 for this application.

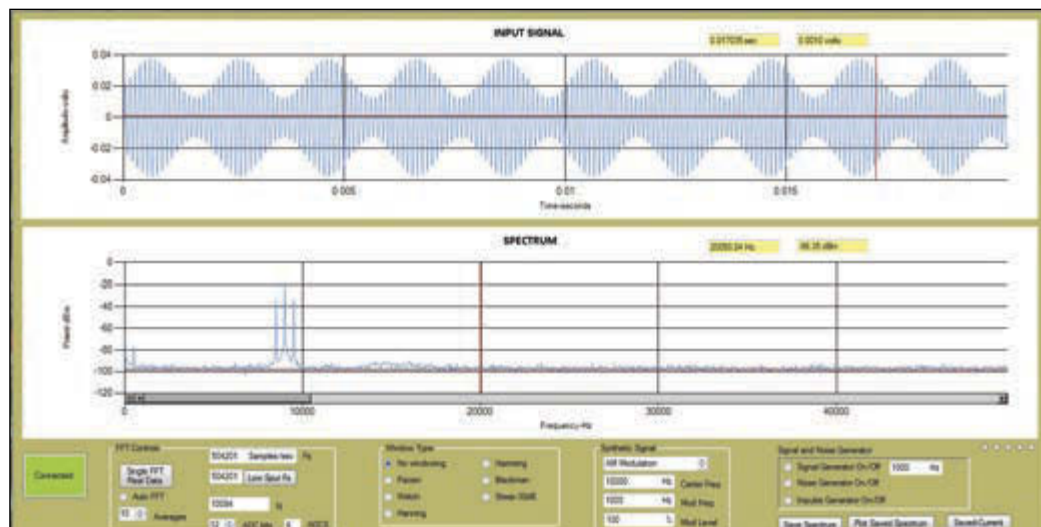
On the PC end of the Ethernet communication, the C# code required to establish a connection and pass commands and data back and forth is very straightforward. The TcpClient Class available in Windows handles the communication. **Listing 2** is the code snippet to make the connection.

Once the connection is established, the TcpClient functions `clientStream.Write(TxBuff, offset, size)` and `clientStream.Read(RxBuff, offset, size)` are all that is needed to pass commands and data between the PC and the design at very high rates. This turned out to be much easier than I had expected. Using the WIZ550io and some fairly simple code, I got a high-speed, network-connected device.

As I mentioned earlier, the analyzer has

**PHOTO 4**

An example of the oscilloscope mode output



# SUPERIOR **EMBEDDED** SOLUTIONS



**DESIGN YOUR SOLUTION TODAY**  
**CALL 480-837-5200**

[www.embeddedARM.com](http://www.embeddedARM.com)

**NEW! TS-7250-V2 Embedded Board**  
High Performance and Industrial Grade



Pricing Starts At

**\$165**

Qty 100

**\$199**

Qty 1



Available with TS-ENC720 enclosure  
(Shown with optional microSD card)

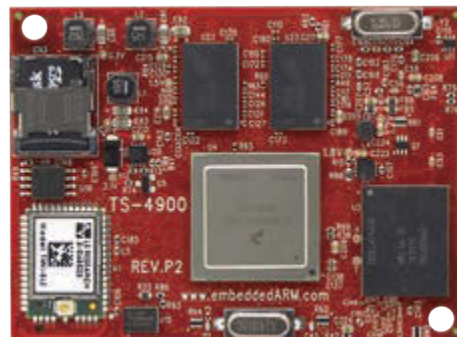
**Features:**

- Up to 1 GHz ARM CPU
- 512 MB RAM
- 8 or 17 KLut FPGA
- 2x SD Card Socket
- 2x 10/100 Ethernet
- 2x USB Host
- 1x USB Device
- 6x Serial Ports
- 75x DIO, 1x CAN
- 1x PC/104 Connector

**Benefits:**

- Hardware flexibility with on-board FPGA
- Several control I/O interfaces
- Launches your application in half a second
- Easy development w/ Debian and Linux 3.14
- High data reliability with 2 GB SLC eMMC flash
- -40 °C to 85 °C industrial temperature range

**NEW! TS-4900 Computer Module**  
1 GHz i.MX6 with WiFi & Bluetooth



Pricing Starts At

**\$99**

Qty 100

**\$134**

Qty 1

- 1 GHz Single or Quad Core Cortex A9 ARM CPU
- Up to 2 GB DDR3 RAM and 4 GB eMMC Flash
- Wireless 802.11 b/g/n and Bluetooth 4.0
- 4 KLut FPGA, 1x Gigabit Ethernet, 1x PCI Express Bus
- 1x mSD slot, 1x SATA II, 1x USB Host, 1x USB OTG
- 70x DIO, 4x I2C, 1x I2S, 2x SPI, 2x CAN
- -40 °C to 85 °C Industrial Temperature Range
- Runs Linux 3.10, Debian, Ubuntu, Yocto, QT, OpenGL
- **Coming Soon:** QNX, Android and Windows Support



**COMING  
SOON!**

**TS-7970:** SBC Version  
of the TS-4900



Available w/ TS-8150  
PC/104 Development Kit



We've never  
discontinued a  
product in 30  
years



Embedded  
systems that  
are built to  
endure



Support every  
step of the way  
with open  
source vision



Unique embedded  
solutions add  
value for our  
customers

[www.embeddedARM.com](http://www.embeddedARM.com)



five basic operating modes. Now let's cover each one.

## DIGITAL SCOPE

The digital oscilloscope (and sampling in general) operating mode is essentially a low-frequency sampling digital oscilloscope. Although it's limited to a sampling rate of 1.4 MHz (10 bit), it is very useful in many applications. **Photo 4** shows an example of the oscilloscope mode output. The upper plot is the input data plotted versus time. In this case, the test signal connected to the system's input is an AM signal generated by my HP signal generator.

In the UI, you can select the precise sampling frequency ( $F_s$ , 1-Hz resolution), the number of samples  $N$  (20,000 maximum), and 10- or 12-bit ADC. The plot update rate is determined by the size of  $N$ . For example, if  $N$  is 5,000, the display is updated approximately eight times per second on my laptop PC. The UI plot has cursor capability (shown in red) for selection/display of time, frequency, and magnitude information. There is also a very powerful zoom capability available. Visual Studio 2010 Express makes adding these features a snap.

The Digital Oscilloscope's sampling front end consists of an antialiasing filter and an amplifier. The operational amplifier and the ADC are internal to the dsPIC33EP512MC502. The passive components are external. The filter architecture is multi-feedback (MFB). I used Microchip FilterLab 2.0 to calculate component values. The design is a two-pole, low-pass filter with a cutoff frequency of 500,000 Hz. The PCB layout reflects this design.

Very early on, however, I realized that the operational amplifier located in the dsPIC33 has a fairly large gain bandwidth product (GBWP) of 6 MHz. Consequently, signals

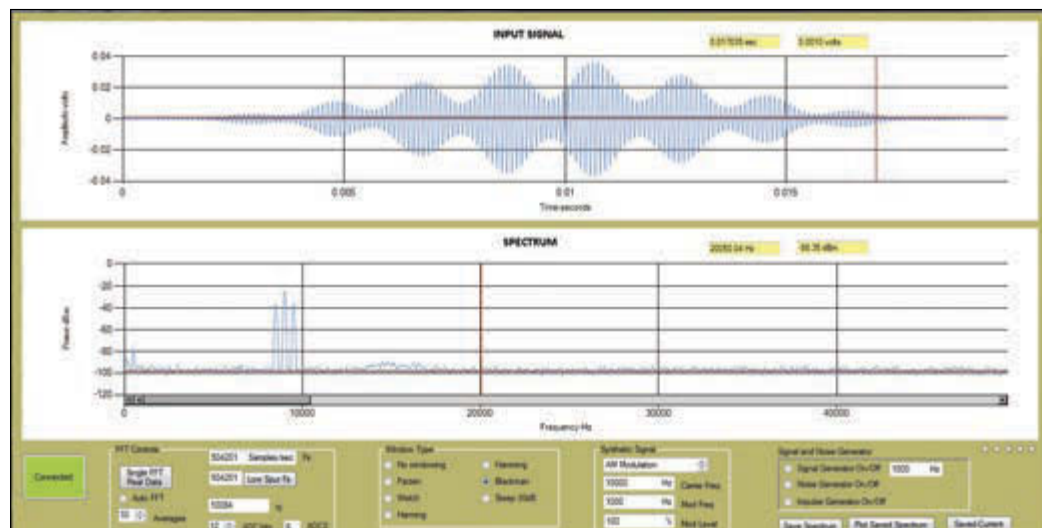
above the ADC's allowable sampling rates easily pass through the amplifier, although the amplitudes are attenuated somewhat. So, if you move the cutoff frequency of the antialiasing filter far up in frequency, you can see signals at much higher frequencies than the frequency of the ADC sampling. This type of sampling is called sub-sampling and allows you to use aliasing to your advantage. Consequently, I am currently using very soft filtering to allow this wide bandwidth capability.

$R_7$  is set to 50  $\Omega$  to provide a standard test instrument input impedance. The signal gain of the amplifier is determined by  $R_9$  and  $R_8$  ( $R_9/R_8 = 4 \text{ k}\Omega/1 \text{ k}\Omega = \text{gain of } 4$ ). As I stated above, to soften the low-pass filter, I eliminated the second pole.  $R_{10}$  is therefore 0  $\Omega$  and  $C_{12}$  is not loaded.  $C_{11}$  is a 10-pF capacitor providing of single pole near 4 MHz.

Since the ADC input needs to be between 0 and 3.3 V, a 0.3-V reference is supplied to the plus input to the operational amplifier. The DC bias voltage that this generates at the ADC input is:  $\text{ADCBias} = (1 + R_9/R_8) V_{\text{REF}} = 1.5 \text{ V}$ . This is nicely in the middle of the operating range of the ADC, allowing bipolar signal input capability.

The range of input signal that can be applied is determined on the high end by the saturation point of the operational amplifier and, on the low end, by the number of bits in the ADC. Using the 12-bit ADC, there is a very linear response from  $0.225 V_{\text{RMS}}$  to  $22.5 \mu\text{V}_{\text{RMS}}$ . For those of you familiar with the decibel (dB) terminology, this is a dynamic range of 80 dB, which is very respectable considering the simplicity of the hardware.

Timer3 in the dsPIC33 is used to trigger the ADC. The trigger period is determined by the sampling frequency  $F_s$  requested by the user. The timer period set by PR3 is:



**PHOTO 5**

Analyzing an AM signal utilizing windowing to minimize leakage in the Power Spectrum

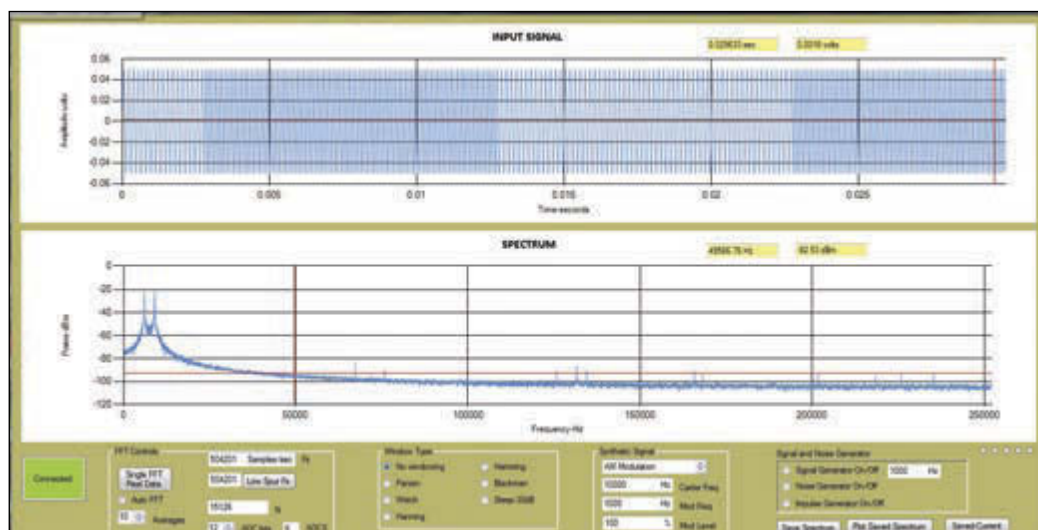


PHOTO 6

Display of a FSK encoded binary bit stream and its Spectrum

$$PR3 = \left( \frac{F_{OSC}}{2 F_s} \right) - 1 = \left( \frac{60 \text{ MHz}}{F_s} \right) - 1$$

$F_{OSC}$  of 120 MHz is selected to provide excellent resolution on the sampling period. Also, an external crystal is included to keep the ADC sampling jitter to a minimum. Incidentally, you may have noticed a box on the UI labeled “Low Spur  $F_s$ .” Although you can select any  $F_s$ , this box calculates the sampling rate closest to the desired  $F_s$  that minimizes jitter. Jitter results when the sampling period is not an exact integer multiple of the ADC conversion cycle time. You can decide to use the low-spur  $F_s$  or not.

One final parameter to mention is the box labeled “ADCS” on the UI. ADCS is a constant used by the dsPIC33 to generate the ADC conversion clock period (TAD). TAD is calculated as follows:  $TAD = T_{cy}(ADCS + 1)$ , where  $T_{cy} = 1/60 \text{ MHz}$ .

Microchip Technology recommends minimum TADs of 75 and 117 ns for 10- and 12-bit ADC, respectively. This is an ADCS of 4 and 6, respectively. I found that you can go down to ADCS of 2 and still get excellent signal detection capability. This allows much higher sampling rates. The only thing that is degraded is that the magnitude accuracy degrades somewhat. It’s fun to play with this parameter and observe the impact on performance.

## SPECTRUM ANALYZER

A spectrum analyzer is a powerful tool used to analyze signals. It provides information about a signals makeup that cannot be seen by looking at the amplitude versus time version of a signal. Because of the inherent gain in the process of calculating the power spectrum, signals buried in noise can be detected with

relative ease. The power spectrum produced by this design shows the breakdown of a signal into its sinusoidal components. The plotted spectrum is a power versus frequency representation that shows what frequency sinusoids are present and at what relative power level. There is an enormous amount of material available on spectrum analysis. Including a full explanation here would be too extensive. For the following discussion, I will assume the reader has a basic knowledge of what a spectrum and its uses. Let me state a few things that will help explain the capability of the system’s spectrum analyzer mode.

The FFT is a very fast and efficient algorithm widely employed that is used to calculate the power levels of  $N/2$  frequency bins. The plot of these  $N/2$  frequency bins versus power levels is the signal’s power spectrum.

The resolution of the resulting power spectrum is  $F_s/N$ . For example, if  $N$  is 5,000 and  $F_s$  is 500,000 Hz, there will be  $N/2$  (2,500) frequency bins in the output spectrum, spaced at  $F_s/N$  (100 Hz).

The larger the  $N$  value, the better the frequency resolution. Also the inherent gain of the FFT process increases as  $N$  increases. A rule of thumb here is that you improve the signal to noise ratio (SNR) by about 3 dB if you double  $N$ . The larger  $N$  also slows stuff down, however.

The sampling frequency  $F_s$ , number of

N	Spectrum Update Rate (Hz)
5000	8.33
10000	3.96
15000	2.48
20000	1.70

TABLE 1  
Speed versus N

```
// Receive data over the network
int read = 0, offset = 0, toRead = 2 * N;
while (toRead > 0 && (read = clientStream.Read(RxBuff, offset, toRead)) > 0)
{
    toRead -= read;
    offset += read;
}

DCterm = myFunctions.DCtermCalc(N, RxBuff, ScaleFactor);    //calculates DC term

//subtracts DC, scales for ADC bits, puts data in real part of din[]
myFunctions.Filldin(N, DCterm, RxBuff, ScaleFactor, AmpGain, din);

myFunctions.windowing(N, din, WindowType, 1, din);           //window data
plotTime.PerformClick();    //plot input time series
fftwf.execute(fplan);       //Do FFT

//convert to dBm and average spectra (code not shown here)

plotFreq.PerformClick(); //plot frequency domain data (spectrum)
```

**LISTING 3**

PC code to get samples over the network and scale, window, FFT, and plot time domain and frequency domain data

samples  $N$ , number of averages ( $n$ ), and window type can all be selected in the UI. Averaging here refers to straight arithmetic summing of the power magnitude levels and dividing by  $n$ . This type of averaging is called incoherent averaging and the resulting improvement to the SNR is approximately:  $10\log_{10}(\sqrt{n})$ .

When a signal has sinusoidal components that do not fall into the exact center of one of the calculated frequency bins, a phenomenon called “leakage” occurs. When this happens, the frequency bins overlap and power spectrum distortion occurs. Windowing is a common technique used to minimize leakage. Simply stated, windowing is the process of taking the input signal and multiplying it by a shaped curve before the FFT is performed. This multiplication effectively softens the amplitudes of the time samples at the beginning and end of the block of samples. There are 6 different popular window types available in the UI.

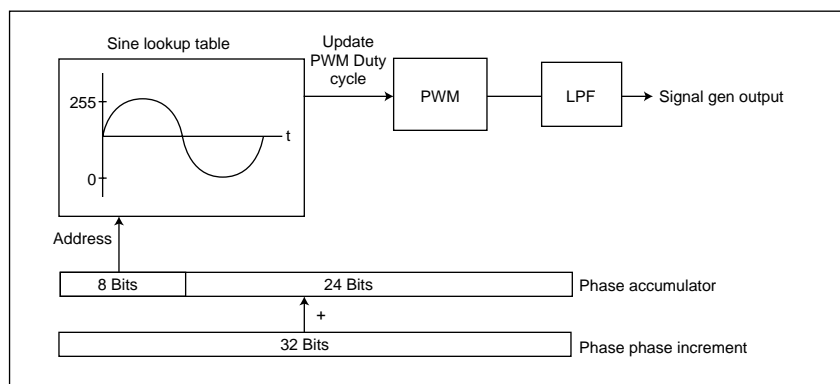
Now let’s look at a few power spectra resulting from real data input into the system. The first example shows the spectrum generated when is the same AM signal shown in the digital oscilloscope discussion above is reused. However, in this test, windowing is applied (see **Photo 5**). First of all, note that the top plot now shows the AM signal after it is multiplied by the window function. Note how the amplitudes fall off as you approach the ends of the block of data. The bottom plot is the power spectrum. I am using the zoom capability here to show the spectrum detail. This is a classical sinusoidally modulated AM signal’s power spectrum. From the spectrum you can see that the sinusoidal tone (carrier) being modulated is at about 9 kHz (center tone). The spacing of the two side signals from the carrier tells you that the modulating frequency is 500 Hz. Also the level of the side signals indicates about a 50% level of modulation. That kind of information would be difficult to extract if you just look at the time based data.

**Photo 6** is another example. This is the spectrum of frequency shift keyed (FSK) signal used to encode a binary bit stream. You can see in the top plot the time domain data where the frequency switching is visible. Looking at the two spectrum peaks, you can see that the two FSK frequencies are 6,000 and 9,000 Hz. The number of power spectra being averaged in this example is chosen to be 10 in the UI.

The FFT and windowing are all done in the PC. These could be done in the microprocessor, but you would not get even close to the rates that the FFTs are done in the PC. The FFT

**FIGURE 3**

The direct digital synthesis signal (DDS) generator





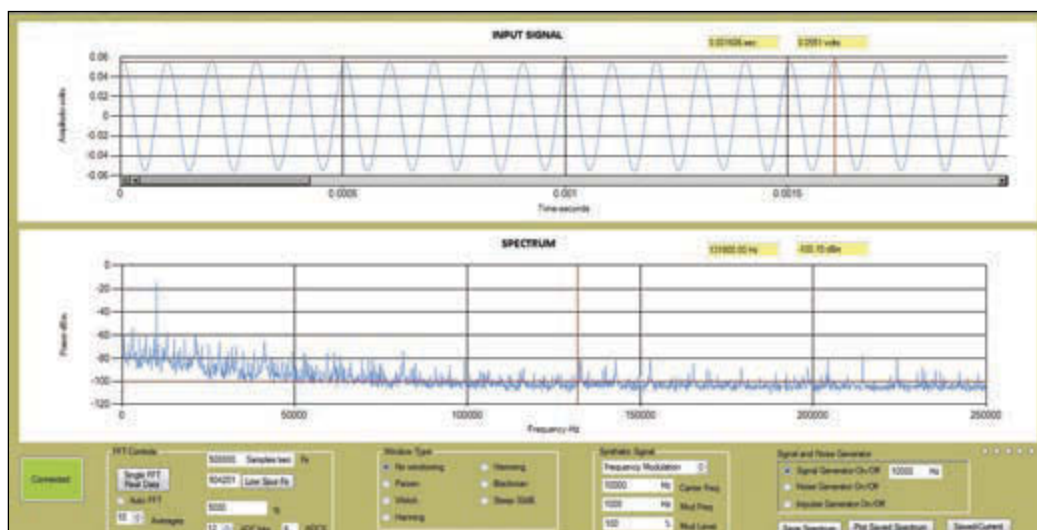


PHOTO 7

Time and frequency output when the DDS Signal Generator is producing a 10-kHz sine wave and is connected to the analyzers input.

library routines I am using were originally developed in 1999 ([www.fftw.org](http://www.fftw.org)). I was fortunate to find another source online that modified the original C-based code into a C# wrapped implementation (<https://github.com/tszalay/FFTWSharp>). I extracted the pieces I needed into my C# application and interfaced the code to my UI.

The overall speed of the spectrum analyzer mode is very good. The speed is of course dictated by the sample block size  $N$  selected. As I stated earlier, you want large  $N$  for higher spectrum resolution and small  $N$  for speed. The following data in **Table 1** will give you an idea of the speed versus  $N$  trade-off. The update rate includes all steps in the process: sampling, data transfer, windowing, and performing the FFT and plotting.

Finally, **Listing 3** is a code snippet that is the heart of the PC based C# application. This is where all the signal processing takes place.

## DDS SIGNAL GENERATOR

**Figure 3** shows the direct digital synthesis signal (DDS) generator. I decided to add a simple signal generator to the system so that the students could exercise the spectrum analyzer without needing an external commercial signal generator. The generated signal can obviously be used to feed some external circuit if you need it.

The DDS technique for generating sinusoids has been around for a while, so I am not claiming to be doing anything new here. Looking at the block diagram, you see a PWM attached to a low-pass filter (LPF). This combination acts essentially like a DAC. If the duty cycle of the PWM was constant, for example, the LPF output would be a DC value. If we vary the duty cycle to match the varying amplitude of a sine wave we want to generate, the LPF output will be that sine wave.

The dsPIC33 PWM is running at 468,750 Hz (i.e.,  $F_{\text{SYS}}/256 = 120 \text{ MHz}/256$ ). This is

LISTING 4

DDS Signal Generator code to create a look-up table, calculate the phase increment and update the PWM duty cycle

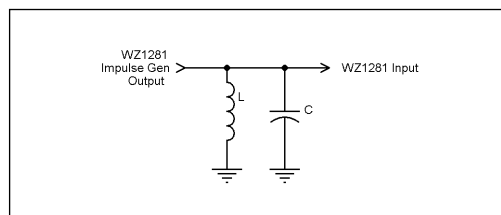
```
// generation of signal generator lookup table; one cycle
for(i = 0; i<256; i++)
    SIN8[i] = (uint8_t) (128 + 127 * sin((PI * i)/128) + .5); // .5
is for rounding

//calculate the phase increment
DDSD = ((FrequencyDesired * pow(2,32))/468750.0) + .5;

//Timer1 interrupt service routine
void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
{
    PDC1 = SIN8[(DDSD>>24)]; //do table lookup using upper 8 bits of DDSD
    DDSD += DDSD; //increment phase accumulator
    IFS0bits.T1IF = 0; //Clear Timer1 interrupt flag
}
```

FIGURE 4

Test setup to demonstrate the Filter Response Viewer. With  $L = 18 \mu\text{H}$  and  $C = 1 \mu\text{F}$ , the band-pass filter's resonant frequency is 37,523 Hz.



also the DDS update rate. Once per PWM cycle, I calculate the amplitude of the desired sine wave and then set the PWM duty cycle to generate that amplitude. To select the desired amplitude/PWM duty cycle, I used a one cycle sine wave table. I simply select the amplitude from the table that corresponds to where the desired sine wave would be in a sine cycle at the DDS update point in time and update the duty cycle accordingly. A 32-bit phase accumulator (called DDSp in the code) is used to keep track of the phase of the desired sine wave. Once per DDS update cycle, the phase accumulator is incremented by what the phase increase (called DDSd in the code) is for the desired sine wave. The phase increment is calculated as follows:  $\text{Phase increment} = (2^{32}) \text{FrequencyDesired} / \text{DDS update rate}$ . For example, if you want a 10,000-Hz sine wave, the phase increment would be 91,625,968. This value is added to the contents of the phase accumulator at the DDS update rate. Therefore the 32-bit phase accumulator accurately tracks the phase of the desired sine wave at any point in time. Since having a look-up table with  $2^{32}$  entries in RAM is prohibitive, we take the upper 8 MSB and use them as the address into the single cycle lookup table.

Incidentally, one needs to be careful when selecting the cutoff frequency of the LPF located at the PWM output. You want the cutoff to be low enough to adequately smooth the PWM pulses (ripple), but high enough to pass the signal you are trying to generate. Fortunately, there is an online resource (<http://sim.okawa-denshi.jp/en/PWMtool.php>) available that makes the design process easy.

**Photo 7** shows a UI output that results when a signal generator frequency of 10,000 Hz is selected. In this test, the signal generator's output is connected to the input of the WZ1281. The upper plot shows the signal generator's output sine wave. It looks pretty good there. The spectrum shows the signal quality in more detail. You can see the 10,000-Hz signal, but you also see that there are other spurious components present. These distortions are down about 40 dB from the desired signal, however, which is pretty good for a simple generator. I did this fairly quickly and intend to revisit the design and make some improvements to reduce the distortion components even more. A few improvements I intend to try are a larger look-up table located in ROM and faster DDS update rates. The pseudocode in **Listing 4** shows the essential pieces of the DDS generator.

## FILTER RESPONSE VIEWER

To understand what this operating mode is all about, take a look at the transfer function of a typical first order low pass filter in Max Kamenetsky's demonstration, "Filtered Audio Demo" ([http://web.stanford.edu/~boyd/ee102/conv\\_demo.pdf](http://web.stanford.edu/~boyd/ee102/conv_demo.pdf)). This particular filter has a cutoff frequency of 1,000 Hz. It shows what frequencies the filter passes and how frequencies are attenuated. As it turns out, this plot is actually the FFT of the filter's impulse response. The beauty of this is that if you know an arbitrary filter's impulse response, you can find its transfer function by taking the FFT of the impulse response. The next obvious question is: How do you get a filter's impulse response?

An impulse is simply a high amplitude pulse that exists over a very short period of time. In my project, I approximate an impulse by outputting a single 3.3-V pulse for one sample period of  $N$  samples and then output 0 V for the other  $(N-1)$  samples. If you feed such a pulse into an unknown filter, the resulting filter output is an approximation of the filter's impulse response. If you FFT this response, you get an approximation to the filter's transfer function!

Now let's look at a real example. I used a simple band-pass filter in this example. **Figure 4** shows how things were connected. The component values are nominally  $L = 18 \mu\text{H}$  and  $C = 1 \mu\text{F}$ . This band-pass filter has a resonant frequency equal to 37,513 Hz (i.e.,  $1/(2\pi\sqrt{LC})$ ). **Photo 8** depicts the UI that results when the filter analysis mode is on. The upper plot is the impulse response and the lower plot is the filter's transfer function. Observe the band-pass filter's peak in the FFT output. Very cool!

## SOURCES

ADP151 Low-dropout low noise voltage regulator

Analog Devices | [www.analog.com](http://www.analog.com)

dsPIC33EP512MC502 Digital signal controller

Microchip Technology | [www.microchip.com](http://www.microchip.com)

WIZ550io Ethernet module

WIZnet | [www.wiznet.co.kr](http://www.wiznet.co.kr)



[circuitcellar.com/ccmaterials](http://circuitcellar.com/ccmaterials)

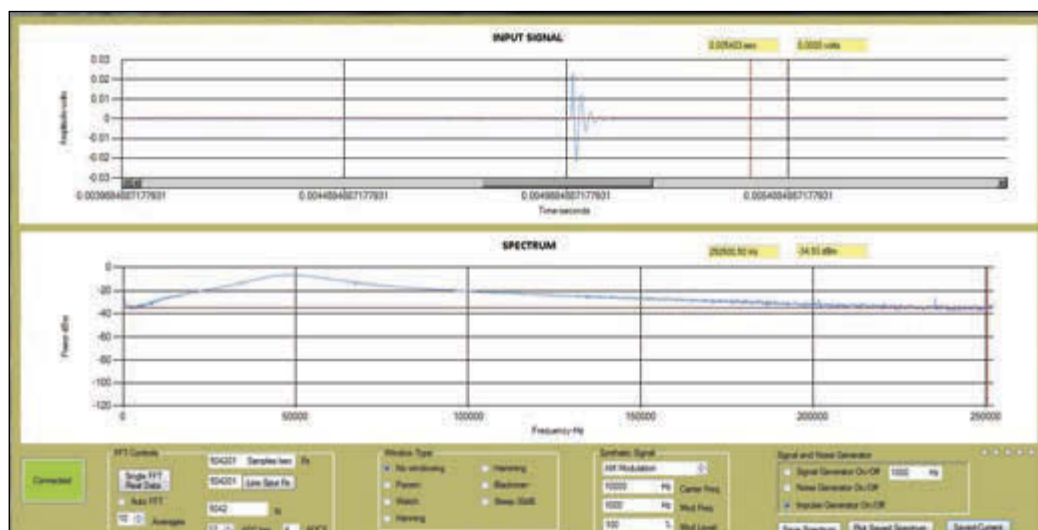


PHOTO 8

Test band-pass filter impulse response and filter transfer function

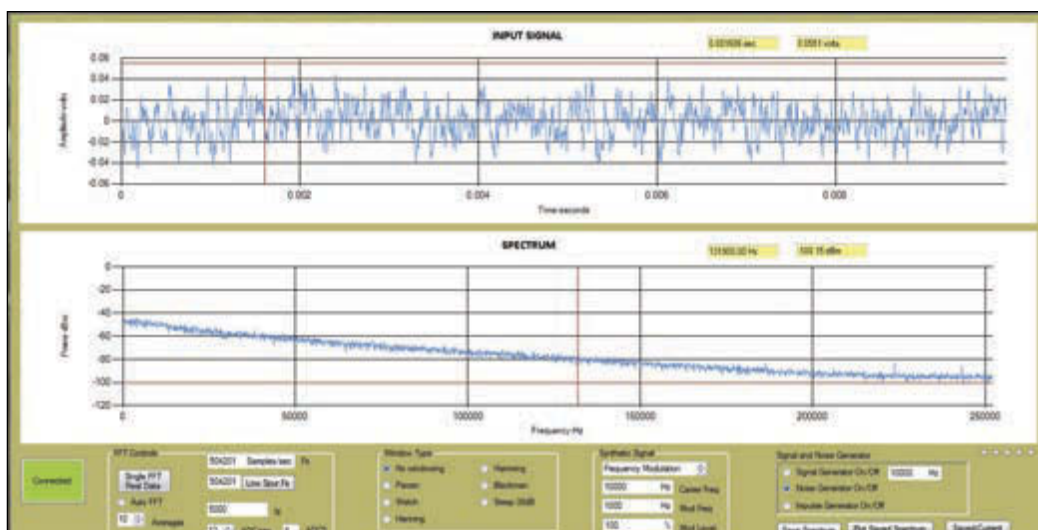


PHOTO 9


The Noise Generator connected to the analyzer produces random amplitude levels in the time domain data

Besides determining the transfer function of an arbitrary filter, you can use Filter Response Viewer to analyze unknown components. For example, if in the above test you used a known capacitor and an unknown inductor, you could read the resonant peak off the power spectrum and then use a resonance equation in reverse to estimate the L value. I intend to add this feature to the UI in the future.

## NOISE GENERATOR

The noise generator uses the same PWM and LPF combination that is used for the signal generator. The difference, however, is that once per PWM cycle the duty cycle is changed to a random 8-bit number. The LPF output is consequently a random level. **Photo 9** shows the system in the noise generation mode. Observe the random nature of the time domain data in the upper plot. The lower plot shows the FFT of this random signal. The noise is not “white”

(flat across frequency), but it is still useful in many applications. Before I got the Filter Analysis mode working so successfully, I was using this noise signal to drive filters I wanted to analyze. To do this, you first get the spectrum of the noise signal generator connected to the system’s input, without the filter present. Then you save that spectrum by pressing the button in the lower right of the UI. Next, you insert the filter and press the Saved-Current button in the lower right of the UI. The spectrum is then the transfer function of the filter being analyzed.

I guess this write-up got a little long. There is a lot here to talk about. There are also many improvements that I am contemplating. The next big feature I am going to attack is to add radio signal demodulation to the system’s feature set. Since I can already “see” signals up to several megahertz. By using simple demodulation techniques, I will be able to “listen” to all kind of radio signals. 



# Reverse Engineering Review

## Insights from Four Projects



Reverse engineering an electronic system can be a rewarding yet challenging endeavor. This article details four reverse engineering projects and how the initial challenges were overcome.

*By Fergus Dixon (Australia)*

**R**everse engineering is the process of starting with a finished product and finding out exactly how it works. Normally, an electronic project starts with a requirement from the customer. The engineer then prototypes and tests any new circuit designs, draws the schematic, lays out the board, writes the firmware, tests everything, and then repeats the design-fix-test process continually raising the quality, performance, and reliability of the system. When the product passes the test criteria and is ready, it is shipped to the end user. Reverse engineering is the opposite. The engineer starts out as the end user, discovers what the product does, studies how it works, and derives out the schematic from the board layouts and sometimes even works out the firmware. Reverse engineering is an extreme form of engineering. It's like comparing a leisurely Sunday run to finishing a marathon or a hill climb to climbing Everest. (Note how sneaking in references to extreme sports makes this sound much cooler.) This article describes four separate reverse engineering projects and how the initial challenge was

overcome. One of the projects involves reading the code from a locked Microchip Technology microcontroller.

### PROJECT 1

One of my colleagues, who is the biomedical manager at a large hospital, was having issues with hospital gas panels failing and wanted a cheaper repair option. The gas panels were designed and manufactured by a local company that had gone bankrupt several years earlier. After taking a unit away to look it over, I found that the gas panel had a bright green vacuum fluorescent display with connectors for up to 24 inputs. Each input would show whether the gas supply was normal or in alarm, and thanks to some clever design would also show on the display an open or short circuit on the cable to the gas cylinder. There were 0 to 5-V analog inputs. There was a rechargeable 3.6-V battery on each gas panel to save RAM memory on power off (now this is usually done with EEPROM memory). The problem was that the gas panel would lose its memory when the

battery failed or dropped below 2 V. Random characters would then appear on the screen, and the system error light would illuminate (see **Photo 1**).

The suggestion was to look at the microcontroller since this is usually where the memory was stored. The microcontroller was the popular but now obsolete Motorola 6805. A quick glance at the datasheet showed that it had no EEPROM or nonvolatile memory (i.e., memory that is not affected by a power-off cycle). Looking at the chips, one of the eight-pin chips was a Philips PCF8570 I<sup>2</sup>C memory chip with 256 bytes of memory and there were five of these making up to 1,280 bytes of memory. Since the display had one line of 40 characters and there were 24 alarm inputs each with an alarm message, a start-up message, and a normal operation message, there needed to be at least 26 messages × 40 characters or 1,040 characters, so this had to be where the message data was stored. The battery was the backup for this RAM, so it appeared the memory was battery-backed RAM (BBRAM). The memory voltage supply was held up by the battery, but when the battery failed, it dragged down the voltage supply rail. A quick inspection of the battery terminals showed some fuzziness and fine crystals indicating that it was leaking and was probably not operational any more.

To read the memory required an I<sup>2</sup>C reader. The easiest way to do this at the time was to make a prototype board using a Atmel ATmega32 and use two pins to drive the SDA and SCL lines. The output data was ported through a RS-232 converter to a computer. I wish I had more research here since I<sup>2</sup>C reader/writers are very cheap and I did not realize that the Atmel TWI port was actually an I<sup>2</sup>C port but with a different name due to the Philips trademark. Anyway, I read the datasheets for the I<sup>2</sup>C interface and made a small circuit which could read and write to one of the I<sup>2</sup>C memory chips. The I<sup>2</sup>C interface consists of Start bits, Write bits, Read bits, and Stop bits with the SCL clock line always being driven from the microcontroller but the SDA line being bidirectional (i.e., an input or an output).

After building the prototype and reading and writing to memory, the circuit managed to read and write the whole 1,280 bytes of memory in the gas panel, which was quite easy since the memory chips addresses lines were sequential (i.e., 000 001 010 011 100). The microcontroller was removed from the PLCC socket during this process to prevent any spurious I<sup>2</sup>C communications. The next part was to read the memory from a working machine since the gas panel I had was now full of corrupted data. After a few trips to

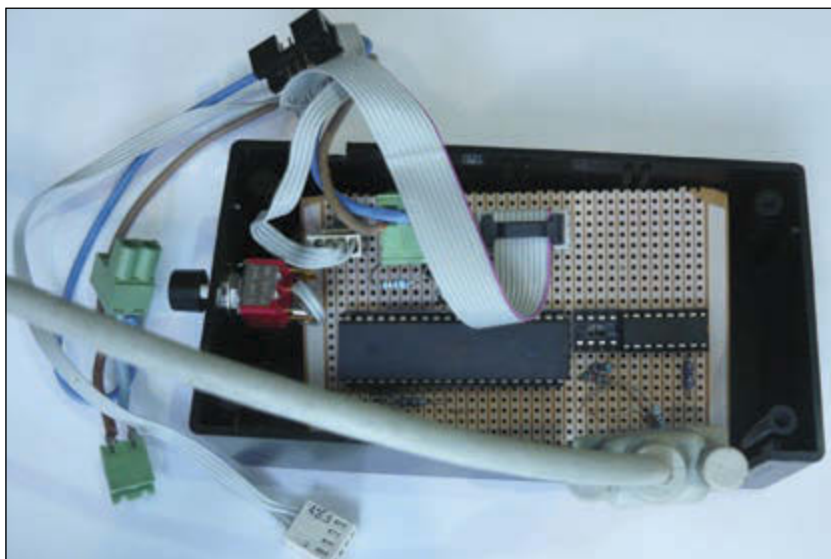


**PHOTO 1**

Medical services alarm panel

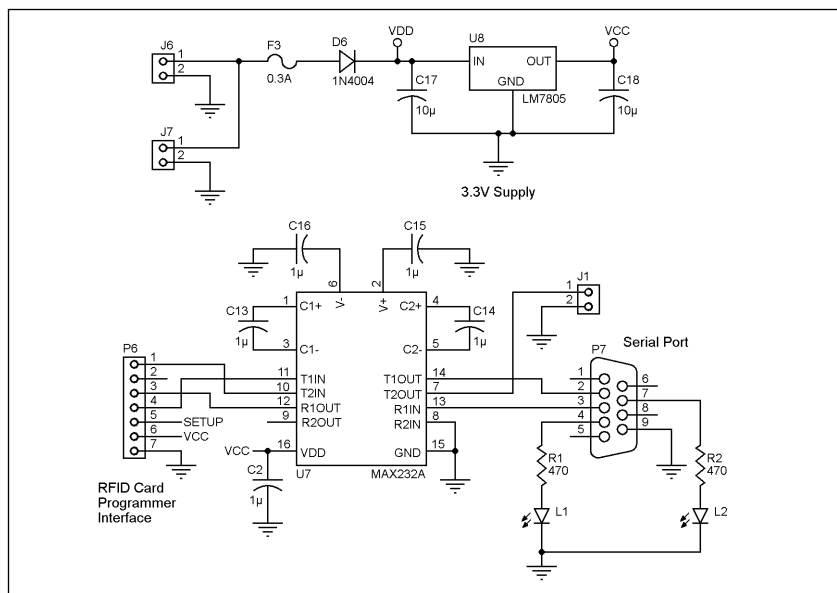
the hospital later, I had the memory in a file, and straight away, the alarm messages could be seen as ASCII data. Each message was preceded by one byte which determined whether the gas alarm input was a warning, an alarm or turned off (see **Photo 2**).

The last challenge was the system error light. Even though the gas panel could now be programmed with the correct messages, the system error light remained on. A quick solution was to remove the driving resistor to this light, but then that meant any real system error would be missed. Looking through the gas alarm panel memory again showed that each alarm message had a trailing byte which looked like a checksum. The simplest checksum can be found by adding up all the bytes and this almost worked. Then I realized that the trailing spaces in the alarm messages were also used in the checksum and the game was over. Since then, a lot of gas panels have been able to repaired using the prototype circuit.



**PHOTO 2**

Gas panel programmer

**FIGURE 1**

The RS-232 to 5-V converter

## PROJECT 2

The next project involved a beauty therapy device. This device involved a pump which supplies cyclic vacuum pressure to ladies' bottoms (seriously) to turn golf ball skin textures into a ping pong texture (according to the advert). This is also called "body contouring." The machine had a graphic monochrome display and had a countdown counter which recorded how many treatments were left. When the counter was at zero, a card could be swiped across the machine resetting the treatment counter to 25 or 50 more treatments. The problem was the customer could not source supply of these cards and asked if the cards could be replicated.

**PHOTO 3**

RFID card programmer

When the machine was disassembled, there were two boards, a power supply board and a control/display board. To remove the control board, I had to remove four PCB clips, and over one clip was a captive magnet which acted as a security device. On the other side, the RFID reader/write board could be inspected which showed a model number. A Google search revealed the manufacturer of the RFID cards was also in France (same as the manufacture of the machine). After several emails, I had all the datasheets and we purchased a few sets of the readers and cards to play with.

The simple RFID reader/write used a basic TTL-based (0 to 5 V) serial port. Understanding how the interface worked was not trivial, but after a few days the RFID interface started to work. Worryingly, the RFID card had a 64-bit encryption key. If each combination of the key was checked running at one test a second, it would take 58 billion years and I didn't really have that sort of spare time, not to mention what the bill would be to the customer!

Then the project took a wrong turn. A cable between the power board and the control board had French-style blade connectors that were impossible to insert without shorting out two pins—nowhere near as good as the standard IDC ribbon cable box connectors. During one test, I noticed I had left this cable off the control board, which explained why there was no display, so I powered off the unit. Unknown to me at the time, the power board retained charge and the voltage levels were still active for a few hours when turned off especially with no load connected (i.e., the control board). Plugging the control board with the blade connector with power off created a small spark which was a bad sign. On the next power on, the display was blank and so the unit was now faulty. After a bit of cursing and venting over the next day, the best course of action was decided to let the customer know straight away and get another machine, and get this project finished before any more problems could happen.

The problem of breaking the 64-bit security code was proving to be a bigger hill to climb than at first thought especially since the hill was in the order of 58 billion meters times higher than expected (maybe higher than Everest). Sometimes the best way to deal with problems is to go around and not keep banging into them in order to get over them—the so-called workaround or lateral thinking.

So rather than breaking the code, what about programming a card and reader with a new code and then swapping the RFID reader out with the new reader? Since the data was sent through a serial port and serial ports cannot be encrypted, the best way would be



to swipe a card and eavesdrop on the RS-232 lines at the same time. Since I had used the two cards I had, it was back to the customer to take one of the last cards he had and there were signs of the customer's patience wearing a bit thin now. Anyway, back at the ranch, I wired in a serial port on the laptop to eavesdrop and then took a break to prepare mentally (see **Figure 1**). When the card was swiped, it all worked perfectly, and I could see that 16 bytes were being read out of the RFID card (see **Photo 3**). One of the numbers was a 50 indicating 50 treatments, while another number was 90 for the number of days the card would be valid. And that was pretty much it. The process would now be to reprogram some RFID readers, which took around 10 s each, and then swap the RFID reader in each unit. The first beauty salon took around an hour and went well. The same day, three more machines were upgraded in various parts of the city and then I handed over the upgrade work to a service company who could travel into country areas. In the instructions there was a large message to not disconnect the interconnecting cable.

So overall, the project was mostly a success except for the one damaged machine. Later on, there was some more work done by reprogramming a lot of cards. Since each card took around 20 s to program, this was easy work compare to the original project.

### PROJECT 3

The next project involved a business relationship between two companies that had soured. The contract assembly company had been making a PCB assembly for several years for a local company and doing an excellent job, so good that they decided it would be in everyone's best interests if they not only manufactured the boards but also sold them direct to the public. The customer had paid for and owned the IP and was right to be outraged.

While this project was simple, there was still 10 years of design behind it. The aim was to bring the design back in house. The schematic and layout could be worked out but writing the firmware from scratch was a formidable task. Using the board, the track connections were used to work out a schematic and this schematic was then used to re-layout the board. This layout was then checked against the original board and was a double check that the schematic was accurate.

With the firmware, a Microchip Technology PICkit2 was plugged into the board to read the firmware. After the program was read, it showed all zeros. It was unlikely that the real firmware would be all zeros and the

### ABOUT THE AUTHOR

Fergus Dixon holds a BE from Sydney University. After working for 12 years in various fields such as packaging, mining, and medical and control systems, he started Stirling Rock International (SRI), which provides custom electronic designs. After 14 years (and two Australian design awards), Fergus now runs several businesses, including Electronic System Design and Virtronics. He spends most of his spare time maintaining and improving the popular program "Simulator for Arduino."

PICkit showed in the configuration byte that the ROM lock bit was turned on. Well that was disappointing but to be expected. The next step was to call for reinforcements. The Arduino is an excellent kit that allows for fast prototyping with a free C compiler and IDE. Since Microchip makes available the programming interface in a datasheet, an Arduino Uno with a breadboard was used to build a custom Microchip programmer with the hope of finding a side door into the firmware. After all, apart from the three power pins (VCC/5V, GND, and VPP/12V), there was only the data and clock pins. How hard could it be? Well it took a full day of programming to write an Arduino sketch which could read and program the data to a new chip. One difficulty was to apply 12 V to the VPP pin before turning on the 5 V to the VCC pin. The most obvious way to read the firmware was to turn off the lock bit in the configuration byte and then read the firmware. After several attempts, the lock bit was turned off and then the code

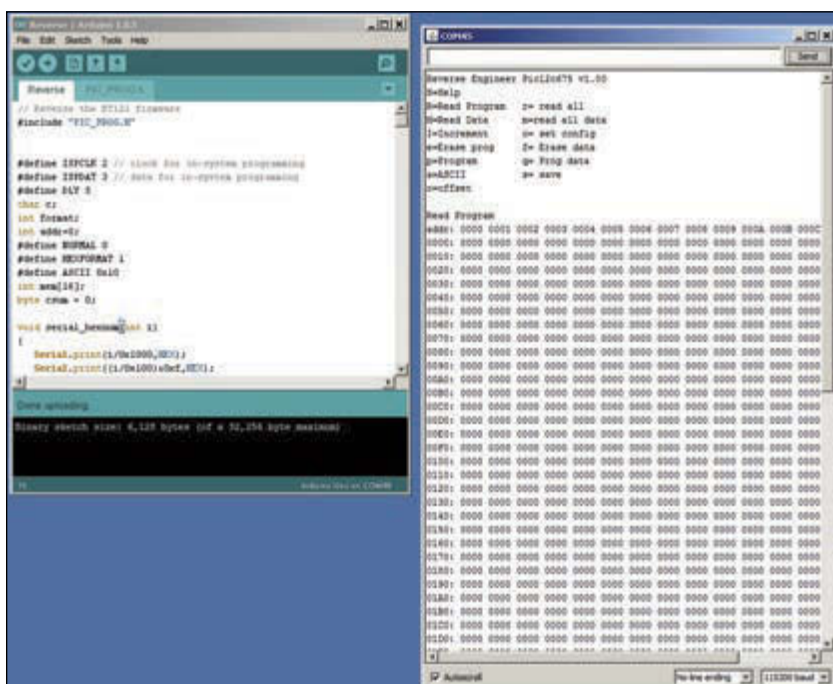


PHOTO 4

Arduino screenshot

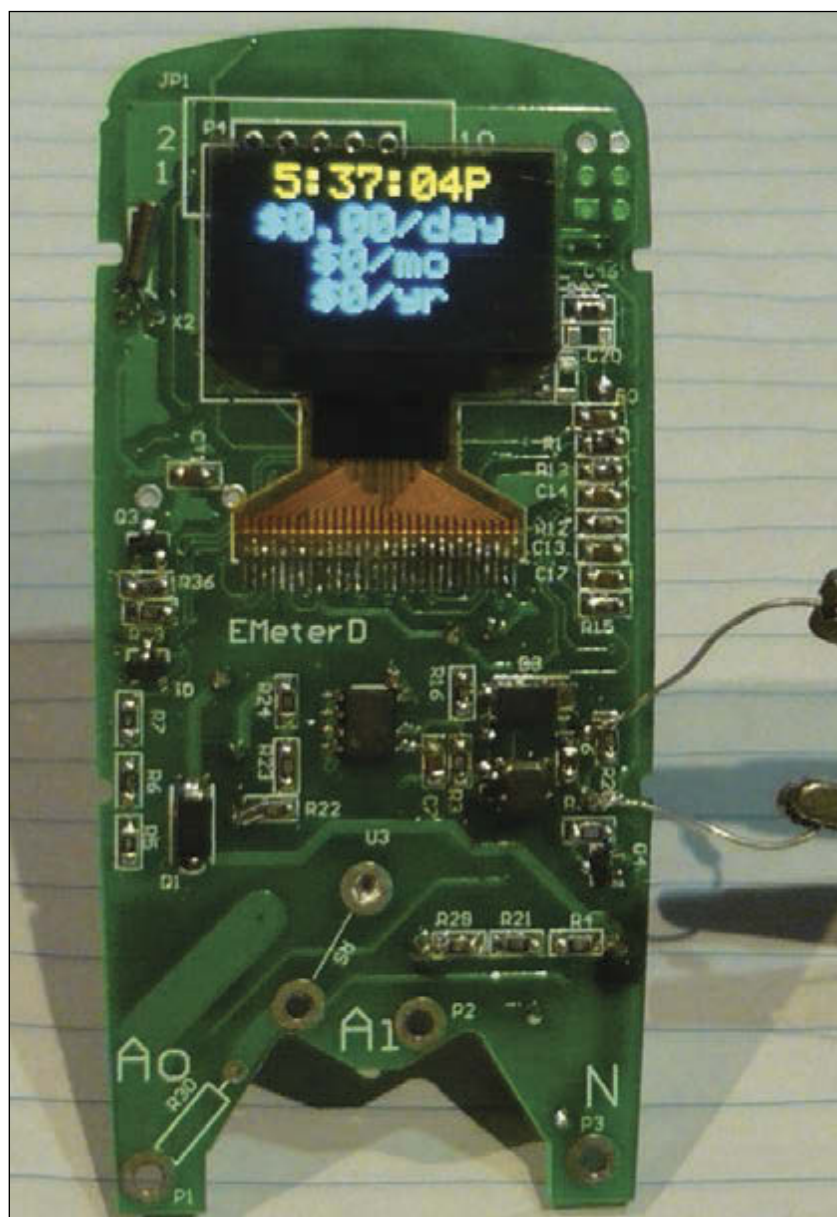


PHOTO 5

The new energy meter



circuitcellar.com/ccmaterials

**RESOURCE**

Digital Millennium Act, [http://en.wikipedia.org/wiki/Digital\\_Millennium\\_Copyright\\_Act](http://en.wikipedia.org/wiki/Digital_Millennium_Copyright_Act).

**SOURCES**

PIC12F675 Microcontroller and PICkit2  
Microchip Technology | [www.microchip.com](http://www.microchip.com)

RFID cards and card readers  
STid | [www.stid.com](http://www.stid.com)

X2 Self-healing capacitors  
Vishay Intertechnology | [www.vishay.com](http://www.vishay.com)

was read. When the bytes starting appearing in the Arduino Serial Monitor in a non-random manner, there was a great feeling that the door had been opened (see **Photo 4**). Doing a quick upgrade to the Arduino sketch, the bytes were then also shown as ASCII characters in the serial monitor. When readable messages such as "Version 1.03" appear, it is looking good. Since the customer had paid for the firmware design, there were no ethical dilemmas here.

Microchip quotes the Digital Millennium Act for all the security protection devices on its parts, and it is worth reading up on this. After this project, I added a page to my website offering a service of unlocking Microchip firmware for a small \$60 deposit. Well, there were many responses, but not one offered any money with even this amount being too much. In the end, emails were ignored (there were too many, maybe 50 a month), the webpage was taken down, and the whole idea was dropped, which sorted out any ethical dilemma problem.

**PROJECT 4**

I'm currently working on an energy meter, but the reverse engineering part is finished. With rising energy prices, climate change a hot topic and a drive to become more green, energy usage is very important and this is a growing industry for engineers. The aim of the energy meter is to monitor the power from a mains power outlet in the home and send it wirelessly to an iDevice or smart phone. Since energy meters are very common and also very cheap, we bought a few to find out how they worked and what their weaknesses were. The answer was very interesting.

With the energy meters, there were three main circuit modules: a power supply to create a 5-V supply from 230 or 110 VIN; an energy-measuring circuit including current and voltage sensors; and a microcontroller and display. With the power supply, most of the circuits used an X2 capacitor and low value resistor (i.e., 100  $\Omega$ ) and a 12-V Zener diode that would generate a 12-V supply that could only supply 30 mA at most. This is called a capacitive dropper power supply. The drawback with these circuits is the X2 capacitor can degrade over a few years and then the supply will stop working. Metalized polyester film capacitors are better with the best option being an XY capacitor which has self-healing properties.

With the energy measuring circuit, the current sensor was either a current transformer or a low value shunt such as 0.01  $\Omega$ . The current transformer needs some extra wiring to wrap the wire through the center of the current transformer and the cost is a lot higher than a simple low value resistor or




shunt. On all the off the shelf energy meters, the energy measuring chip usually had the markings rubbed off but they were all the same 24-pin-wide SOIC package. On one board, the chip markings were still there, which showed the chip to be a Cirrus Logic CS5460 chip.

The microcontroller was usually a black blob called a chip-on-board. Black blobs are used in mass production to keep the cost down. At around \$12 for the cheapest energy meter, this method was definitely working. The displays were usually LCD and this was the biggest problem. LCDs are great for outdoor use and the main benefit is high visibility in daylight, low cost, and low current drain. The downside was that the LCDs just were not visible inside especially at night, and if the power outlet was at ground level they were impossible to view at any time. Another strange point was that every board had a 24C02 eight-pin memory chip. The 24C02 is a 2K EEPROM chip and was a strange choice considering much larger memory chips are now commonly available. The reason for this could be this chip was from old surplus stock, which meant all the meters were designed and made at the same factory (in China, of course), which also would explain why the meters all used similar LCDs.

The main problem with these energy meters was that they were almost unusable. After being set up, they would be forgotten about in a short period. Basically, they were complicated, boring, barely visible and mostly unusable. So the plan for the new energy meter was to design one which was exciting to use, easy to use, had a bright display, and could be connected remotely. **Photo 5** is current circuit of the new energy meter design which uses a high contrast OLED display.

### RE-ENGINEER CAREFULLY

So overall, reverse engineering can be rewarding once the initial learning curve is climbed. Reviewing someone else's circuits is a great way to pick up new tips and tricks on design methods. The downsides are the legal issues since no company wants their designs copied. Of the four projects above, two of them resulted in legal issues and one more resulted in a nasty email from a competitor. And whatever you do, think twice about reverse engineering any military hardware unless you are a thrill seeker. Before tackling any reverse engineering project, ask yourself if it is really worthwhile and always make sure the customer takes legal responsibility. 

# When it comes to robotics, the future is now!

Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies, making it ideal reading for beginners and experts alike.

With this book, you'll learn about:

- Communication Technologies
- Control Robotics
- Embedded Technology
- Programming Language
- Visual Debugging... and more



**ADVANCED CONTROL ROBOTICS**  
HANNO SANDER

**Get it today at**  
**ccwebshop.com**



## RS485/422/232/TTL

### ASC24T

RS232<=>RS485 ATE Converter



\$45.00 board only

### IBS485HV

5 Port Isolated

RS485 Repeater  
\$349.00



Enclosures, Cables,  
Power Supplies and Other  
Accessories



### INTRODUCING THE SMFCOMX!

- Converters
- Repeaters
- Multi-Repeaters
- Hubs
- Fiber Optics
- Isolators
- Extended Distance Units
- Serial to Digital I/O
- Large Multi-Drop Networks
- Custom Units & Smart Units
- Industrial, 3.0 KV Isolation

**RESmith**  
WWW.RS485.COM

Call the RS485 Wizards  
513-874-4796  
www.rs485.com

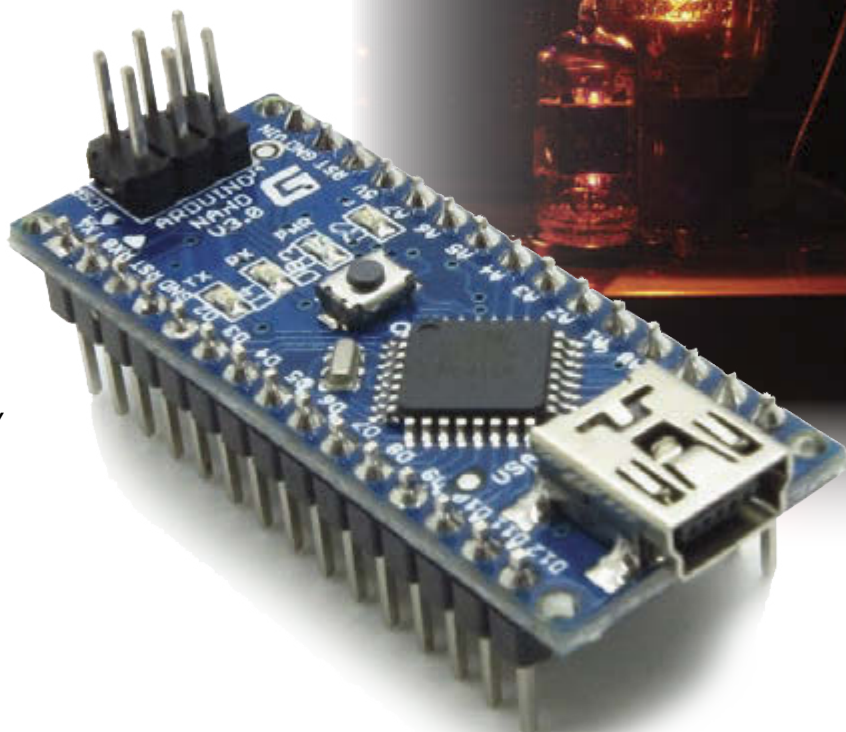


# Budgie

## An Arduino-Based Tube Stereo Preamplifier

In the 1980s, commercial stereo preamplifiers started using IC microcontrollers that permitted cleaner designs with push-button control, relays for signal switching, and a wireless remote. While reading an article about the Arduino, Shannon realized these modern features could easily be incorporated into a DIY preamplifier design.

*By Shannon Parks (US)*



**PHOTO 1**

The Budgie preamplifier contains a single-ended, Class-A, all vacuum tube audio path.

In recent years, open-source Arduino prototyping boards have become an attractive option for hobbyists. You can start tinkering with an Arduino for less than \$10 and a programmer tool is not needed. All you really need is a PC with a USB cable. The intuitive Arduino software enables you to compile and load programs called “sketches” onto the Arduino board. It consists of an integrated development environment (IDE) and C-like programming language and is compatible with Windows, Mac OS, and Linux. Arduino “shields,” which are add-on modules, permit fast prototyping with many I/O capabilities. During a single afternoon, a useful gadget—possibly a garage door monitor or wireless attic temperature sensor—can be assembled with minimal effort. Browsing the Internet, you can find applications that range from ingenious to ridiculous. The Arduino prototyping system is incredibly versatile.

Several of the Arduino's most often used capabilities became the genesis of my Budgie preamplifier (see **Photo 1**). I found these features were incredibly useful:

- A bank of relays could switch between the four stereo inputs as well as control mute, standby, gain, and bass boost settings.
- A red power LED could use PWM to indicate if the preamplifier is muted or in standby.
- An IR receiver with a remote could control a motor-driven volume potentiometer, change the source input selection, and turn the unit on/off. Any IR remote could be used with a code learning mode.
- A backlit display could easily show all the settings at a glance.
- Momentary push buttons could select the input device, bass boost, gain, and mute settings.
- Instead of using several Arduino shields wired to an Arduino board, all the circuits could fit on one custom PCB along with the power supply and the microcontroller (see **Photo 2**).

While there are many versions of Arduino boards, I chose the Arduino Nano. At only 0.73" × 1.70", the tiny Nano can be embedded using a 32-pin dual in-line package (DIP) socket, which cleans up the design. It can be programmed in-circuit and be removed and easily replaced (see **Photo 3**).

The audio circuit remains the heart of any preamplifier. With a little creativity, the well-regarded 12B4 triode can be used. Its high perveance means that current will flow with a much smaller cathode-plate voltage differential than is typical with vacuum tubes. An external 24-V desktop supply can provide all the voltages the tube requires without need for further regulation, which simplifies the design.

The Arduino boards, as with all microcontrollers, are limited by their number of I/O pins, which are simply referred to as "pins." The Nano has 14 digital pins (D0 to D13) and eight analog pins (A0 to A7). Digital pins are either low (0 V) or high (5 V). As an output, a digital pin can source up to 40 mA and do tasks (e.g., turning on an LED or forward biasing a transistor switch). A digital pin can be set up as an input to process the remote control code stream of digital 1s and 0s received from the IR sensor.

The analog pins have an internal ADC, which enables 1,024 values (10-bit resolution) between 0 and 5 V. This is useful for reading voltages in approximately 5-mV increments. The Budgie preamplifier uses one such input to monitor a voltage divider circuit implemented with five 1-k $\Omega$  resistors in



series. This pin receives four unique voltage levels from the push buttons to determine which was pressed. A special feature of the analog pins is that they can also be configured as digital pins. The Budgie does this for the display. For this application, A0 is renamed D14, A1 is renamed D15, and so on.

## SHIFT REGISTER CIRCUIT

The Budgie preamplifier uses a serial-in, parallel-out (SIPO) shift register to drive a bank of relays (see **Figure 1**). Only four Arduino digital outputs—enable, clock, latch, and data—are needed to control eight DPDT relays. These correspond to the four outputs labeled D3, D4, D5, and D7 shown in **Figure 2**. The Texas Instruments TPIC6C595 shift register used in this project has heavy-duty field-effect transistor (FET) outputs that can handle voltages higher than logic levels. This is necessary for operating the 24-V relays. It also acts as a protective buffer between the Arduino and the relays.

## POWER LED

The red power LED is directly driven by the Arduino (see **Figure 2**). Normally the output (D6) is high with the maximum LED brightness limited by a 200- $\Omega$  resistor that biases the LED at about 16 mA. When the preamplifier output is active, the PWM control is not used as it can add 400-Hz noise (the PWM switching frequency) to the audio circuit. However, it can be used when the output is muted or in standby.

## IR REMOTE CIRCUIT

The D8 input is the only digital pin configured as an input (see **Figure 2**). An IR sensor with a built-in preamplifier sends all sensed IR commands from a remote control to the microcontroller.

**PHOTO 2**

All the circuits have been placed on one custom PCB along with the power supply and the microcontroller.

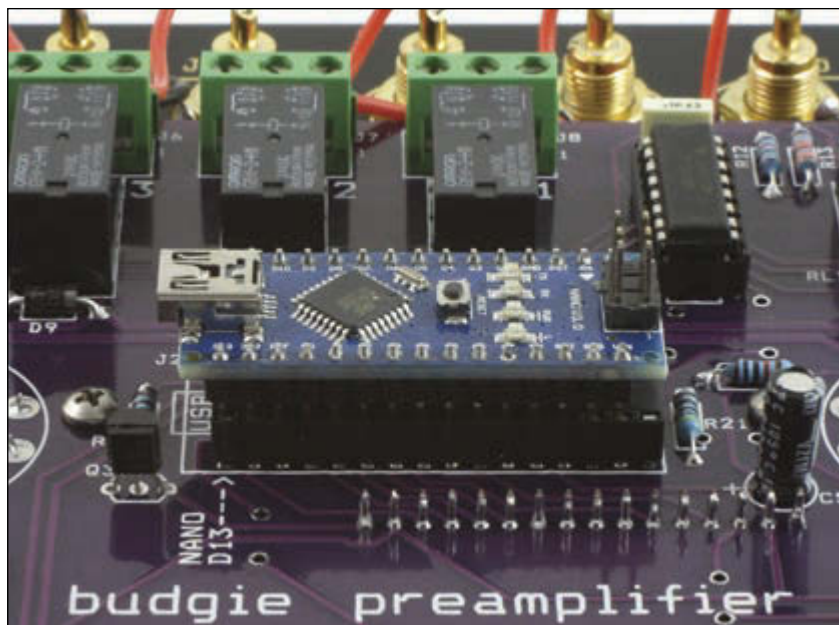


PHOTO 3

The Arduino Nano is a small, complete, and breadboard-friendly board based on Atmel's ATmega328 microcontroller.

A special software library called "IRremote" simplifies the handling of the raw datastream into 4-byte codes. This data can be processed in real-time or stored in the EEPROM during learning mode.

## MOTOR DRIVER FOR THE VOLUME POTENTIOMETER

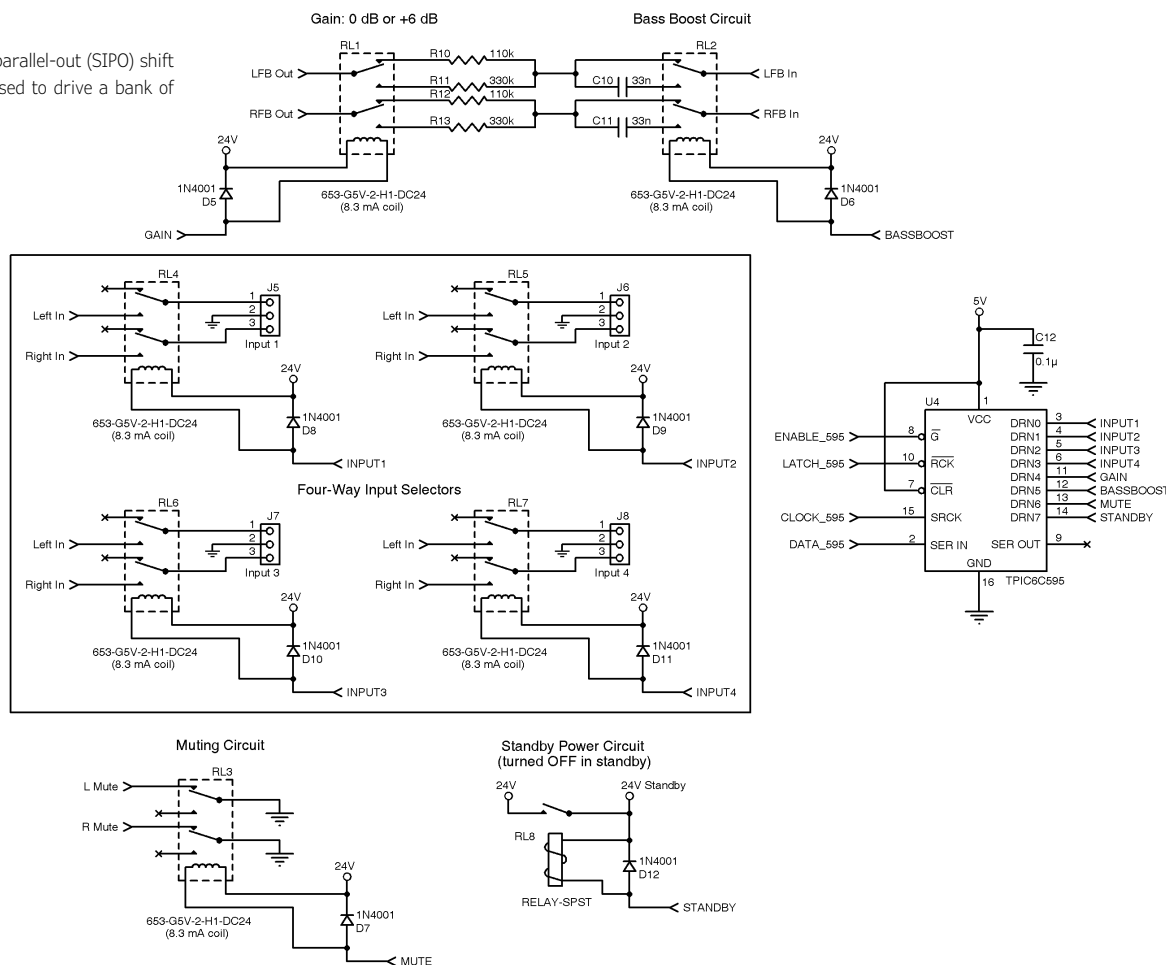
A Texas Instruments SN754410NE specialized integrated circuit (IC) controls the motor of the Alps volume potentiometer and acts as a buffer between the microcontroller and a motor. The D10, D11, and D12 outputs are used. The IC simplifies a process that would otherwise need careful design. Two digital pins control the direction of the motor on the volume potentiometer while a third digital pin enables the motor driver.

## 1602 LCD

The 1602 LCD—16 characters per line by two lines—is probably the most used display in the Arduino realm. It is controlled using the standard software library "LiquidCrystal" in 4-bit mode, so a total of seven digital pins are needed: register select (RS), enable (EN), backlight, and four data lines. These

FIGURE 1

A serial-in, parallel-out (SIPO) shift register is used to drive a bank of relays.





are assigned to Nano outputs D9, and D13 through D18 (see **Figure 2**).

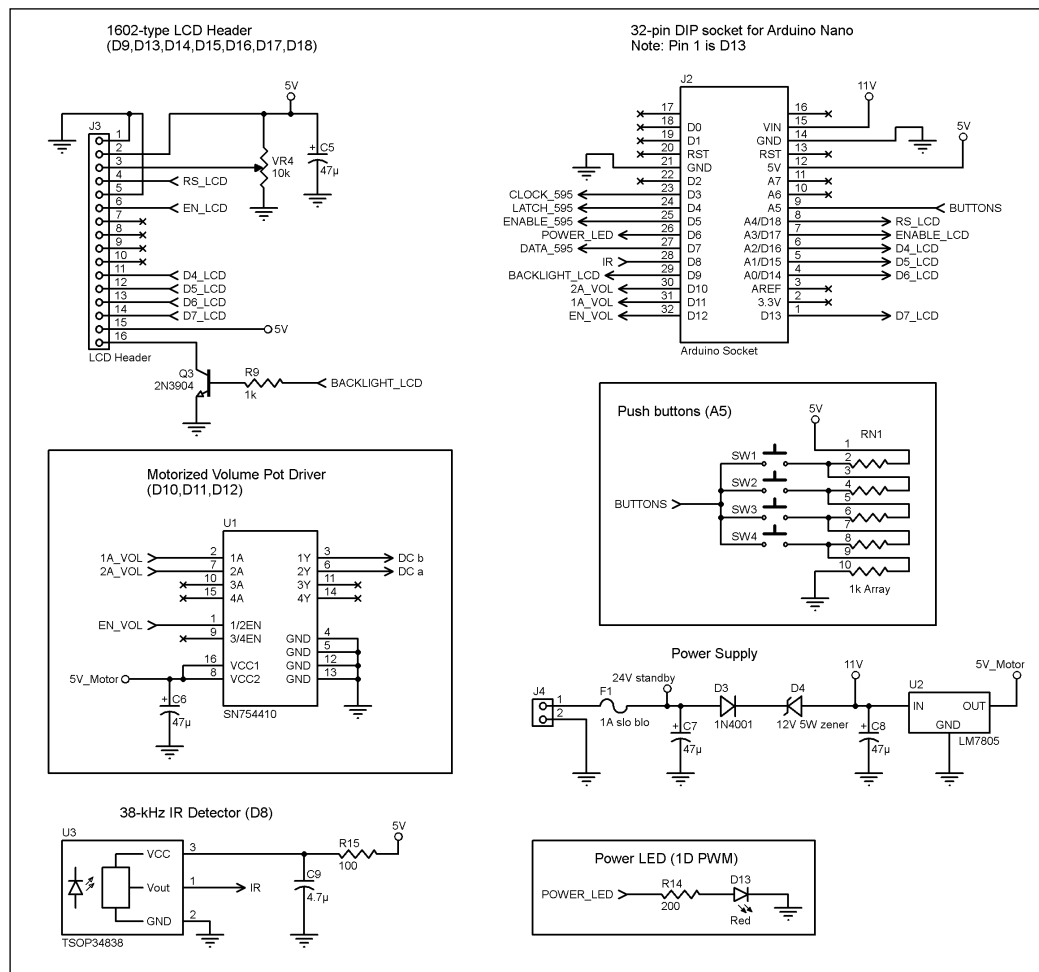
## PUSH BUTTONS

A voltage divider string of resistors is connected to four push buttons: input, bass boost, gain, and mute (see the A5 input in **Figure 2**). Each button has a different voltage level that the analog input reads to determine

which button was pressed. Multiple keys pressed at the same time will generate unique voltages and be processed as special functions.

## POWER SUPPLY

The external 24-V desktop supply must be a Class-I grounded type and should be rated above 1 A. Some supplies may have a tendency to trip their current limit at initial power-up due

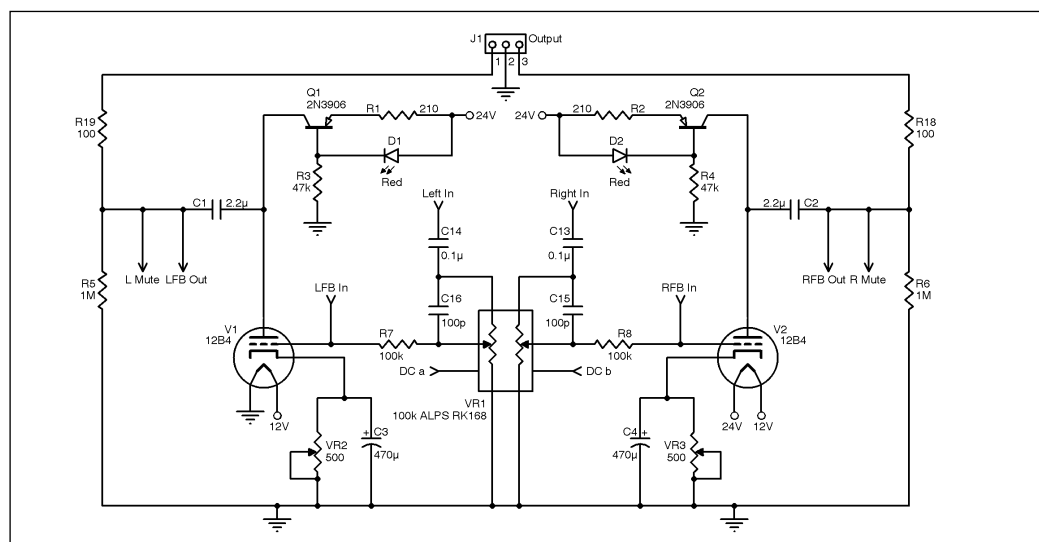


**FIGURE 2**

These Budgie diagrams show how to set up the Arduino Nano, LCD, power supply, push button, IR, and motor control circuits.

## ABOUT THE AUTHOR

Shannon Parks built his first tube amplifier—a clone of the Dynaco ST35—in 2001 and was immediately hooked. He manages the DIY-tube forums and is the owner of Parks Audio, LLC ([www.parksaudio.com](http://www.parksaudio.com)). He lives with his wife and daughter in Mahomet, IL.

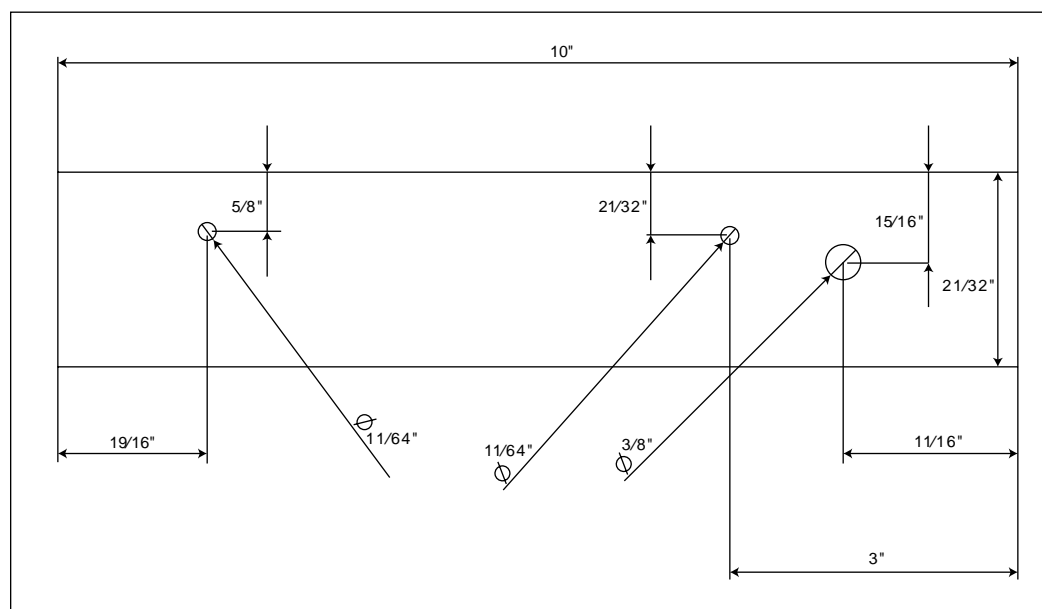


**FIGURE 3**

The 12B4 audio circuit has about 23-dB gain, which is more than is needed.

**FIGURE 4**

The template shows where to drill the holes when assembling the Budgie's chassis.



to the heavy inrush from the cold tube filaments. The relays and tubes operate at 24 V without needing further regulation to eliminate wasted power and heatsinks on the Budgie PCB. The 24-V supply also powers two lower current 5-V supplies. A polarity diode and then a 12-V power Zener diode drop the incoming 24 V to around 11 V so the linear regulators don't dissipate too much heat from the voltage differential (see **Figure 2**). The 7805 regulator on the PCB (U2) is solely for the motorized Alps volume potentiometer. Meanwhile, the 5-V regulator on-board the Nano powers all other 5-V circuits including the display backlight.

## AUDIO CIRCUIT DESCRIPTION

The 12B4 triode was originally designed to be used in televisions as a vertical deflection amplifier. New-old-stock (NOS) 12B4s still

exist. They can be purchased from most US tube resellers. However, a European equivalent doesn't exist. The 12B4 works well in preamplifiers as a one-tube solution, having both high input impedance and low output impedance, without need for an output transformer. An audio circuit can then be distilled down to a simple circuit with few parts consisting of a volume potentiometer and a grounded cathode gain stage.

The 12B4 has about 23-dB gain, which is more than is needed. This extra gain is used as feedback to the grid, in what is often referred to as an anode follower circuit. The noise, distortion, and output impedance are reduced (see **Figure 3**). Using relays controlled by the Arduino enables switching between two feedback amounts for adjustable gain. For this preamplifier, I chose 0- and 6-dB overall gain. A second relay enables a bass boost with a series capacitor.

You only need a lightweight 15-to-20-V plate voltage to operate the 12B4s at 5 mA. Linearity is very good due to the small signal levels involved, as rarely will the output be greater than 2 V<sub>pp</sub>. A constant current source (CCS) active load is used with the 12B4s instead of a traditional plate resistor. This maximizes the possible output voltage swing before clipping. For example, a 12B4 biased at 5-mA plate current with a 20-kΩ plate resistor would drop 100 V and would then require a 120-V supply voltage or higher. Conversely, the CCS will only drop about 2 V. Its naturally high impedance also improves the tube's gain and linearity while providing high levels of power supply noise rejection. A single resistor from the supply to the emitter sets the current, which can be

## RESOURCES

Future Technology Devices International (FTDI), Ltd., [www.ftdichip.com](http://www.ftdichip.com).

Parks Audio, LLC, [www.parksaudiollc.com/arduino](http://www.parksaudiollc.com/arduino).

## SOURCES

RK168 Series volume potentiometers

Alps Electric Co., Ltd. | [www.alps.com](http://www.alps.com)

Arduino Nano

Arduino | <http://arduino.cc/en/main/software>

ATmega328 Microcontroller

Atmel, Corp. | [www.atmel.com](http://www.atmel.com)

TPIC6C595 Shift register and SN754410NE IC

Texas Instruments, Inc. | [www.ti.com](http://www.ti.com)

calculated with the following formula:

$$\frac{(\text{voltage drop of red LED} - V_{BE})}{R_{SET}} = \frac{(1.7 \text{ V} - 0.65 \text{ V})}{210 \Omega} = 5 \text{ mA}$$

The adjustable cathode resistor is initially set to 300  $\Omega$  for good performance, but the lowest distortion can be accomplished by adjusting the trim potentiometer to achieve maximum output signal. This tweak can be done with a free smartphone app such as Signal Generator outputting a 500-mV 1-kHz sine wave. Use any digital multimeter (DMM) set to AC to measure the output level and adjust the cathode potentiometer for maximum output. Note that the DMM doesn't need to be TrueRMS as the measurement is relative, not absolute. Also, don't forget to adjust the volume potentiometer all the way clockwise.


## CHASSIS ASSEMBLY

The Budgie chassis uses an inverted Hammond powder-coated 6" x 10" x 2" box. I can provide a front panel express file for a ready-made top plate via my website (see Resources). Most of the chassis holes are on this one panel, but three holes must be drilled into the front of the chassis for the power LED, the IR sensor, and the volume control. A template with detailed dimensions facilitates drilling these holes (see **Figure 4**).

## COMPILING & LOADING ARDUINO CODE

The first time you connect the Arduino Nano to your computer with a USB cable, the device drivers for the Future Technology Devices International (FTDI) virtual com port will be prompted to load. Once you successfully install the drivers, start the Arduino software. First make sure you have your COM port set correctly in the Arduino software (Tools> Serial Port...). Then configure the software for the Nano: "Tools>Board> Arduino Nano w/ ATmega328." Then go to "Sketch>Import Library...>Add Library..." and select the IR remote library from the Budgie files folder, which decodes all the raw IR data. Clicking the "Upload" icon will first compile and then upload the sketch to your Nano's flash memory. After you've uploaded your sketch, disconnect the USB connection before you power the Budgie with 24 V to prevent reverse current into your computer's USB port and possible damage.

## COMPLETED BUDGIE

The entire project can be assembled for approximately \$250 (see **Parts List**). As an introduction to the world of Arduino microcontrollers, it's a comprehensive starter kit. Elements of the design (e.g., the IR-controlled volume potentiometer) could be used as a stand-alone project. Once you experience the relative ease of microcontroller programming with the Arduino, you'll find no shortage of creative applications. 

## BUDGIE PREAMPLIFIER PARTS LIST

### Capacitors

C1, C2	2.2 $\mu$ F, 250 V, 18.8 x 12.6-15LS	667-ECW-F2225JA
C3, C4	470 $\mu$ F, 10 V, 6.3D 2.5LS	647-UVZ1A471MED
C5, C6, C8	47 $\mu$ F, 35 V, 5D 2LS	647-UVZ1V470MDD
C7	100 $\mu$ F, 35 V, 6.3D 2.5LS	647-UVZ1V101MED
C9	47 $\mu$ F, 50 V, 5D 2LS	647-UVZ1H4R7MDD
C10, C11	6,800 pF, 63 V, 7.2 x 4.5-5LS	505-FKP26800/63/2.5
C12	0.1 $\mu$ F, 63 V, 7.2 x 2.5-5LS	80-R82DC3100AA50K
C13, C14	0.1 $\mu$ F, 100 V, 7.2 x 2.5-5LS	505-MKP20.1/100/5

### Chassis

Hammond 6 x 10 x 2 Black Chassis	546-1441-16BK3
Quantity 2 Clear window plug	593-LPC020
Quantity 2 Clear window plug retainer	593-RTN150

### Circuits

U1	Half-H motor driver	595-SN754410NE
U2	5-V regulator 7805	595-UA7805CKCT
U3	IR detector, 38 kHz	782-TSOP34838
U4	Power shift register	595-TPIC6C595N
Quantity 2	16-pin DIP socket	571-1-390261-4

### Diodes

D1, D2	T-1 0.75 5-mm red LED	859-LTL-4223
D3, D5-D12	1N4001 50-V 1-A diode	512-1N4001
D4	12-V, 5-W Zener diode	863-1N5349BRLG
D13	Red power LED	645-551-0407F

### Miscellaneous

J1, J5-J8	Terminal blocks 3P	571-2828363
J2	32-pin DIP socket	517-4832-6000-CP
J3	16-pin SIP header	649-68001-416HLF
J4	Terminal blocks 2P	571-2828372
SW1-SW4	Round push-button switch	10KB012
SW5	SPST on/off rocker	540-SRB22A2FBBNN
Quantity 1	Arduino Nano V. 3.0	992-ARD-NANO30
Quantity 5	RCA panel jacks—white	568-NYS367-9
Quantity 5	RCA panel jacks—red	568-NYS367-2
Quantity 1	2.1-mm DC connector	163-4021
Quantity 4	Rubber feet	517-SJ-5023BK
Quantity 6	Hex standoff 0.187 x 0.375	534-1892
Quantity 4	#6-32 x 0.5" sheet metal screw, Phillips pan head	
Quantity 12	#4-40 x 0.25" machine screw, Phillips pan head	
Quantity 1	Red LCD 1602 display module (HD44780 or compatible)	
Quantity 1	30 x 22 Aluminum volume knob	
Quantity 2	12B4/12B4A triode	
Quantity 2	9-pin Ceramic tube socket PCB (0.75" top diameter)	
Quantity 1	24-V/1-A DC supply (three prong)	

### Resistors

R1, R2	210 $\Omega$ , 0.25 W	271-210-RC
R3, R4	47.5 k $\Omega$ , 0.25 W	271-47.5K-RC
R5, R6	1 M $\Omega$ , 0.25 W	271-1.0M-RC
R7, R8	100 k $\Omega$ , 0.25 W	271-100K-RC
R9	1 k $\Omega$ , 0.25 W	271-1K-RC
R10, R12	110 k $\Omega$ , 0.25 W	271-110K-RC
R11, R13, R16, R17	332 k $\Omega$ , 0.25 W	660-MF1/4DC3323F
R14	200 $\Omega$ , 0.25 W	271-200-RC
R15, R18, R19	100 $\Omega$ , 0.25 W	271-100-RC
R20	10 k $\Omega$ , 0.25 W	271-10K-RC
R21	4.7 k $\Omega$ , 0.25 W	271-4.7K-RC
RN1	1 k $\Omega$ , 10-pin array, isolated	269-1.0K-RC
VR1	100 k $\Omega$ , dual log ALPS RK168	688-RK16812MG099
VR2, VR3	500- $\Omega$ potentiometer	72-T93YB-500
VR4	10-k $\Omega$ potentiometer	72-T93YB-10K
F1	1.5-A fuse, Slo-Blo	576-047301.5MRT1L
RL1-RL7	Omron G5V 24-V DPDT	653-G5V-2-H1-DC24
RL8	Omron G5NB 24-V SPST-NO	653-G5NB-1A-DC24

### Transistors

Q1, Q2	2N3906 PNP	512-2N3906BU
Q3	2N3904 NPN	512-2N3904BU

Budgie PCBs are available from Shannon Parks ([www.parksaudiollc.com/arduino](http://www.parksaudiollc.com/arduino)). Part numbers in the third column indicate the items are available from Mouser Electronics ([www.mouser.com](http://www.mouser.com)).



## THE CONSUMMATE ENGINEER

# Essential Electromagnetic Compliance (Part 3)

COLUMNS

## Practical EMC Requirements

Last month, George detailed the causes of electromagnetic interference (EMI) and explained the difference between DM and CM signals. This month, he covers the practical EMC requirements for electronic instruments and systems.

*By George Novacek (Canada)*



### ABOUT THE AUTHOR

George Novacek 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. Contact him at gnovacek@nexus.com.net with "Circuit Cellar" in the subject line.

This month I'd like to consider electromagnetic compliance (EMC) requirements for airborne equipment. As you'll see, the requirements are not too different from other industries, such as nuclear, medical, and automotive, where the sustained, safe operation of electronic devices is paramount. EMC standards address more than just the effects of the EMI fields on the internal circuitry of a product. They're also concerned with the effects of the EMI generated inside the product on the outside world, as well as its effects on internal circuits. Power supply fluctuations, effects of lightning strikes, and other electrical issues are also considerations. EMC requirements for different products may vary greatly. It is up to the project engineers to establish the mandatory prerequisites. It is a good engineering practice to always provide for a healthy safety margin.

### DO-160 STANDARD

EMC requirements for airborne equipment are published in DO-160 standard. Similarly, there are specific standards for military, industrial, and consumer products. It is not uncommon to see customers increasing the mandatory requirements to ensure sufficient safety margins. EMC design engineers need to know if and why a customer increased those

test levels to avoid unnecessary and expensive inclusion of their own safety margins.

The first EMC constraint listed in DO-160 is the magnetic effect. This is to make sure that the equipment does not produce magnetic flux affecting compasses or flux gates within a certain distance. It is specified in five categories, with the most stringent one providing for a compass immediately adjacent to the equipment not to be affected. The least stringent one is for the tested equipment to be 3 m (approximately 10') away from a compass. Most electronic controllers enclosed in a metal cabinet with no strong magnetic field generated inside pass the test easily.

The power input specification is divided into several categories, for AC and DC supplies. Typical examples are 14-VDC and 28-VDC systems powered from a generator with a rectifier and a large storage battery. The type of power determines many attributes, such as voltage range, AC distortion, power factor, phase displacement, phase unbalance, ripple, inrush current, reset, momentary power interruption, transients, emergency operation, and more.

For DC operating conditions—for example, for a 14-VDC system (i.e., a 12-V<sub>NOM</sub>, six-cell, lead-acid battery)—the normal operating

voltage is 11.0 to 15.15 V. The equipment must also operate during short abnormal conditions, when the input voltage may vary between 10.25 and 16.1 V, as well as 9.0 V for emergency operation. When operating at the minimum voltage, the equipment must, in Category A, survive a 200-ms power interruption without a hiccup. In Category B, it is 50 ms. In Categories D and Z, it's a full 1 s. When the size and weight of the equipment are limited, a 200-ms interrupt is a tall order, to say nothing about the reduced reliability electrolytic capacitors present. The common solution to this dilemma in systems with two independent power buses is to OR the supplies and provide just 50-ms power interruption storage capacity. (Refer to my 2012 article, "Diode ORing," *Circuit Cellar* 263.)

Surviving a power supply surge voltage is another tough requirement. Category Z calls for 80 VDC to be applied to the power input for 100 ms, followed by 48 VDC for 1 s. Category D calls for 425 and 345 VDC, respectively! Considering the power the clamps (e.g., transzorbs) must dissipate, a preregulator is usually needed for the equipment to survive this condition.

Another issue is the survival of a voltage spike without failure or degradation of performance. Category A must survive a 600-V, 2- $\mu$ s spike. In addition, the equipment must not be susceptible to an audio frequency appearing on the power lines. This is a conducted interference, which can be caused, for instance, by common connections with a load. It is a ripple of varying frequency and magnitude to which a system must be immune. A 14-VDC system must not be affected by 2  $V_{RMS}$  1 to 15 kHz ripple superimposed on its power lines.

Induced signal susceptibility defines the levels of interference induced in interconnect circuits. This applies to interference caused by AC power—which is 50 or 60 Hz in most applications—but can range between 350 to 650 Hz on aircraft with 400-Hz<sub>NOM</sub> power. It also applies to transients caused by switching inductive loads.

## SUSCEPTIBILITY OR IMMUNITY

The effects of the external interference on the equipment are called either susceptibility or immunity. The two terms are reciprocal; both are used. We strive for the lowest susceptibility, which means the highest immunity. Susceptibility/immunity can be either radiated (i.e., in response to electromagnetic (EM) fields) or conducted, meaning there is a conductive path through which the interference can get inside the equipment. There are numerous categories for various types of environment and use,

Frequency [MHz]	Category Levels [mA]						
	M	O	R	S	T	W	Y
0.01	0.6	3	0.6	0.03	0.15	3	6
0.5	30	150	30	1.5	7.5	150	300
1	70	250	30	1.5	7.5	150	300
30	70	250	30	1.5	7.5	150	300
40	*	*	30	1.5	7.5	150	300
100	*	*	*	*	*	*	300
400	32	50	3	0.15	0.75	32	100

TABLE 1

Conducted susceptibility test levels. (\*Values are interpolated.)

including high-intensity radiated fields (HIRF) immunity. **Table 1** shows conducted susceptibility for seven categories. The immunity is measured from 10 kHz to 400 MHz by maximum current in milliamperes induced into the conductors through a transformer, as the magnetic field is predominant at these frequencies.

Radiated susceptibility is determined by the maximum field strength in volts per meter (V/m) to which the device is immune. The e-field can be steady, modulated and/or pulsed. The immunity is typically measured from 100 MHz to 18 GHz. I've had specifications going up to 40 GHz. There are 10 categories, with the field strength ranging from as little as 1 to 7,200 V/m between 4 and 6 GHz.

Emission of RF energy is just as important as susceptibility. It can also be conducted

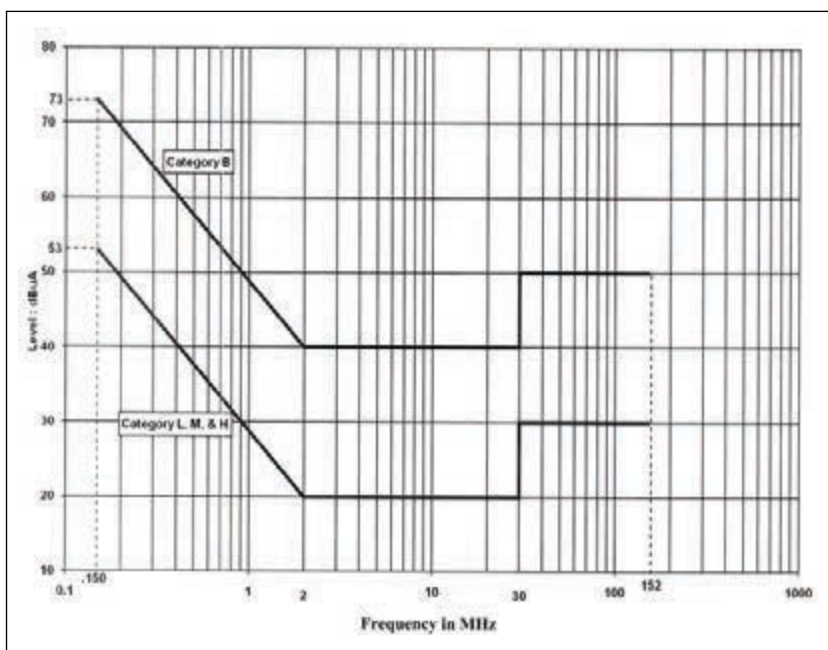


FIGURE 1

Conducted emissions levels versus frequency per DO-160 on power lines

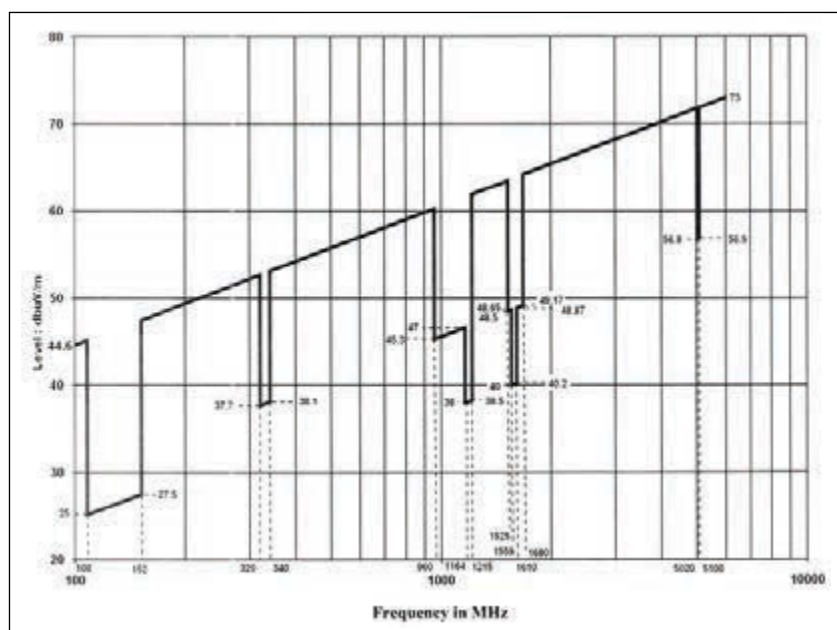


FIGURE 2

Radiated emissions levels for categories P and Q

and/or radiated. System designers, when specifying immunity and emissions for individual instruments in a system, make emission levels significantly lower than immunity to achieve safe margin within the system.

**Figure 1** shows conducted emissions levels for power lines per DO-160. The emissions are measured in decibels referenced to

microamperes (dBμA) of interference current on the external lines.

The levels are +20 dBμA higher for interconnecting lines (to sensors, loads, etc.) as they are not shared with other equipment. These categories are typical. For others, refer to the document, "DO-160G Environmental Conditions and Test Procedures for Airborne Equipment," RTCA, 2010.

Radiated interference is defined by the maximum field strength the equipment is allowed to radiate. It is expressed as dBμV/m, which is the decibel level referenced to 1-μV per meter e-field strength. **Figure 2** shows Category P and Q.<sup>[1]</sup> Notice the notches of reduced field strength. Their purpose is to prevent interference with aircraft communications equipment, satellite links, and other official equipment operating at those frequencies.


The next issue an EMC engineer must address is the susceptibility to lightning-induced transients, which are called indirect lightning effects. Direct effects apply to externally placed equipment, such as power and telephone lines or antennas. Indirect lightning immunity is verified by two types of tests: pin injection and bulk injection. Pin injection tests the circuits' damage tolerance. The device is not operational, while each and every pin of its connectors is zapped by pulses of different characteristics and strength. The device must survive all these tests.

**Table 2** shows pin injection tests performed with three different waveforms in five different susceptibility levels, level 5 being the strongest.  $V_{OC}$  stands for the pulse generator's open circuit voltage.  $I_{SC}$  stands for short circuit current. Notice that a Level 5 circuit must be able to survive 3,200 V and up to 1,600 A without damage.

Bulk injection tests levels ensure there is no functional upset when the test currents are injected through transformers. These are induction coils with device's harnesses threaded through them. Single and multiple strokes, and bursts are applied while the device must continue to work.

Finally, the device must survive electrostatic discharge (ESD) without damage or functional upset. Many microelectronic components today contain a rudimentary ESD protection that might be sufficient for consumer use. Equipment with extensive EMI, HIRF and lightning protection (such as discussed here) is inherently protected against ESD, although this needs to be verified by test.

## EMC IMPLEMENTATION

I trust you now have a good understanding of EMC requirements. In the last installment of this article series, I'll cover the practical implementation of EMC. 

Level	Waveform		
	3/3 $V_{OC}/I_{SC}$	4/1 $V_{OC}/I_{SC}$	5A/5A $V_{OC}/I_{SC}$
1	100/4	50/10	50/50
2	250/10	125/25	125/125
3	600/24	300/60	300/300
4	1,500/60	750/150	750/750
5	3,200/128	1,600/320	1,600/1,600

TABLE 2

Lightning pulse generator settings for damage tolerance test



circuitcellar.com/ccmaterials

## REFERENCE

[1] RTCA, "DO-160G Environmental Conditions and Test Procedures for Airborne Equipment," 2010.

## RESOURCE

K. Armstrong, Interference Technology Webinar, [www.interferencetechnology.com](http://www.interferencetechnology.com).

O. Hartal, Electromagnetic Compatibility by Design, R&BV Enterprises, 1993.



# The DIY Approach to ISE Project Management

Managing an FPGA project that involves several automatically generated (COREGen) modules can be a difficult task. This article shows how to use a simple Python script to allow you to easily target different hardware versions with the same source files, without needing to manually recreate or update the COREGen modules or other files.

*By Colin O'Flynn (Canada)*

Like most pieces of software, you probably use Xilinx ISE without giving too much thought to its internal project file format. But if you've tried keeping a Xilinx ISE Project File in a revision-control system (such as GIT), you were probably quickly forced to learn details of the file format. This is because if multiple people open and save the project, they might each cause small conflicting

changes in the project file. This is made even more difficult if you need to support CORE Generator (COREGen) modules, which have been generated through a graphical wizard (see Figure 1), and files may change with different versions of the tools.

This month I'm going to describe how I solved this problem for my own projects, and ended up with something I called



**FIGURE 1**

Xilinx provides the CORE Generator (COREGen) utility, which graphically configures a wide variety of blocks for your FPGA design. These blocks have been tested by Xilinx and are a great resource for the FPGA designer. But because they are generated through a GUI, they can be difficult to reliably port to different projects.

```

...
<version xil_pn:ise_version="14.6"
xil_pn:schema_version="2"/>

<files>
  <file xil_pn:name="interface.v" xil_pn:type="FILE_VERILOG">
    <association xil_pn:name="BehavioralSimulation"
xil_pn:seqID="1"/>
    <association xil_pn:name="Implementation"
xil_pn:seqID="3"/>
  </file>
</files>

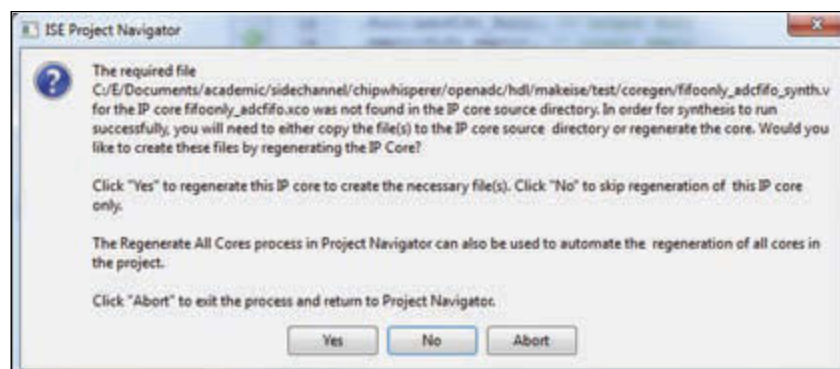
<properties>
  <property xil_pn:name="AES Initial Vector spartan6" xil_
pn:value="" xil_pn:valueState="default"/>
  <property xil_pn:name="AES Key (Hex String) spartan6" xil_
pn:value="" xil_pn:valueState="default"/>
  ...
  <property xil_pn:name="Device" xil_pn:value="xc6slx25"
xil_pn:valueState="non-default"/>
  <property xil_pn:name="Device Family" xil_
pn:value="Spartan6" xil_pn:valueState="non-default"/>
  <property xil_pn:name="Device Speed Grade/Select ABS
Minimum" xil_pn:value="-3" xil_pn:valueState="default"/>
  ...
  <property xil_pn:name="Package" xil_pn:value="ftg256"
xil_pn:valueState="non-default"/>
  ...
  <property xil_pn:name="Watchdog Timer Value spartan6" xil_
pn:value="0xFFFF" xil_pn:valueState="default"/>
  <property xil_pn:name="Working Directory" xil_pn:value="."
xil_pn:valueState="non-default"/>
  <property xil_pn:name="Write Timing Constraints" xil_
pn:value="false" xil_pn:valueState="default"/>
</properties>

...

```

**LISTING 1**

This shows a snippet of some areas of the Xilinx ISE Project File, which has the .xise extension. The file is XML-based, and this simple example has a single source code file.

**FIGURE 2**

If the project using a COREGen file only contains an XCO file, the system will offer to automatically generate the other required files. But it's sufficient to completely define the module using just the XCO file.

MakeISE. Rather than attempting to keep the auto generated files in the GIT repository, I instead keep a simple file that is able to generate all the required project files for the Xilinx ISE tools. Once you have the project files locally, ISE can make all the changes it requires without having to worry about these changes conflicting with other users.

In addition, this scripting method makes supporting the CORE Generator modules more reliable. If you want to port a design to a new device (e.g., moving from Spartan 6 LX9 to LX16), you need to recreate the COREGen modules using the graphical wizard. This requires all sorts of manual adjustment to settings (What type of FIFO? What flags to enable?), and it's easy to forget one of the flags when regenerating the project. Instead, the system I'll demonstrate here stores all those settings, for all the COREGen modules, in a single file. When it's time to port to a different device, you can be confident your new modules have all the proper settings.

It's worth noting that this isn't the first attempt to solve such a problem. There is a larger project called HDLMake which is part of the Open Hardware Repository, which can also do things like run the synthesis or simulation commands without requiring the GUI. For my projects I wanted something simpler—something I could easily include as part of my project, using only a single Python file. With that goal in mind, let's begin to look at the file formats in use.

**ISE FILE FORMATS**

There are effectively two separate file types of interest to us, and I'll briefly outline what those files contain. The first is the Xilinx ISE project file, which has the .xise extension. Note there is also a file with the .gise extension, but that will be automatically generated by ISE. We can completely define the project with just a XISE file, so the tool only needs to generate that file.

The XISE file is XML-based, making modification fairly easy. Opening an example file shows up that there are three main sections. Examples are shown in **Listing 1**. First, we have the version number of the ISE tool. We can generally set this to something "older" if required (i.e., set to version 14.2), and later versions of the tools will open the file no problem. Next, we will have a listing of all the project files. This can include our constraint file (UCF), source files (Verilog/VHDL), and COREGen files (XCO). Finally, we have a list of all the properties. This can include device and package, options about synthesis or implementation, command-line options for the place and route tools, and

everything else you can configure through the GUI.

The next file of interest is the COREGen module, which is defined by a file with the .xco extension. Based on this file the tools will auto-generate a number of other files, but we can completely define our core using a single XCO file. The first time it is opened ISE will offer to automatically generate the missing files, which we happily accept (see **Figure 2**). The format of the XCO file is shown in **Listing 2**, where we can see a simple file format to set various parameters.

## INTRODUCING MakeISE

MakeISE uses a template file as a reference, which is then modified with specifics to your project. This template is simply an empty project generated by ISE. We modify this file to adjust properties (such as the FPGA device), along with adding our source files. Similarly to generate COREGen files, we will first generate an example XCO file by configuring the core through the GUI. This configuration will be used as the base, and modifications (such as changes in the part number) will be made to this file. The MakeISE flow is shown in **Figure 3**, where you can see the template files are part of the input to MakeISE.

The MakeISE program is a single Python file. It does not depend on anything besides the default modules present in Python 2.x, making it easy to deploy to end users, as almost any Python install will allow them to run the file.

MakeISE is called with one or two arguments: at minimum it requires the name of the MakeISE Project file (normally using the extension .in although you can use any extension), and optionally the name of a XISE file to write, although by default uses the same root filename as the input file. An example input file is shown in **Listing 3**, and I'll discuss in more detail each of the sections next.

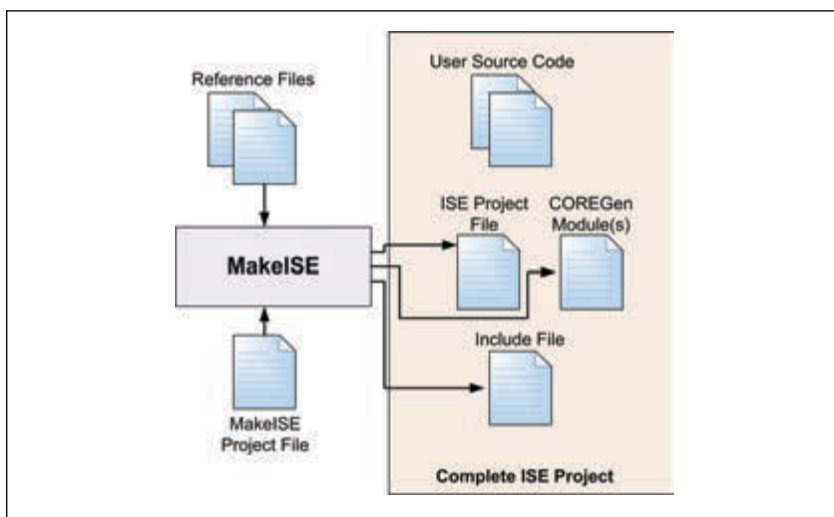
## SECTION OVERVIEWS

The first section, titled [ISE Configuration], is directly used to modify properties in the XISE file. The template file is given by the InputFile line, and the remaining lines are used to change properties from the template. This section contains information about the target device (family, part number, package, and speed grade) that will be automatically be remembered and used in writing the COREGen XCO files in addition to the XISE file. If you need to change any project options from their default, you can do this here: the example shows changing the include directory along with adding a command-line option to the map process.

```
...
SET createndf = false
SET designentry = Verilog
SET device = xc6slx25
SET devicefamily = spartan6
SET flowvendor = Other
SET formalverification = false
SET foundationsym = false
SET implementationfiletype = Ngc
SET package = ftg256
...
CSET clock_enable_type=Slave_Interface_Clock_Enable
CSET clock_type_axi=Common_Clock
CSET component_name=fifoonly_adcifo
CSET data_count=false
CSET data_count_width=13
CSET disable_timing_violations=false
CSET disable_timing_violations_axi=false
CSET dout_reset_value=0
CSET empty_threshold_assert_value=4
CSET empty_threshold_assert_value_axis=1022
...
```

**LISTING 2**

This shows a snippet of the .xco file, which defines a COREGen module.



**FIGURE 3**

The MakeISE project uses a number of templates as a reference, which avoids it needing to understand the exact file format. It also allows you to use existing working files as templates, and it simply modifies the target device or other parameters. The resulting project file references your existing source code (Verilog/VHDL), and the automatically generated include file makes supporting multiple hardware targets easier.



```
[ISE Configuration]
InputFile = ise_verilog_template.xise.in
Version = 14.4
Device Family = Spartan6
Package = ftg256
Device = xc6slx25
Speed Grade = -3
Verilog Include Directories = ../../../../hdl|../../
refproject
Other Map Command Line Options = -convert_bram8

[UCF Files]
system.ucf

[Verilog Files]
#Can have comments too anywhere
simpletop.v
simplemodule.v
setup.v = Setup File

[CoreGen Files]
fifonly_adcfifo.xco = ADC FIFO CoreGen Setup

[ADC FIFO CoreGen Setup]
InputFile = fifonly_adcfifo.xco.in
input_depth = 8192
output_depth = CALCULATE $input_depth$ / 4
full_threshold_assert_value = CALCULATE $input_depth$
- 2
full_threshold_negate_value = CALCULATE $input_depth$
- 1
write_data_count_width = 16
read_data_count_width = 16
data_count_width = 16

[Setup File]
BOARD_REV2
UART_CLK = 40000000
UART_BAUD = 512000
```

**LISTING 2**

This shows a snippet of the .xco file, which defines a COREGen module.

```
//AUTOMATICALLY GENERATED - MAY BE OVERWRITTEN
`define BOARD_REV2
`define UART_CLK 40000000
`define UART_BAUD 512000
```

**LISTING 4**

This shows the contents of the setup.v file that was automatically generated, and allows you to easily use the same source code for multiple hardware versions. The only file that needs to change between hardware versions is the MakeISE project file.

The next two sections are source files which are added to the project. The section titled [UCF Files] adds the UCF constraint file to the project. Note the script doesn't verify the location of files or copy them. It simply generates a project file with the given filename included as a source. You are responsible for ensuring the UCF file is located in the correct location, which could be the same location as the .in file, or some subdirectory.

The next section titled [Verilog Files] can be a simple list of Verilog source code files, but it can also include auto-generated files. In the example in **Listing 3**, the file setup.v actually doesn't exist, but is automatically generated by the MakeISE script. The generation of this file will be described in a later section.

Finally we get to the [CoreGen Files] section, which is one of the more powerful aspects of this script. If you have existing COREGen files, you can add them to the project, just like the Verilog files. But the more interesting aspect is the ability to automatically generate COREGen files based on a template, which is used in the example from Listing 3. Here we have a new section named [ADC FIFO CoreGen Setup] which will be used for generation of the COREGen file fifonly\_adcfifo.xco.

The section [ADC FIFO CoreGen Setup] again uses an input template file, which was a dual-clock FIFO core with different input and output widths, generated using the CORE Generator wizard. This allows me to take advantage of the COREGen wizard in ensuring the appropriate options were correctly configured, and I only need to specify any deviations from the original file. The device part number is automatically changed based on settings in the [ISE Configuration] section—meaning, I don't need to worry about if I change from a Spartan 6 LX9 to a LX16. The COREGen file will automatically be updated to reflect the correct device and package.

Various options can also be configured. In this case the depth of the FIFO is configurable from the project file. When changing from a larger (LX25) to a smaller (LX9) device, I might need to adjust the size of the FIFO. Note that you can use a \$CALCULATE\$ directive to make the parameter settings automatically dependent on each other. In this example, because my FIFO input width is 8 bytes, and my output FIFO width is 32 bytes, the output depth is four times less than the input depth. You must verify that any properties which should be linked have been. It's wise to try changing a property you'll be overwriting using the wizard, and

seeing which properties in the resulting XCO file change. In this example, beyond just the output depth being linked, the thresholds for the full flags also change.

Finally, we come back to the automatically generated Verilog file. In this case the section called [Setup File] will be used for automatically generating a file called setup.v. This file simply has a number of defines, the resulting file is shown in **Listing 4**. The idea of this file is it can be used to set parameters that might change between various hardware versions of your design, so you can simply use setup.v as an include in your source files. Using standard `ifdef...endif` sections allows you to enable or disable certain modules based on the defines.

## AUTOMATIC JOY


Supporting multiple hardware targets shouldn't be a major hassle, as one of the nice aspects of programmable logic is the ability to fine-tune the design for larger or smaller implementation sizes. Your high-end product version might contain a larger (and thus more expensive) FPGA compared to another configuration. But you shouldn't have to manually maintain all the project

## ABOUT THE AUTHOR

Colin O'Flynn (coflynn@newae.com) has been building and breaking electronic devices for many years. He is currently completing a PhD at Dalhousie University in Halifax, NS, Canada. His most recent work focuses on embedded security, but he still enjoys everything from FPGA development to hand-soldering prototype circuits. Some of his work is posted on his website at [www.colinoflynn.com](http://www.colinoflynn.com).



files and COREGen modules between the two versions. That's just asking for trouble!

Hopefully, this article gave you enough of a teaser that it encouraged you to try out MakeISE yourself, and save yourself some headaches of manually recreating your ISE project files. As usual, I linked some additional video examples on ProgrammableLogicInPractice.com. For full details, you can check out the MakeISE project at <https://github.com/colinoflynn/makeise>. 

# Sign up for the FREE Circuit Cellar Newsletter!

You'll receive electrical engineering tips, interesting electronics projects, embedded systems industry news, and exclusive product deals via e-mail to your inbox on a regular basis. If you're looking for essential electrical engineering-related information, we've got you covered: microcontroller-based projects, embedded development, programmable logic, wireless communications, robotics, analog techniques, embedded programming, and more!

Subscribe now to stay up-to-date with our latest content, news, and offers!

**circuitcellar.com**



## EMBEDDED IN THIN SLICES

# Estimating Your Embedded Systems Project (Part 2)

## Challenges Unique to Embedded Software Development

COLUMNS

This month Bob continues his series on estimating the costs for designing and developing your embedded systems project. He covers the issues unique to estimating an embedded systems project.

*By Bob Japenga (US)*

**R**ecently, we were asked to estimate the cost to develop a system that would interface with every device of a particular type manufactured in a particular country. Our job was to design a system to extract data from these devices. There are 10 international standards applicable to these devices in this country. These standards define the protocol for accessing this data. The data is available on one of three possible hardware data busses. Some of the data available on these busses is in the public domain and some is only available from the manufacturer. There are over 150 different types of these devices sold each year in this country. Each year, another 150 new or similar types are sold. The specification is perfectly clear. By the way, can you have the estimate to me by Monday? Hmmm! I saw an advertisement the other day that said a company's one-day seminar would teach me to accurately estimate firmware schedules. Maybe if I took that class on Friday, I could get the accurate schedule by the end of business on Monday.

How can one estimate something like this? Here is axiom number one: Don't believe anyone who tells you he can teach you to accurately estimate your firmware schedule in a one-day seminar. Or in a one week seminar. Or even after 10 years of doing it every day. Accurate? No! But we can get better. And the best way I know how is by first defining the problems. That was the focus of the first article in our series.

Last time, we looked at the general problem of estimating software development costs. This month we will look at the challenges that are unique to embedded software development. Certainly there are things that make embedded software more challenging to develop than other types of software. But what makes embedded software that much harder to estimate?

### BIGGER SURFACE AREA

Recently, I reviewed last month's article with our team and asked the question: Why is estimating embedded systems more difficult



than estimating other kinds of software? One engineer said, "The surface area is much bigger." What he was saying is that all of the standard problems with estimating just got multiplied. Let's just review what we said last time and see how some of these issues are more complicated for embedded systems.

## UNCLEAR REQUIREMENTS

The accuracy of our software estimates can only be as good as our understanding of the requirements. This difficulty is multiplied with embedded systems because of the complexity of the interfaces. In addition, there are a lot of requirements that only become clear after you implement. The datasheet of a small microprocessor we use on one project is 1,400 pages long. There are just a lot more requirements that can be unclear or misunderstood. We approved a rework once to one of our designs that required the manufacturer to add a wire to one end of a capacitor. After about a hundred were shipped, the capacitors started shorting (especially problematic for bypass capacitors). Buried in the capacitor's datasheet was the requirement to not touch the capacitor with a soldering iron. The rework needed to be performed with a hot air process. It was very clear on page 78 of the capacitor's datasheet!

The specifications can also be wrong. Many times errata come out after you have started your design. We once missed an errata in an 800-page microprocessor datasheet that said, "Oh, by the way, this device has a 256-MB address range but can only address 16 MB of NOR flash!"

## THAT ELUSIVE BUG

Embedded real-time systems and systems with concurrency make debugging much more difficult. That we can plan for. But those elusive bugs that take two weeks in non-embedded systems can take two months on embedded systems because your tools are not as powerful and the complexity of the design is that much greater.

## HIDDEN COMPLEXITY

The scale of complexity is greatly multiplied in embedded systems. We are supposed to write software that interfaces with other very complex devices. Take this simple requirement from a datasheet of

chip we interface with: "To reset the chip, hold RESET\_N low for 300–500 ms." On the surface that seems straightforward. But what is hidden and not written in the manual is that if the RESET\_N is held low for more than 1,000 ms, the chip powers down and will not start when the RESET\_N line is brought high. If for some reason your function that releases RESET\_N gets delayed, the chip would not become operational as you expected. This requirement of raising RESET\_N becomes a hard deadline that you might not expect to be as such. These kinds of hidden complexities are legion in embedded systems.

## PROGRAMMER EFFICIENCY

Two years ago, I sat with one of the best embedded designers I know. He was running out of real time on a project. The problems were so complex that it took two of us with a combined experience of 60 years of designing embedded systems to figure out what was going on and how to fix it. Where a less-efficient programmer might be four times less efficient than your best designer, in an embedded environment that same programmer might be 10 times less efficient.

## OPTIMISM & HUBRIS

A couple months ago, one of our customers asked us to add a splash screen and a progress bar to the start of a device. One of our best designers saw that u-boot had hooks for sending an image to

our display. Linux had a progress bar app (psplash) that worked with our display. (If you want to have an open-source progress bar for Linux (psplash), check out the Yocto project's distribution <http://git.yoctoproject.org/cgit/cgit.cgi/psplash/tree/>.) The system was built on a BeagleBone architecture so others must have done this before. The on-line community support for this architecture is huge. We knew we could get lots of help. In addition, we have done similar projects in about four days without this kind of support. We know how to do this. We can deliver this fully tested in four days. (The BeagleBone open source reference design is showing up in the designs of a number of companies. You can find more about it at <http://beagleboard.org/>.)

At the end of four days, we found that the hooks in u-boot didn't work. No one in the online community knew how to make them work. At the end of two weeks, we discovered

*Embedded real-time systems and systems with concurrency make debugging much more difficult. That we can plan for.*



### ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. With a combined embedded systems experience base of more than 200 years, they love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at [rjapenga@microtoolsinc.com](mailto:rjapenga@microtoolsinc.com).

that the u-boot image was inverted from what the Linux driver was expecting. At the end of four weeks, we discovered that the progress bar did not play well with this particular display. At the end of six weeks, we discovered that the customer did not provide us with the right code base to start with. We were optimistic. Embedded systems will amplify the negative effects of your optimism and hubris enough to put you out of business. (The u-boot open-source universal bootloader software has been at the start of every Linux project we have designed. You can find more about it at [www.denx.de/wiki/U-Boot/WebHome](http://www.denx.de/wiki/U-Boot/WebHome).)

### CUSTOMER SCHEDULE CREEP

Customer schedule creep is a specific instance of “The Other Guy” problem we talked about last time. But it has a unique feature to it. We are now six months behind schedule releasing a new version of embedded software for a product we designed for a customer. One of the driving factors in the delay is that the customer still doesn’t have their portion of their web server operational. Every day it slips, our team has to work on other things instead of completing the testing. Each day the team might spend a half hour coordinating with the customer. None of this 60 hours was estimated. The inefficiency of this schedule creep is even more costly. Fred Brooks in *The Mythical Man-Month* puts it this way: “Disaster is due to termites, not tornadoes.”

Some of this is common to non-embedded software. But embedded software by its very nature is embedded in stuff. And often, stuff that is being designed in parallel. As a minimum, it must talk with hardware that is often not completely designed. It may also talk with other machines that are being developed in parallel. How well those are designed and when they are delivered can be a multiplier in the schedule and cost of an embedded system.

### PARTNER QUALITY

Another instance of “The Other Guy” problem is with your partners. Some of the partners we interface with are the hardware we run on, the busses we communicate on, the networks we connect to, the other devices we talk to, the hardware designer who designed our board, the hardware layout team that laid out the PCB, and the hardware build team that actually built the board. How well they do their job has a direct bearing on how much it will cost you to develop your embedded system.

Let me share two examples. We have a supplier who builds our printed circuit boards and assembles them during our development stage. We love this supplier because their work is impeccable. Sometimes our customers require us to get the boards built someplace else or by them. Invariably, parts are put in backwards. Ball Grid Arrays (BGA) parts are not X-rayed to verify their connections. Flow soldering techniques cause modules to reflow and not re-center on their footprint. When we get the boards, it might take us two to three days more to debug and troubleshoot these problems because of the supplier. Remember that we are checking out a new design which can have flaws in it as well. How does one estimate for that extra two to three days? You don’t know the quality of that supplier until you have used them.

Another problem we have is with other hardware designers. When we design the boards, we know the quality factor of our designers. They may not be perfect, but they are a known quantity. We know by experience how long it will take to integrate the boards designed by our own people because we have metrics and experience. But what if you are designing embedded software that runs on a board that is designed by “the other guy?” Our experience shows that it can ruin a schedule in two ways. The first is the extra time it takes to “bring the board up” because there are more errors in the design than you are used to. This can easily add several weeks to a schedule. But often we find that it takes more turns of the board than it normally takes you to get an operational board. During that extra two to three weeks, your team is much less efficient. Do you assign them to a new project? That is not practical. So the software team becomes less efficient. They work on “cleaning up the code” and “doing some documentation.” Sounds good, but these are schedule killers. And for estimating, the problem is: how do you know the quality factor in advance?

### TESTING DIFFICULTIES


Embedded systems are much more difficult to test than conventional software

systems. That additional difficulty can be planned for and the estimate adjusted to take that into account. The problem comes when we don't think through these difficulties when we estimate the project. We developed a tiny embedded device that was implanted into a human body. This device communicated to the outside world via infrared. The device sent 8 bytes every millisecond. We accurately estimated the time it would take to design the hardware and the software necessary to accomplish these requirements. However, when it came to test it, we did not have a means to easily do that. There were no off-the-shelf tools to read the IrDA and provide an integrity check to it. How does one know that all 8,000 bytes are correct every second? A special test tool was needed to display and analyze that it was meeting its requirements. But special test tools take time and money to design. They can drastically expand the effort required to design and develop an embedded system.

Another thing that can affect our ability to estimate embedded system is the time delay inherent in many designs between making a change, testing the change and reprogramming the device. When the time

delay is very small (as in non-embedded systems), iterative designs can be created much more quickly. Where this impacts our estimates is that we often don't know what the time delay is and exactly how it will impact the schedule. For example, let's imagine that over the course of the project you make 1,200 changes to your software requiring a compile and load. If the compile and load time takes 70 s compared to 10 s, this can add three extra days to your project. Often, during the time we estimate, we don't know with that precision the compile and load time.

### FACE THE IMPOSSIBLE

The surface area of complexity in estimating embedded systems is many times more complex than designing non-embedded software. Knowing what some of the problems are can help us get better at this impossible task. Next time, we will look at how we can address these problems and get a little better. If you have some other suggestions about the problems in estimating embedded software systems and how you deal with them, drop me a line. This is a field in which I need constant improvement. And of course, I only improve in thin slices. 

## Circuit Cellar 2014 Digital Archive

With this digital subscription, you have access to all 12 issues of Circuit Cellar 2014 from any computer or tablet at anytime. Readers can explore project ideas, bookmark pages, and make annotations throughout each issue.

### Circuit Cellar 2014 CD

CD includes 12 issues of Circuit Cellar in PDF format along with related article code.

Order yours today

# cc-webshop.com





## THE DARKER SIDE

# Let's Count Errors

## An Introduction to BER Testers

COLUMNS

When working on applications that require high-speed digital transmissions, signal transmission errors can cause a variety of problems. As Robert explains, bit error rate (BER) is the ratio of errors to total transmitted bits. In this article, he explains how a BER tester will help you properly measure error rates.

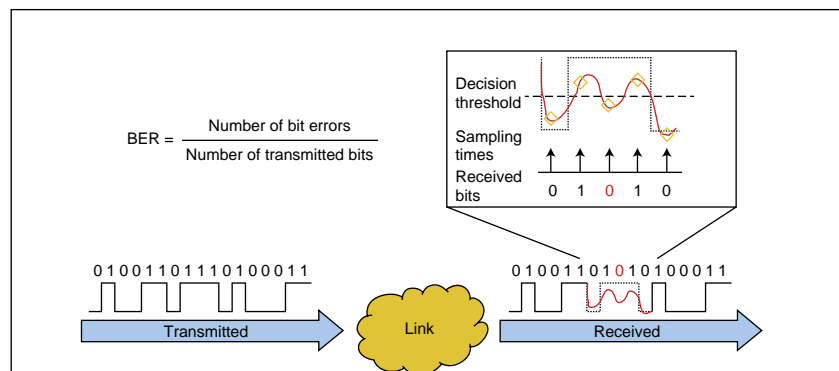
*By Robert Lacoste (France)*

**W**elcome back to the Darker Side column. I read that Thomas Jefferson once said, "Delay is preferable to error." This may be often true, but in telecommunications—or, more specifically, when it comes to signal transmission—this is far from obvious. For example, let's think about a digital audio

transmission (music or voice). In such a system, you can be sure that a wrong bit value from time to time will be less noticeable than a blank in the signal until the good bit value can be received. Why? Simply because the receiver (a human) has a quite high tolerance for errors. The same is true for video, but it is also the case for any data transmission, even between two machines as long as an error-correcting code is used to recover from transmission errors. The key here is that "some" transmission errors could be acceptable, as long as they are not too numerous. If not, they will jeopardize the transmission itself. Where do transmission errors come from? How can you measure them? This is my topic for this month.

### BIT ERROR RATES?

As you know, any information can be digitally encoded as a stream of bits. These bits then can be transmitted on the transmission channel, wired or wireless, through a succession of encoding and decoding steps. I already presented the



**FIGURE 1**

A receiver needs to compare a signal amplitude with a given threshold, which may lead to erroneous decisions in the presence of noise. A bit error rate (BER) is simply the number of errors divided by the total number of transmitted bits.

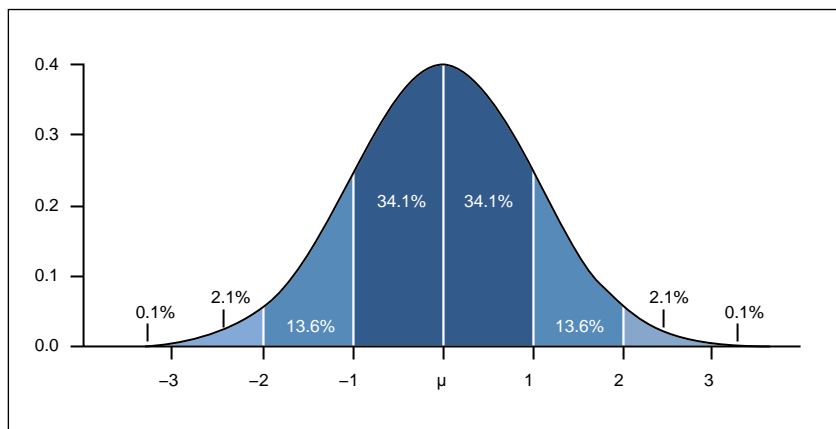
different ways to transform bits into so-called base-band signals in my 2011 article, "Line Coding Techniques" (*Circuit Cellar* 255). I also covered how to use modulation to put them on a carrier frequency in my 2009 article, "Digital Modulation Demystified" (*Circuit Cellar* 233).

This month, I won't detail a fancy encoding scheme. I will use the example of the simple nonreturn to zero (NRZ) base-band encoding. This is a very complicated way to say that the successive bits are transmitted as two voltages (e.g., 0 V for a "0" and 3.3 V for a "1") as with a simple UART. In such a system, how does the receiver recover the transmitted bit value? Of course, it will use a voltage comparator and will check if the line voltage is above or below a given threshold. This threshold is ideally the middle between 0 and 3.3 V (say, 1.65 V). Everything above is assumed to be a one; everything below is a zero. The measurement and decision are done once per bit at the bit-sampling time. This process generates a received bit stream, which is identical to the transmitted bit stream as long as there are no errors. Now, by definition, the bit error rate (BER) is simply the ratio of the number of errors divided by the total number of transmitted bits (see **Figure 1**).

## BIT ERRORS, WHY?

In my February 2010 article, "Living with Errors" (*Circuit Cellar* 235), I introduced the usual key contributor to bit errors—noise. Plenty of noise sources interfere with the transmission and degrade the clean 1s and 0s generated by the transmitter. This noise can be either human-generated (e.g., electrical motors, wireless transmissions, or sparks) or natural (e.g., high-energy particles, lightning, or ionospheric effects). All these noises will add to each other. Now the magical trick: If you add plenty of independent phenomena (and the key word here is "independent"), then you will always get a Gaussian distribution (see **Figure 2**). That's why Gaussian shapes are everywhere. Such a Gaussian distribution can be characterized by two values: its average (0 for a DC-centered noise) and its standard deviation, which is exactly the same as its round mean square (RMS) voltage.

An important fact about Gaussian distribution is that it is boundless. Gaussian noise with a 1-mV RMS voltage will have occasionally very high values, even if the associated probability is very low! As shown in **Figure 2**, a 1-mV RMS noise will have an instantaneous value higher than 3 mV only 0.1% of the time; but, from time to time, it will be far higher than that. In fact, for a "perfect" Gaussian noise, you will get a voltage as high

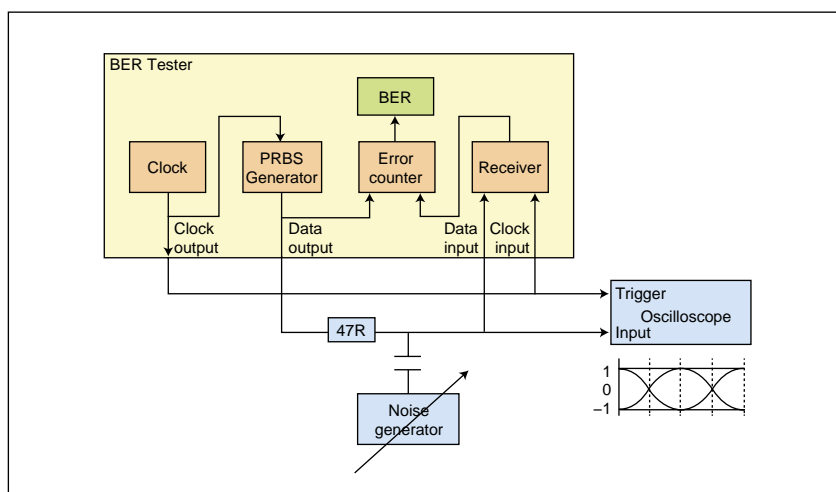


**FIGURE 2**

This figure shows you the ubiquitous Gaussian distribution. As you can see, 95.4% of the time the noise amplitude stays closer than two times the standard deviation around the average. But be careful because its amplitude is not bounded. (Source: Wikipedia, [http://en.wikipedia.org/wiki/Normal\\_distribution](http://en.wikipedia.org/wiki/Normal_distribution))

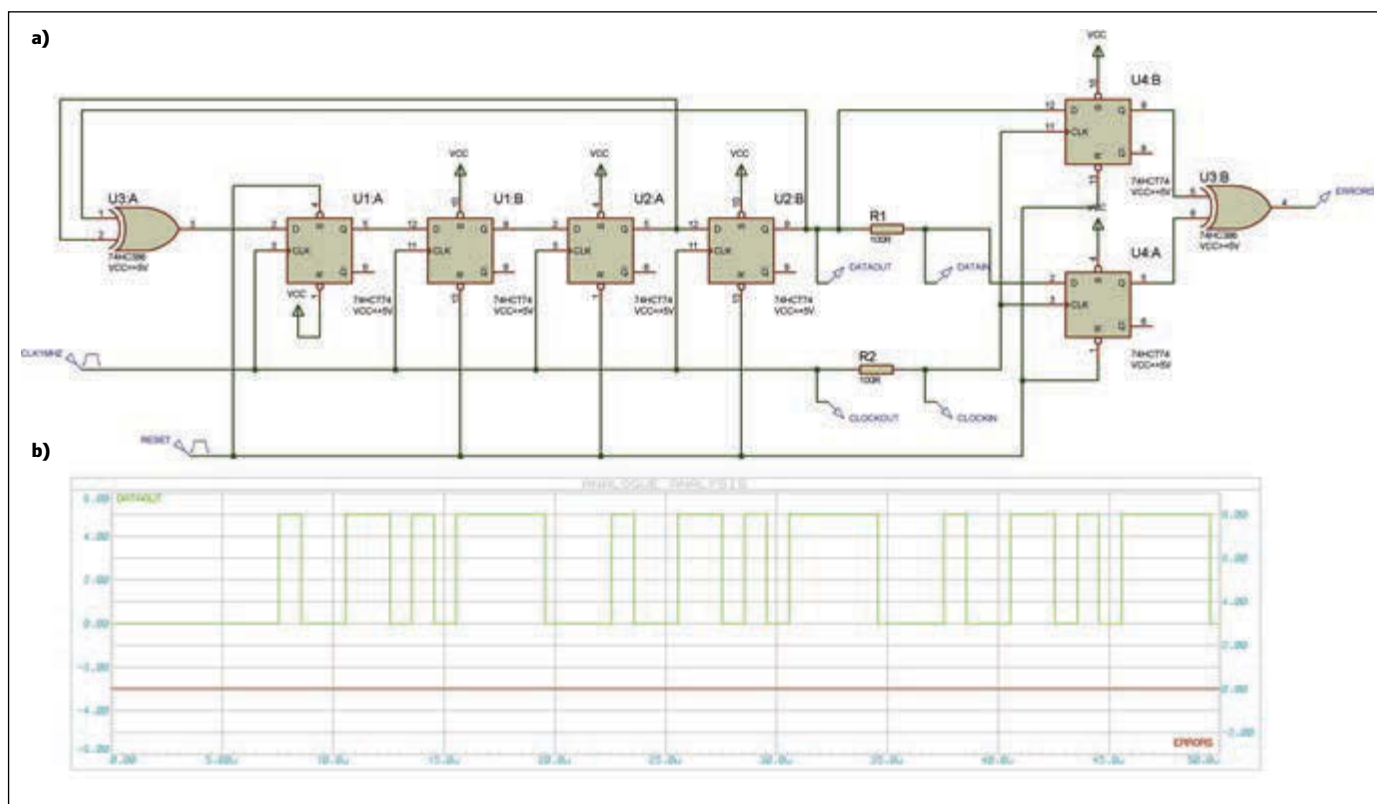
as you want if you wait enough time. That is exactly what boundless means. The story is a little different for real-life signals that can't go to thousands of volts for other reasons, but you've got the idea.

Let's go back to our NRZ transmission example. If such a Gaussian noise is added to the received signal (and you can be sure that it will), the noise value will be higher than half the voltage decision threshold from time to time. The comparator will be fooled and you will receive a wrong bit value. This explains why any transmission link is prone to bit errors. Of course, you can design a system for a BER as low as you want, just by increasing the signal over noise ratio, but you will never be sure that the BER will be zero. That's why checksums are useful. They enable you to



**FIGURE 3**

Here my experiment's setup. This figure also illustrates the internal blocks of a typical BER tester.

**FIGURE 4**

This is a simplified schematic of a BER tester, with a PRBS generator on the left and an error comparator on the right. The plot shows you the simulated output signal.

detect remaining errors, but that's another story.

Gaussian noise is not the only source of bit errors, but it is usually one of the predominant contributors. There are numerous other sources of error, such as inter-symbol interference (i.e., increased error rate on a bit depending on previous bit values) and timing jitter. But for my purposes here, Gaussian noise will be enough for the topic I want to cover next: how to measure a bit error rate.

## BER TESTERS

Specific test equipment is available for measuring bit error rates. The concept of a

BER tester is very simple (see **Figure 3**). First, it needs to generate a bit stream to test the transmission link. This bit stream must be as random as possible in order to detect potential errors linked to specific bit sequences. This is exactly what so-called pseudo-random binary sequence (PRBS) generators are made for. (More on that in a minute.) This PRBS generator is clocked by a simple square wave oscillator set at the desired bit rate. The output of the PRBS generator, as well as the clock if needed, is then connected to the transmission channel to be tested.

At the other end of this transmission channel, the BER tester needs to include an associated receiver. In its simplest form, this receiver is a voltage comparator with a user set threshold and a register to latch the comparator output value at each clock front.

Lastly, a BER tester includes an error detector, which is nothing more than a digital comparator between the sent and received bits and a counter. Add some software and you have a full-featured bit error rate tester (BERT).

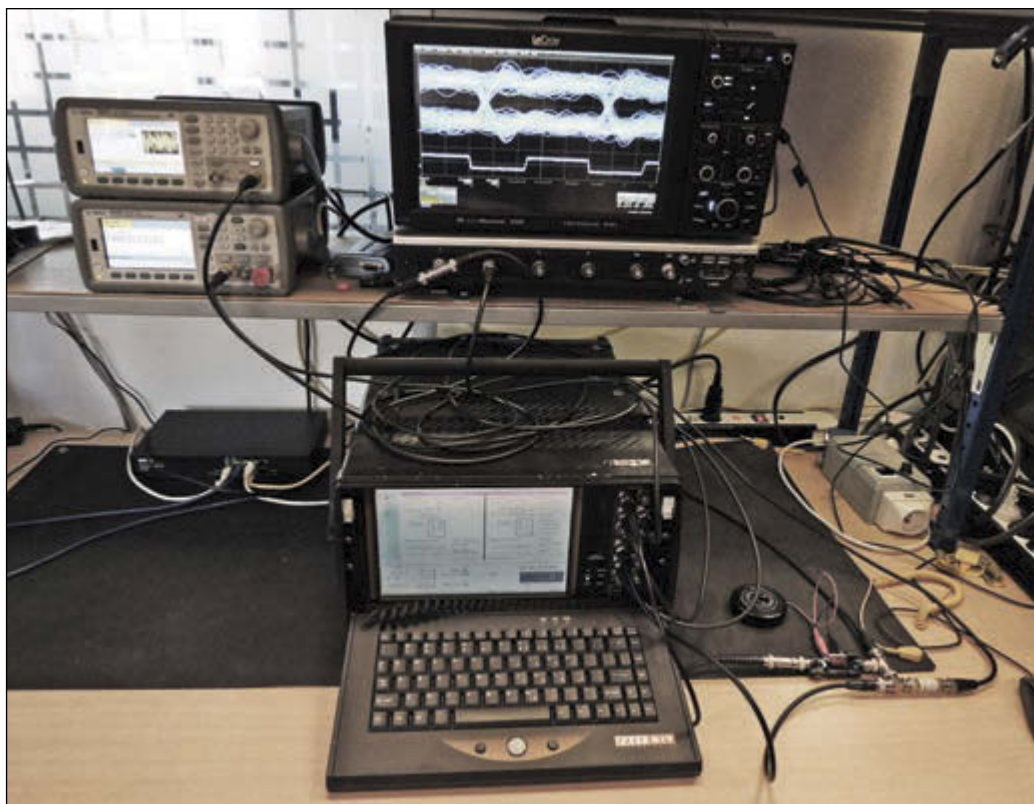
A BERT is a nice piece of test equipment, but I assume you don't have one in your lab. BERTs cost a fortune and are rarely used. But maybe you'd like to build one? This is actually easy as long as you stay with reasonable clock rates (say, a few tens of megahertz). Although I won't cover how to build a BERT in this article, I did create a basic schematic

## ABOUT THE AUTHOR

Robert Lacoste lives in France, near Paris. He has 25 years of experience in embedded systems, analog designs, and wireless telecommunications. A prize winner in more than 15 international design contests, in 2003 he started his consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. His book (*Robert Lacoste's The Darker Side*) was published by Elsevier/Newnes in 2009. You can reach him at [rlacoste@alciom.com](mailto:rlacoste@alciom.com). Don't forget to put "darker side" in the subject line to bypass spam filters.





**PHOTO 1**

This is my experimental setup. The BER tester is lying on the table, with noise generator and frequency meter on the left and the digital oscilloscope on the top.

for you (see **Figure 4**). Warning: This is only a starting point. I've never actually built the BERT and plenty of things are missing. But I hope you'll get the idea. The leftmost section—from U3:A to U2:B—is an example of a four-step PRBS generator. Such a generator provides a pseudo-random sequence of 16 bits ( $2^4$ ), which then repeats indefinitely. I used my Labcenter Electronics Proteus simulator to generate the output in Figure 3. A simple web search for "PRBS" will bring up find variants with any number of steps. An excellent reference is Clive Maxfield's 2006 *EETimes* article, "Tutorial: Linear Feedback Shift Registers (LFSRs)."

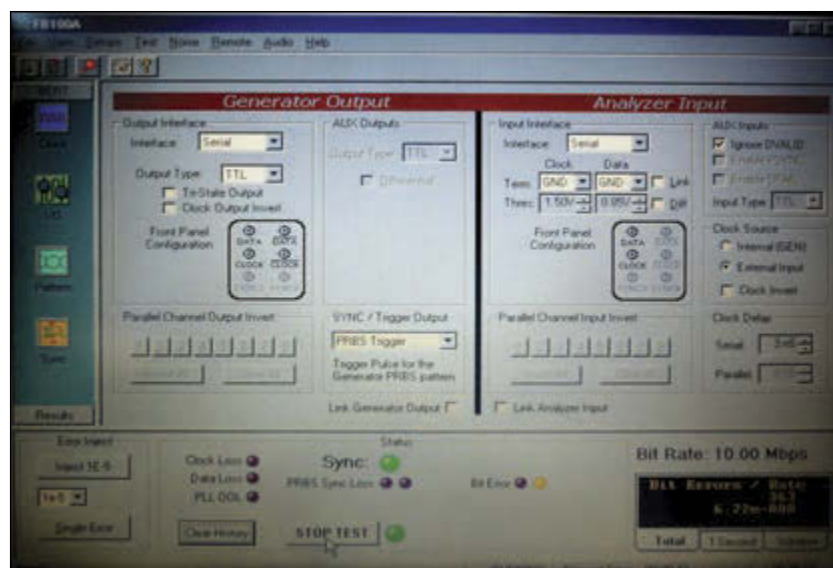
In this example, I used four D registers in the form of old 74HCT74 chips to design this PRBS generator, but FPGAs aficionados will enjoy playing around here. The resistor R1 on my schematic is where the actual transmission channel should be inserted. It is followed by a basic comparator built around two more D registers and an exclusive OR gate. Just add an error counter and a clock generator and you'll have a basic BERT.

## TO THE BENCH

OK, now I must confess that I'm lucky enough to have two beautiful BER testers in my company's lab, and that's why I didn't have to build one by myself! For these tests, I used an Aeroflex Fastbit FB100A, configurable from 100 bps up to 50 Mbps. It is running

under Windows NT, so you will conclude that this is not actually a very recent system, but it is working flawlessly. I bought it online for about \$500, which is probably around 1% of its original price tag. It was a nice deal, even with the hefty shipping cost.

I wanted to show you some actual bit errors, so I switched on this BER tester, and connected its transmitter output to its receiver input through a 47- $\Omega$  resistor. I

**PHOTO 2**

This is a screenshot of the BER tester's main window. Notice the calculated BER value on the bottom right.

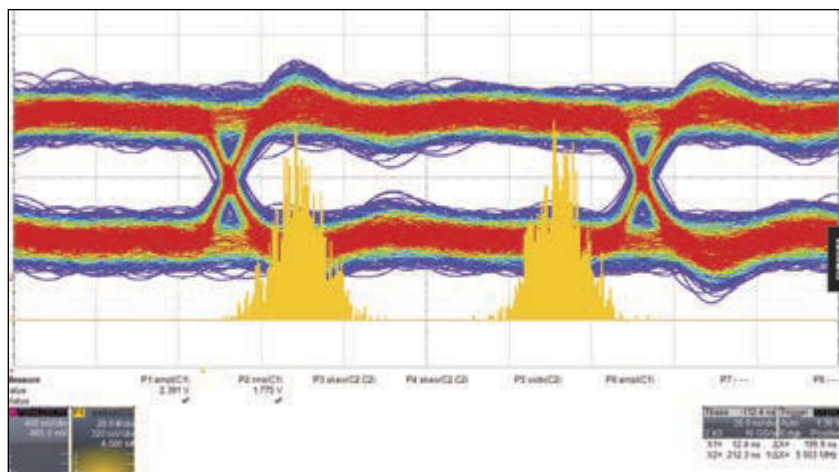


FIGURE 5

With a low noise voltage—190-mV RMS in this case—the plot called an eye diagram stays open. The histogram confirms that the voltage values for bit “0” and “1” are far enough from each other.

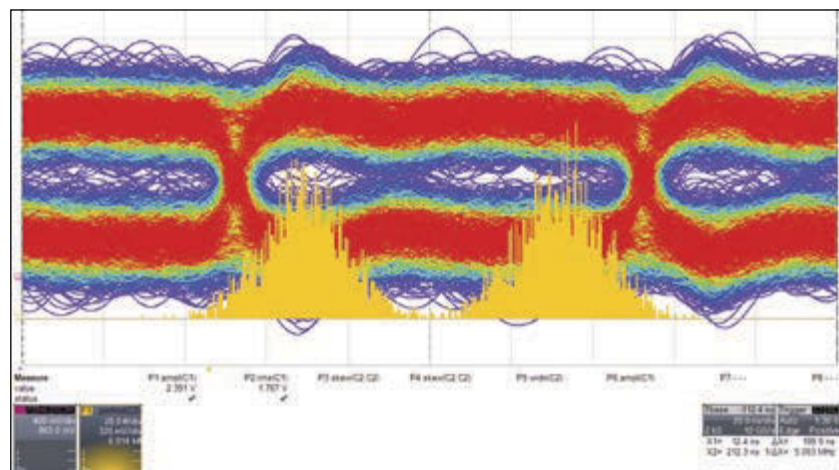


FIGURE 6

The situation is drastically different with a noise increased to 300 mV, giving a far higher bit error rate.



circuitcellar.com/ccmaterials

C. Maxfield, “Tutorial: Linear Feedback Shift Registers (LFSRs) - Part 1,” EETimes.com, 2006, [www.eetimes.com/document.asp?doc\\_id=1274550](http://www.eetimes.com/document.asp?doc_id=1274550).

#### SOURCES

**Fastbit FB100A Error rate tester**  
Aeroflex | [aeroflex.com](http://aeroflex.com)

**Agilent 33521A arbitrary signal generator (discontinued)**  
[www.keysight.com](http://www.keysight.com)

**Proteus CAD tool suite and simulator**  
Labcenter | [www.labcenter.co.uk](http://www.labcenter.co.uk)

**Waverunner 610ZI oscilloscope**  
Teledyne LeCroy | [www.teledynelecroy.com](http://www.teledynelecroy.com)

#### REFERENCES

C. Bianchi and A. Meloni, “Natural and Man-Made Terrestrial Electromagnetic Noise: An Outlook,” *Annals of Geophysics*, Vol. 50, No. 3, 2007, [www.earth-prints.org/bitstream/2122/3674/1/11bianchi.pdf](http://www.earth-prints.org/bitstream/2122/3674/1/11bianchi.pdf).

then took an Agilent 33521A arbitrary signal generator, configured it as a pseudo-Gaussian noise source, and added it to the receiver input through a DC-coupling capacitor. Lastly, I connected my Teledyne LeCroy Waverunner 610ZI oscilloscope to the signal to show you the actual signal waveform. The scope must be triggered by the clock signal in order to get a stable picture. The full setup is shown in **Photo 1**. Nice, isn't it?

A BER tester has usually plenty of configuration options. **Photo 2** shows you the main instrument display. It is logically split into two sections: transmitter and receiver. I manually set the transmitter to 10 Mbps and the output amplitude to 3.3 V, and measured the received signal amplitude on the oscilloscope. I got a signal varying from 500 mV to 1.4 V (which is a 900-mV peak-to-peak signal with a 0.95-V DC offset), as it is heavily damped by the 47-Ω resistor and the impedance of the noise generator. This roughly reduced the signal amplitude by half but allowed to inject easily some noise for my tests. So, I set accordingly the threshold of the receiver to 0.95 V, roughly in the middle of the signal amplitude.

My first test was with the noise generator set to an amplitude of 160-mV RMS. This noise voltage is quite low as compared to the 900-mV amplitude of the signal so I expected no bit errors. I simply clicked on RUN TEST and left the test run for a while. Ten minutes and a coffee later, the calculated bit error rate was displayed by the analyzer:  $6.22 \times 10^{-8}$ . That's very low, but not null! In fact the analyzer told me that 5,836,012,884 bits were received during these 10 min (which is close to the expected count with a bit rate of 10 Mbps) and 363 of them were wrong. That shows you the strength of such a test: even errors that would be barely noticeable during a functional test are quickly highlighted by a BER tester. This could help you to quickly compare different design choices, and to select the best one.

It is also interesting to see what happens in the time domain. Have a look on **Figure 5** and **Figure 6**. Both are screen copies of the oscilloscope, with respectively a noise of 190-mV RMS and 300-mV RMS. Some explanations are needed to understand these plots: I configured the Teledyne LeCroy Waverunner oscilloscope in a special persistence mode, which display different colors depending on the recurrence of a signal. The hotter the color, the more frequently the signal had this voltage value. This is a wonderful tool to feel the statistical properties of a moving signal. Here you see that the signal is still quite clean with 190 mV of noise, it doesn't come too close to the middle of the screen (see **Figure 5**). In such a case telecommunication guys say that

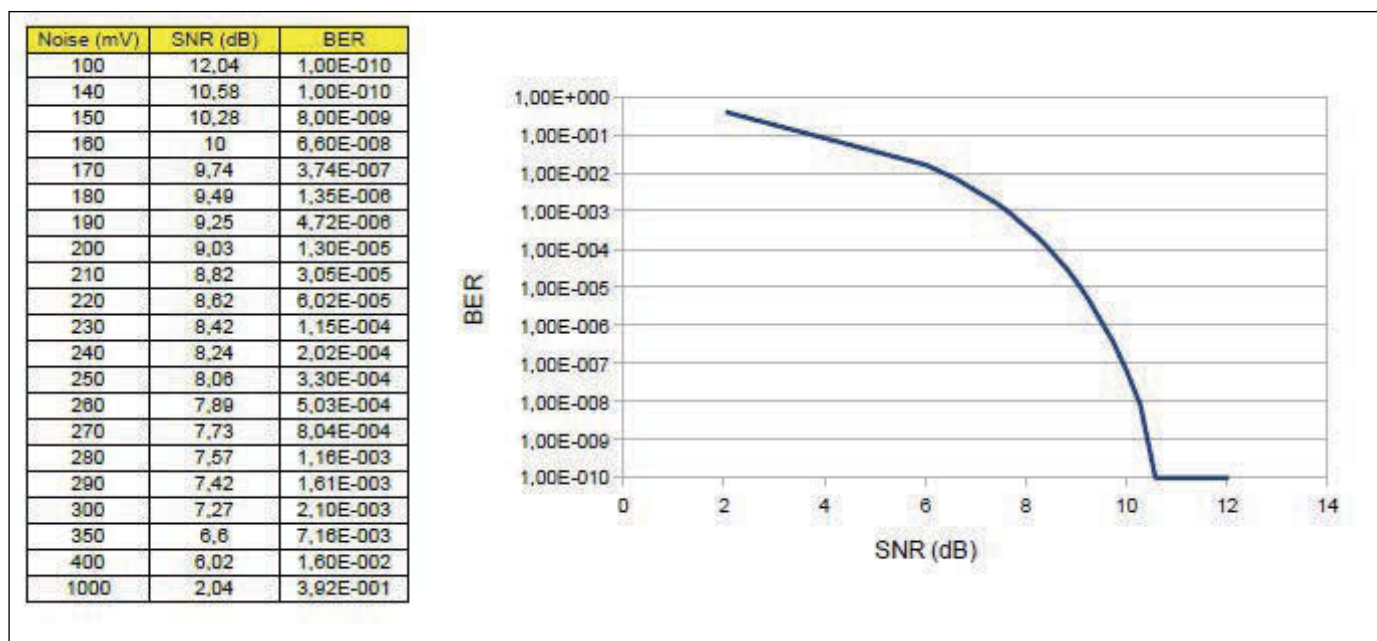


FIGURE 7

This plot of BER versus noise level, expressed as a signal-over-noise ratio, is very typical. Take care as the scale of both axes is logarithmic.

the eye diagram (which is the name of such a plot) is open. Conversely the eye is closed when I increase the noise to 300-mV RMS (see **Figure 6**), meaning that the BER must be far higher. And the measurement with the BER tester confirmed these assumption: between these two tests the BER climbed from  $4.7 \times 10^{-6}$  to  $2.1 \times 10^{-3}$ , which is 500 times higher!


Do you want to see another interesting feature of a high end digital scope? They could calculate and plot for you histograms of the signal. Think of such an histogram as a projection of the signal's amplitude on the vertical axis. The more the spot stays at a given voltage, the higher the corresponding histogram value will be. The result is also represented in Figure 5 and Figure 6 as a superposed yellow chart. As expected, the histogram shows more or less two Gaussian shapes. These two Gaussian shapes correspond to the two stable signal levels with the addition of the Gaussian noise. When the noise is low (see Figure 5), the two Gaussian's are far from each other, implying low BER. Conversely, Figure 6 shows that the two distributions are starting to merge, which is another way to say that the eye is closed and the BER is high.

Before dismounting the experiment and putting back the BER tester on the shelf, I measured the BER for different noise levels. The resulting plot is shown on **Figure 7**. This is a very typical BER curve. The scale may be varying, but you will find the same overall

shape in plenty of applications ranging from wireless receivers to Ethernet transceivers or telecommunication systems.

## WRAPPING UP

OK, that's the end of our journey in the world of BER testers. I know it's unlikely that you'll ever have a commercial tester at your workbench. However, I am also sure that you're projects will occasionally suffer from errors in signal transmissions. In some cases, knowing what's going on and how to measure a BER can make the difference. Just imagine two colleagues working on a data transmission problem. One of them is trying to find out what's happening with its debugger, scope, and multimeter, while the other knows about BER testers and decides to rent one for a couple of days. Who do you think will solve the problem sooner, get a big thanks from the customer, and receive a bonus from the manager?

Don't get me wrong. I'm not saying that oscilloscopes can't help, especially because eye diagrams and advanced scope features are invaluable. But knowing about dedicated testing equipment and how it can provides insights about data transmission can't hurt! Moreover, the increase in bit rates and system complexity everywhere is going to make these topics hotter than never. Running a fast USB or gigabit Ethernet connection without too many errors is significantly more difficult than connecting a 9,600-bps RS-232 port! As usual, I encourage you experiment yourself! 



## FROM THE BENCH

# Solid-State Lighting (Part 1)

## *An Introduction to Electroluminescence*

COLUMNS

Electroluminescence (EL) is the nonthermal conversion of electrical energy into light energy. Jeff explains how he set up an eight-channel EL controller for programming and animating up to eight EL panels.

By Jeff Bachiochi (US)

I always thought of night lights as a bit of Christmas all year round. These candelabra-based 7.5-W bulbs (C7), which are available all year in clear or white colors, are used as appliance bulbs. Toward the end of the year, the lights are available in retail stores in many colors as replacements for old strings of holiday lights. These days, miniature plug bulbs have taken over and provide more twinkle and less heat than the larger C7 bulbs. LEDs are quickly replacing the miniature incandescent bulbs and many feature the ability to change color.

I grew up in a two-story house on a quiet street in a once-bustling textile town. The second floor had four bedrooms and the home's only bathroom. A short hall connected all rooms and the stairway down to the main floor. Mom was always nervous that one of us would accidentally take a wrong turn in the dark on a night trip to the bathroom, so she put a night light in the hallway. We would always find a colored Christmas bulb to screw into the night light whenever the bulb burnt out. One night the normal festive shadows were replaced by an eerie glow. Was

this some alien invasion from a distant world? Unfortunately, no.

This was my first exposure to the world of electroluminescent (EL) materials. EL is the nonthermal conversion of electrical energy into light energy. Electrons passing through a powder phosphor in between two electrodes excite the phosphor to emit photons, giving off an eerie glow. Product manufacturers quickly caught on to this and augmented their products with this new solid-state light. The term light-emitting capacitor (LEC) was coined in the 1960s. While this is a great descriptive name, I've never heard it used.

### BACKLIGHTING

The EL panel's nondirectional emissions make it a natural choice for backlighting. It's been used in watches, pagers, thermostats, dashboards, and other products. Because of its unique properties (see **Table 1**), it will continue to be used in products for years to come.

If you've hiked or camped in the woods after dark, you might have come across some exposed phosphorus that takes on an eerie glow. I'm guessing this is some form of either

photoluminescence (emission due to exposure to light) or chemiluminescence (emission due to a chemical reaction). Nature is full of luminescence. As a kid I would never allow a summer evening to slip by without catching fireflies in a jar. Let’s take a closer look at how this material is manufactured and can be put to use.

The most common EL devices are known as thin-film electroluminescent (TFEL). Most TFEL devices comprise six layers: a substrate, a conductor, an insulator, a layer of phosphors, another insulator, and a second conductor. **Figure 1** depicts a typical TFEL construction. Starting from the outside, the substrate is a rigid or flexible material used as a base on which the additional layers are applied. The substrate can serve as the back or if transparent, as a protective front surface. Conductive layers form the device’s capacitive electrodes exposing the inner phosphorus to an alternating high voltage. Obviously, at least one conductive layer must be transparent to allow the emitted photons a way to escape. At least one transparent insulation layer is used to prevent the high voltage from arcing and shorting out the conducting electrodes. The high-voltage field penetrating the innermost phosphorus layer provides an energy gain to a phosphorus electron. The loss of energy in the alternating field then allows a transition of an excited phosphorus electron back to a lower energy state releasing the difference in energy between the two states as a photon.

While EL panels used in backlighting for LCDs are typically a single continuous emitter, a panel may contain multiple and separately controlled areas. Custom EL panels are becoming cost effective. While material, AC frequency, and voltage can have an effect on a panel’s color, you can choose from a number of standard colors (see **Photo 1**).

LINE VOLTAGE

A simple nightlight doesn’t required external electrical components beyond the EL panel. The 120 VAC delivered by our power grid falls within the specs of a typical EL panel:

Voltage: 100 to ~220 V (Optimum: 120 V)  
Frequency: 50 to 5,000 Hz (optimum: 1,500–2,000Hz)

When an EL panel is used beyond the reach of an AC outlet, the required AC must be produced locally, via a converter. Most distributors of EL panels also offer DC-AC converters. Since the AC they produce is potentially dangerous (high voltage), most are encapsulated to insulate said circuitry from the user. These may consist of an AA or AAA

Advantages	Disadvantages
Low wattage	Not practical for general lighting of large areas due to low lumen output of phosphors
Long life (reduced lumen output over time)	Poor lumens per watt
No external circuitry required (when plugged directly into AC power)	Reduced lumen output over time
Can be manufactured into flat flexible panels and other small shapes	Flexible flat EL sheets should not be flexed
More durable and light weight than LCDs or Plasma as display backlights.	The lamps require significant AC voltage: 60–600 V
Not directional emission, looks good at all angles	EL requires a converter when used with DC sources (uses higher frequency AC power, audible)
EL displays can handle –60°C to 95°C	

TABLE 1  
Depending on the application, EL panels have distinct advantages over other types of backlighting.

battery holder, power switch, converter, and output connector enclosed in a plastic case or simply a potted converter with input and output leads. The DC supply voltage required can be from 3 to 12 V.

I find most EL converters have a CM rating associated with them that is an important consideration when choosing one for a project. CM stands for square centimeters and is an indication of how much EL material it will drive. This is easy for EL panels, where the size is listed (i.e., 10 cm x 10 cm is 100 CM). However, EL material also comes in tape and wire. If you are using this material, you can figure that 20’ of wire or 40’ of tape is equal to approximately 100 CM. Many converters must have some minimum load to operate without self-destructing. Driving a smaller

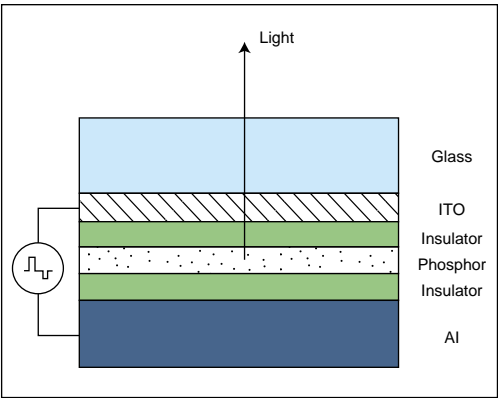
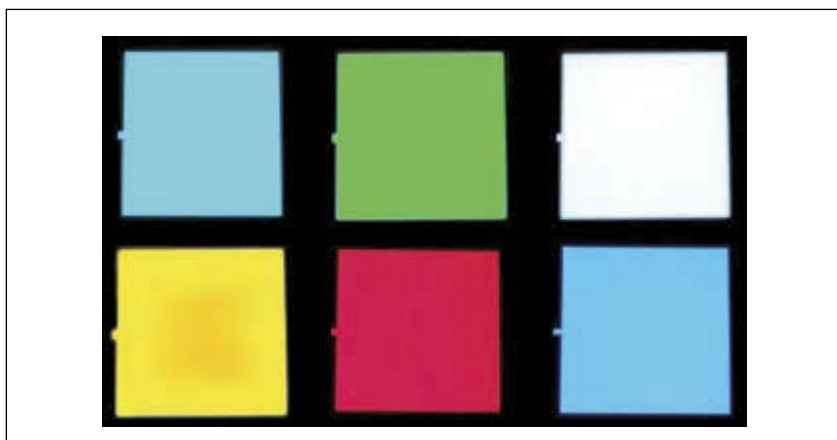


FIGURE 1  
TFEL devices utilize a thin phosphor film, such as manganese-doped zinc sulfide (ZnS:Mn), sandwiched between two insulating films, surrounded by transparent indium-tin-oxide (ITO) and non-transparent conductive electrode layers. An alternating (pulsed) electrical potential (about 120–200 V) applied between the two conducting electrodes generates an electric field that can cause the phosphorus electrons to obtain a higher energy state and then release photons with a loss of field energy.

**PHOTO 1**

These are some 10 cm x 10 cm panels I purchased from Adafruit.com and Sparkfun.com. This picture was taken in a darkened room as the lumen output is not overwhelming.

load will increase light output (as the voltage will be on the high side), while driving a larger load will decrease light output (as the voltage output will decrease with additional load).

## ONE IS NEVER ENOUGH

Because you are already familiar with EL used as backlighting on some phones, tablets, laptops and monitor/TVs you may not be intrigued until you see anything other than white. There is something about color that makes things really pop. Many are dabbling in the wearable wave, which is a mating of clothing and electronics. It could be adding headphones to a hat, medical sensor to a shirt, or EL as accents. While this brings back visions of *Tron*, a totally different industry has been gearing up to replace “high-priced” neon with EL.

Advertising signage has embraced the EL idea. While it doesn't have the intensity of neon, it has that same feel. It is inexpensive to duplicate as multiple colors and shapes can be combined into a single flat panel. Having this type of single panel is not cost effective for us. However, since the material can be easily cut, you can be creative in your use. Silk screening or masking with vinyl are other approaches that may not even require cutting standard-sized panels.

To achieve the animation of separate glowing shapes you need to have control over these separate areas. For instance, a simple three-cell sign might use three single panels each masked with the individual words “Eat,” “At,” “Joes.” You might want the words to illuminate one at a time and then flash together. This would require a way of turning on and off each EL panel individually. In this case, we need three channels of control. Since a port has 8 bits, this is a nice number to work with. Let's look at setting up an eight-channel EL controller that will provide you with a simple way of programming the animation for up to eight EL panels.

## CONTROLLING AC

Many of you have probably used a TRIAC to control an AC voltage. While we could use mechanical relays to do this, the solid-state approach is a bit less expensive and doesn't have the potential contact wear issues that relays can experience. Before we had electric drills with squeeze speed control, there was only one speed, on. I remember buying a speed control from RadioShack with a knob that let you control the speed of your drill. This was a circuit containing a silicon control rectifier (SCR), the precursor to the Triac. The SCR is an electrically controlled one-way switch. The device affects only one polarity of the AC waveform. The knob or potentiometer along with a fixed capacitor formed an RC network that would delay the SCR switch from turning on for its polarity once each full cycle. This meant that if the SCR was prevented from turning on at all that the drill would receive only half of the AC voltage waveform and run very slow (or not at all). By allowing the SCR to turn on at various delays from the zero-crossing point, you have full control over the other half of the AC waveform and have full control over one polarity of the waveform. The AC itself provides the repetitive timing necessary to control the SCR's gate.

The obvious next step was using two parallel SCRs (in opposite polarity) to control both halves of the AC waveform. This was again improved by combining all the circuitry necessary into a single three terminal device, the TRIAC. While many of today's tools are DC Lithium Ion based, TRIACs are still used in devices that run on AC.

The DC-to-AC converter you might use for you EL panels can be controlled using a TRIAC. I'm talking about a rather high voltage here (greater than 100 V), so use caution when it comes to exposed circuitry. In addition, if you want a microcontroller to control this, it should be isolated from the high voltage. Use an optical isolator to isolate a microcontroller from the high-voltage AC. Many configurations are available. I used a device that has an LED as the control source for an optically coupled switch for an external TRIAC. Since the switch is optically coupled to an LED, the logic side circuitry remains physically disconnected from the high voltage (except for optionally grounding one side of the AC). The external TRIAC's gate is turned on whenever the isolated LED is illuminated.

The “when” is the important part here. That's because once a TRIAC has been turned on by its gate, the TRIAC will stay on even if the gate signal is removed until the AC voltage reaches zero (which it does twice per cycle). If the gate remains on, the TRIAC is turned on again until again reaching zero and the



### ABOUT THE AUTHOR

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@imagine-thatnow.com](mailto:jeff.bachiochi@imagine-thatnow.com) or at [www.imaginethatnow.com](http://www.imaginethatnow.com).



gate drive has been removed. You can choose when to turn it on for any half cycle based on the zero-crossing. It is best to turn a TRIAC on when the current is zero to prevent (potential) instantaneous high currents. To assure this is the case, you need either to monitor zero crossings and apply a control signal only when the voltage is minimum or use a isolated device that contains a zero-crossing detector.

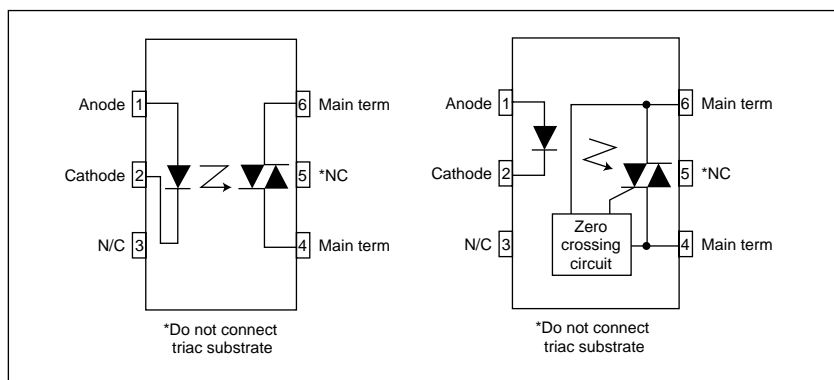
**Figure 2** shows the block diagrams of two similar isolated devices, the MOC302x and MOC303x. When all you want to do is turn on or off the device, use an isolator that has a zero-crossing detector (ZCD) built-in. This allows the device to decide the appropriate time to apply the gate drive. When you need to perform PWM control of each cycle (for dimming purposes), you must use an isolator without zero-crossing so that you can force the device on during any point in the cycle. If a delay of 4 ms (approximately half of each 60-Hz half cycle) is repeated, the average power to the load is reduced to around 50%. This only works when you sync the delay to each zero-crossings.

## PROTOTYPE 1

The first circuit in this project will be used to turn eight EL loads on and off. It is based on providing an external AC source using an available DC-to-AC converter. **Figure 3** shows eight similar TRIAC circuits controlled via a microcontroller using zero-crossing optoisolated devices for circuit and safety protection. In this instance, we have no control of the converter's frequency—that is, the AC frequency—so we allow the ZCD to handle when in each cycle to apply the gate voltage to the external TRIAC.

If I were manufacturing an “Eat at Joe’s” sign, I could hard code the timing parameters of the animation into the code. There would be no reason to ever change these. However, for experimental purposes, I want to be able to change this on the fly. This means that there must be some kind of user interface. A simple serial interface handles this nicely.

I had to define a few commands to define parameters. My first thought was to use music instrument digital interface (MIDI), which was designed as a way to translate all aspects necessary to replicate a musical score into individual command instructions like note, duration, beat, instrument, and volume. This is very structured and allows multiple instruments to play in one cohesive group. This orchestration is much like what is necessary to animation objects. That’s probably why other control equipment often uses formats similar to the MIDI standard. There is plenty of available software for



**FIGURE 2**

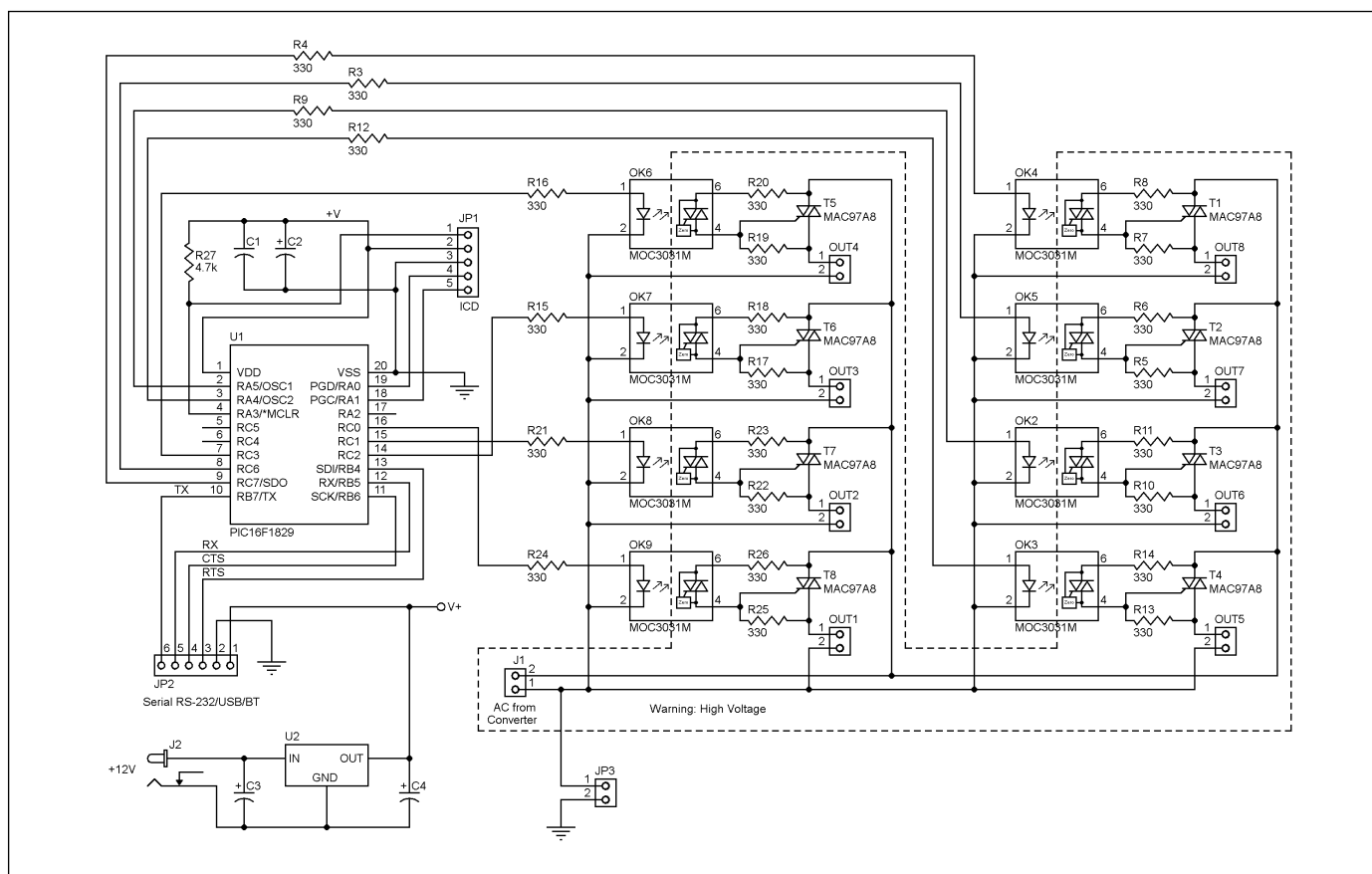
Here are two optically isolated TRIAC drivers that can be used with an external TRIAC to control the high-voltage AC required to excite an EL panel. These devices have identical footprints so using an IC socket in a circuit allows for some experimenting with either device. Without the zero-crossing detector the device can turn on an external TRIAC at any point in each half cycle. The driver with the ZCD will delay any command to turn on until a zero-crossing has occurred enabling an external TRIAC when the voltage is minimum.

constructing MIDI commands with an easy-to-use GUI. The MIDI command can then be interpreted as actions other than choosing how long a particular note is sounded for. I’m not using this for two reasons: it requires a special interface and the data is binary and not ASCII. While the interface is not complex (it’s essentially serial), there is a learning curve associated with MIDI that really isn’t necessary here. Also, I want all the commands to be ASCII, so they can be typed in using a simple terminal program, like RealTerm.

The protocol I used consists of a string of commands that make up a single channel’s animation. We only need three commands here: Channel, ON, and OFF. Channel is represented by the letter C, ON by the letter O, and OFF by the letter X. Each of these commands has a value associated with it. For Channel, we use this to select the channel number (1 to 8) for the remaining commands. ON and OFF require a time or duration represented by a decimal number between 1 and 255. The time is tenths of a second. So, if you want a duration of 1 s, you use the value 10. This allows each command to indicate a duration of up to 25.5 s. If we break down the animation of the “Eat at Joe’s” sign into three channels, we get:

```
C1 05 X5 X5 X5 X10 X10 X10 X10 X10 <CR>
C2 X5 X5 05 X5 X10 X10 X10 X10 X10 <CR>
C3 X5 X5 X5 X5 010 010 010 X10 X10 <CR>
```

Note the following groups in Channel 3: X5, X5, X5 X5, 010, 010, 010, X10, and X10. Each group totals 10/10 tenths or



**FIGURE 3**

This is the basic control circuitry for eight EL panels connected to an external DC-AC converter. EL animation, or the programmed on/off sequences of each channel, is handled using three commands via a serial terminal. Programmed sequences are saved to EEPROM.

1 s. I wrote it like this so that it's easier to understand, as all actions are in sync (they don't necessarily have to be). Channel 1 is on for 0.5 s and then off for 0.5 s, while the others channels remain off. Next, Channel 2 is on for 0.5 s and then off for 0.5 s, while the others channels remain off. Then, Channel 3 is on for 3 s while the others channels remain off. Finally, all channels are off for 2 s. This animation would then repeat endlessly.

The following command strings produce the same results, but it is much more difficult to relate the actions of one channel to another.

C1 05 X65 <CR>

```
C2 X10 05 X55 <CR>
C3 X20 030 X20 <CR>
```

When commands are entered, execution of the present animation stops. Command strings entered are saved into EEPROM and remain with the application until changed. Animation begins using any commands stored in EEPROM upon the next reset. The present commands are sent out the serial port before execution begins. This will allow the user to check what is presently programmed into the device. Only Channels entered will be changed. Previously stored commands for other Channels will remain unless they are deliberately removed using a empty string.

C4 &lt;CR&gt;

## TIMING IS EVERYTHING

As described earlier the animation timing is based on tenths of a second. Therefore, we need to initialize a timer to give a interrupt every 0.1 s. Based on an internal clock speed of 8 MHz, the execution speed of each instruction will be 2 MHz (8 MHz/4), or 500 ns.



[circuitcellar.com/ccmaterials](http://circuitcellar.com/ccmaterials)

## RESOURCES

Edison Tech Center, "Electroluminescent Lamps," 2013, [www.edisontechcenter.org/electroluminescent.html](http://www.edisontechcenter.org/electroluminescent.html).

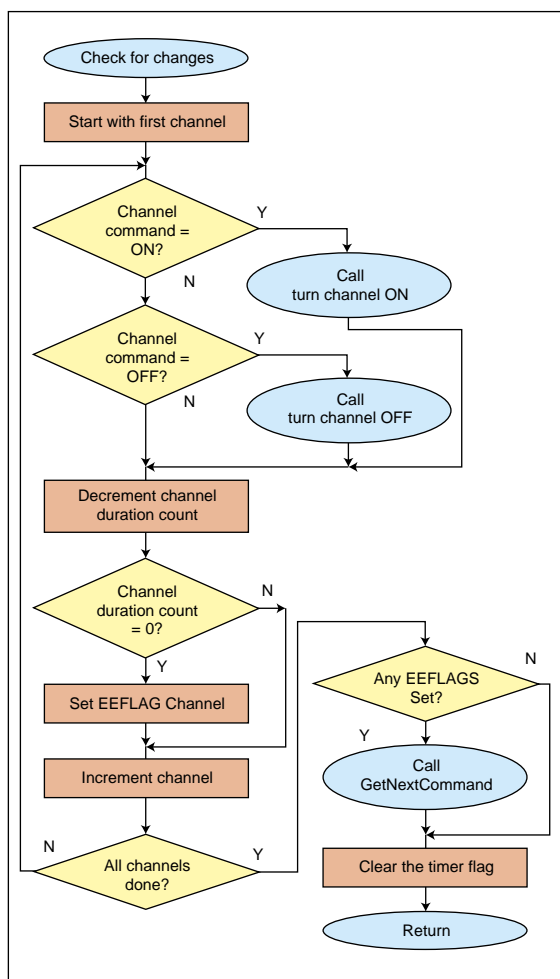
What-When-How.com, "Organic Photovoltaic Cells (OPVCs)," <http://what-when-how.com/electronic-properties-of-materials/applications-optical-properties-of-materials-part-7/>.

This is also the clock input for Timer 2, Timer 4, and Timer 6. Each timer has a prescaler, an 8-bit auto-reload counter, and a post-scaler. The prescaler can slow the clock by a factor of 1, 4, 8, or 64. A prescale value of 64 will slow the (500 ns) clock down to 32  $\mu$ s. If we divide this into the time we want (100,000  $\mu$ s), we get 3,125, which is too big for 8-bit register, so we need reduce this further by the post-scaler. Setting the post-scaler to divide by 16 will bring the count down to 195.3125. By initializing the PR4 register to 195, the 8-bit count down register is automatically loaded with that value each time it reaches zero. So, we have a clock slowed to 32  $\mu$ s, counting from 195 to 0 (32  $\mu$ s  $\times$  195 counts = 6,240  $\mu$ s), divide by a post-scale of 16 (6,240  $\mu$ s  $\times$  16 = 0.099840  $\mu$ s). In reality, the actually interrupt time will be off by 160  $\mu$ s, which is approximately 0.16%. This is due to the fractional part of a 32- $\mu$ s count times the post-scaled divisor (32  $\mu$ s  $\times$  0.3125  $\times$  16 = 160  $\mu$ s).

With a time base of 0.1 s, I can make decisions on potential channel changes every interrupt. Refer to **Figure 4** to see what must happen every 100 ms. At power-up, the EEPROM content is sent out the serial port as record of which channel parameters are executing. Each channel is initialized with the first command and duration read from the EEPROM. With this done, the interrupts are enabled, and when Timer 4 overflows, we get an interrupt. Actually, there is no reason we can't just poll for the Timer 4 overflow in the main loop; however, I use the interrupt to set a "Check for Changes" flag and also produce a debug pulse on an unused output pin. The main loop does nothing until a character is received or a Timer 4 interrupt sets the "Check for Changes" flag.

It is important that we can complete any necessary execution in less time than it takes Timer 4 to overflow again. Therefore, I set a second unused output bit when the routine begins and clear it when finished. With these two outputs, I can see when Timer 4 interrupts occur and how long the "Check for Changes" routine takes in the main loop. It's a good idea to determine the routine time. You can physically count the number of program steps required for the longest path through the routine, or you can use the stopwatch function in the simulator to record the time.

Extensive use of the indirect registers allows the same routines to be used as the code cycles through all eight channels. In reality, you don't need to turn on any channel that is already on (or vice versa). However, it takes longer to check for the state of an output then it does to just set or clear it again. This is done for all channels each interrupt.




**FIGURE 4**

Here's a flow chart of the "Check for Changes" routine that is executed each time an interrupt occurs from the Timer 4 overflow (100 ms). The present channel command determines the channel's new output state until the command's duration value (number of 100-ms ticks) has been decremented to zero. Once the state of all channel outputs has been set, if any command's duration has reached zero, a new command is fetched for that channel from the EEPROM.

In addition we need to reduce the duration count each interrupt. When that duration reaches zero, an EEFLAG flag is set to indicate that a new command is required. When a new command read is equal to 0x0D, the command pointer for that channel is reset, allowing the string to repeat. When a channel reads 0x0D, the command pointer is also checked to see if there are any commands. If there are no actual commands for the channel, a fake OFF command with a duration of 255 is created to prevent anything from happening on that channel's output. Any time you program multiple channels to be on at the same time, the brightness will be somewhat less since all channels are all sharing the same AC source.

## ANIMATE

At times I wish I had a more artistic flair. Coming up with flashy (no pun intended) artwork is just not one of my talents. However, with the basics under my belt, I can experiment using EL to illuminate and animate. Next month, I'll continue this discussion with a bit more information and a few more circuits you might find interesting. 



## CC SHOP



1

**1 CC VAULT**

CC Vault is a pocket-sized USB that comes fully loaded with every issue of *Circuit Cellar* magazine! This comprehensive archive provides an unparalleled amount of embedded hardware and software design tips, schematics, and source code. CC Vault contains all the trade secrets you need to become a better, more educated electronics engineer!

*Item #: CCVAULT*

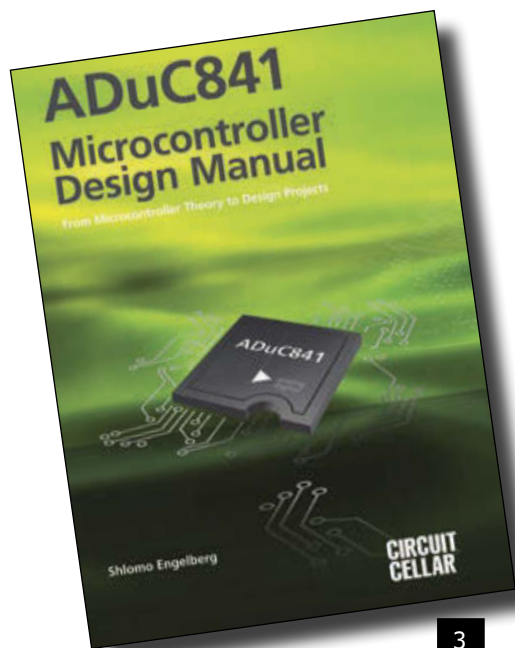


2

**2 CC 2014 CD**

2014 was an exciting year for electronics engineers! The continued success of open-source solutions, Internet of Things (IoT) revolutions, and green-energy consciousness has changed the face of embedded design indefinitely. In *Circuit Cellar's* 2014 archive CD, you can find all of these hot topics and gain insight into how experts, as well as your peers, are putting the newest technologies to the test. You'll have access to all articles, schematics, and source code published from January to December 2014.

*Item #: CD-018-CC2014*



3

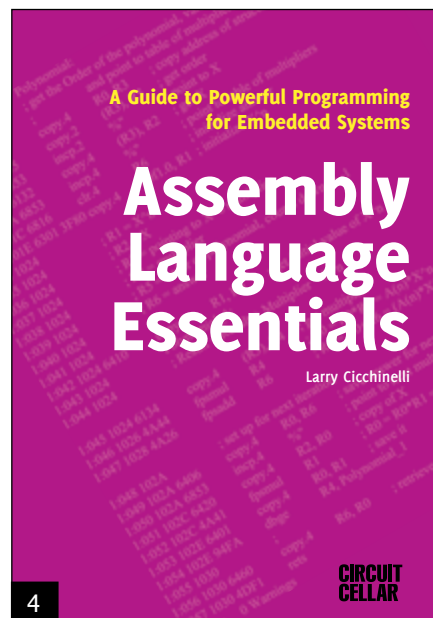
**3 ADUC841 MICROCONTROLLER DESIGN MANUAL**

This book presents a comprehensive guide to designing and programming with the Analog Devices ADuC841 microcontroller and other microcontrollers in the 8051 family. It includes a set of introductory labs that detail how to use these microcontrollers' most standard features, and includes a set of more advanced labs, many of which make use of features available only on the ADuC841 microcontroller.

The more advanced labs include several projects that introduce you to ADCs, DACs, and their applications. Other projects demonstrate some of the many ways you can use a microcontroller to solve practical problems. The Keil  $\mu$ Vision4 IDE is introduced early on, and it is used throughout the book. This book is perfect for a university classroom setting or for independent study.

*Author: Shlomo Engelberg*

*Item #: CC-BK-9780963013347*



4

**4 ASSEMBLY LANGUAGE ESSENTIALS**

Looking to brush up your programming skills? Get back to the basics with this matter-of-fact guide to Assembly language. Perfect for advancing students and academics, this book introduces you to a processor's most fundamental programming language. It includes essential terminology pertaining to higher-level programming, important algorithms that can be built into high-level language, a free downloadable Assembler program, and much more.

*Author: Larry Cicchinelli*

Further information and ordering: [www.cc-webshop.com](http://www.cc-webshop.com)

**CONTACT US:** Circuit Cellar, Inc. | Phone: 860.289.0800 | E-mail: [custservice@circuitcellar.com](mailto:custservice@circuitcellar.com)

**What's your EQ?** The answers are posted at [www.circuitcellar.com/category/test-your-eq/](http://www.circuitcellar.com/category/test-your-eq/). You can contact the quizmasters at [eq@circuitcellar.com](mailto:eq@circuitcellar.com).

### PROBLEM 1

The capacitance of the plates drops with increasing distance, so the voltage between them rises, because the charge doesn't change and the voltage is equal to the charge divided by the capacitance. At first, while the plate spacing is still small relative to their diameter, the capacitance is proportional to the inverse of the spacing, so the voltage rises linearly with the spacing. However, as the spacing becomes larger, the capacitance drops more slowly and the voltage rises at a lower rate as well.

While the plate spacing is small, the electric field is almost entirely directly between the two plates, with only minor "fringing" effects at the edges. Since the voltage rise is proportional to the distance in this regime, the electric field (e.g., in volts per meter) remains essentially constant. However, once the plate spacing becomes comparable to the diameter of the plates, and fringing effects begin to dominate, the field begins to spread out and weaken. Ultimately, at very large distances, at which the plates themselves can be considered points, the voltage is essentially constant, and the field strength directly between them becomes proportional to the inverse of the distance.

### PROBLEM 2

There is an attractive force between the plates of a capacitor created by the electric field. Physically moving the plates apart requires doing work against this force, and this work becomes the additional potential energy that is stored in the capacitor.

# TEST YOUR EQ

*Contributed by David Tweed*

### PROBLEM 3

Dielectric materials are made of atoms, and the atoms contain both positive and negative charges. Although neither the positive nor the negative charges are free to move about in the material (which is what makes it an insulator), they can be shifted to varying degrees with respect to each other. An electric field causes this shift, and the shift in turn creates an opposing field that partially cancels the original field. Part of the field's energy is absorbed by the dielectric. In a capacitor, the energy absorbed by the dielectric reduces the field between the plates, and therefore reduces the voltage that is created by a given amount of charge. Since capacitance is defined to be the charge divided by the voltage, this means that the capacitance is higher with the dielectric than without it.

### PROBLEM 4

With certain dielectrics, most notably quartz and certain ceramics, the displacement of charge also causes a significant mechanical strain (physical movement) of the crystal lattice. This effect works two ways—a physical strain also causes a shift in electric charges, creating an electric field. This effect can be exploited in a number of ways, including transducers for vibration and sound (mics and speakers), as well as devices that have a strong mechanical resonance (e.g., crystals) that can be used to create oscillators and filters.

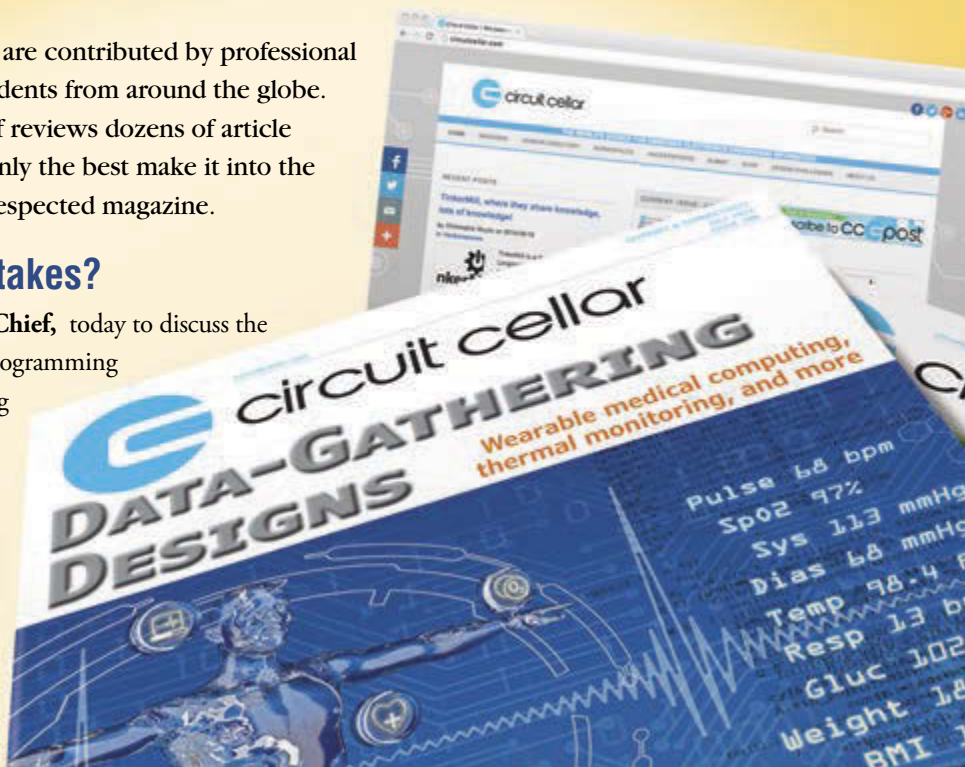
## Get PUBLISHED. Get NOTICED. Get PAID.

*Circuit Cellar* feature articles are contributed by professional engineers, academics, and students from around the globe. Each month, the editorial staff reviews dozens of article proposals and submissions. Only the best make it into the pages of this internationally respected magazine.

### Do you have what it takes?

Contact **C. J. Abate, Editor-in-Chief**, today to discuss the embedded design projects and programming applications you've been working on and your article could be featured in an upcoming issue or online at [circuitcellar.com](http://circuitcellar.com).

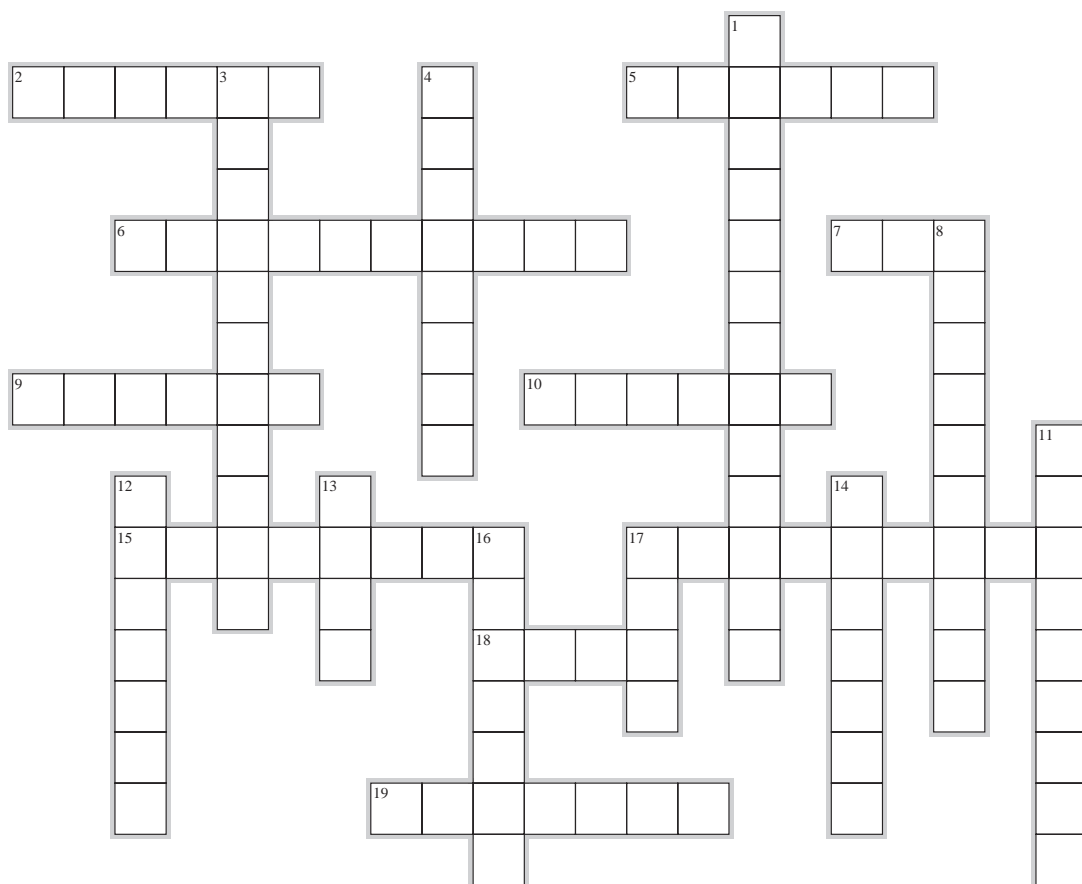
Email: [editor@circuitcellar.com](mailto:editor@circuitcellar.com)



# CROSSWORD

**FEBRUARY 2015**

The answers will be available at [circuitcellar.com/crossword](http://circuitcellar.com/crossword).



## ACROSS

2. A connection or conductor that a number of circuits share
5. Comprises a cathode, plate, and control grid
6. The tendency for electrons at high frequencies to travel along the surface of a conductor [two words]
7. Big Blue
9. Pa
10. Tube that protects a bundle of wires
15.  $6.0221415 \times 10^{23}$  atoms/mole
17. A capacitor
18. M
19. Measures the damping of resonator modes

## DOWN

1. Figure-8
3. Self-sustaining generation of a continuous electrical signal
4. Cathode bias [two words]
8. In this code, common in telecommunications, each bit of data is represented by at least one voltage level transition.
11. WW
12. Straight line that touches a circuit a single point
13. Hot signal connector
14. Containing iron
16.  $R = V/I$  [two words]
17. 0.01 bar



# IDEA BOX the directory of PRODUCTS & SERVICES

For current rates, deadlines, and more information contact Peter Wostrel at 978.281.7708 or circuitcellar@smmarketing.us.

**ALL ELECTRONICS CORPORATION**

**Electronic and Electro-mechanical Devices, Parts and Supplies.**  
Many unique items.

We have what you need for your next project.

**www.allelectronics.com**  
Free 96 page catalog 1-800-826-5432

**Silver Ball Matrix Sockets**

- Over 40GHz bandwidth @ -1dB for edge pins
- Contact Resistance under 15mOhms
- Self Inductance under 0.2 nH
- Capacitance under 0.1 pF
- Operating temperature range -55C to +150C
- Insertion/Extraction life over 500,000 cycles with protective plunger matrix
- Current rating at 15C temperature rise is 4 Amps per pin

**Ironwood ELECTRONICS** 1-800-404-0204  
**www.ironwoodelectronics.com**

**MaxBotix®**  
High Performance Ultrasonic Rangefinders

**4-20HR-MaxSonar®-WR™**

- 4-20mA output
- High noise tolerance
- IP67 rated
- 1.6 mm resolution
- Multi-Sensor operation
- Calibrated beam pattern
- 8Hz read rate
- End user solution
- Ideal for industrial use

**UCXL-MaxSonar®-WR™**

- Great for design engineers
- Multiple mounting options
- Light weight industrial sensor
- Incredible noise immunity
- Smallest IP67 sensor in size
- 1cm resolution
- Automatic calibration
- Ideal for outdoor UAV use

Phone: 218-454-0766 Email: sales@maxbotix.com  
**www.maxbotix.com**

## \$20 for 5PCBs

2 layer, 4x4inch, FR4(RoHS), 0.063", 1oz,  
2LPI, Green, 15K, Lead free HASL

Standard PCB: Promotion code:

CC14061

## PCB & PCBA

Small to Mass QTY

INSTANT QUOTE AT:

**www.myropcb.com**

OR CALL:

**1-888-PCB-MYRO**

## Join The INTERNET of THINGS REVOLUTION



PIONEERING IoT SINCE 2001

**TRI**

Programmable Logic Controllers

Powerful & Easy Ladder  
+BASIC Programming  
Ethernet integrated  
MODBUS TCP/IP  
DI/Os & AI/Os integrated

OEM Prices as low as \$119  
for full-feature Nano-10 PLC

tel : 1 877 TRI-PLCS  
web : [www.triplc.com/ccl.htm](http://www.triplc.com/ccl.htm)

**TRI** TRIANGLE  
RESEARCH  
INTERNATIONAL

**PIC-SERVO**  
MOTION CONTROL

MOTION CONTROLLERS FOR  
BRUSH, BRUSHLESS AND  
STEPPER MOTORS.

- controller chips
- controller boards

**www.picservo.com**  
JEFFREY KERR, LLC

**RFID ASAP!**  
Develop RFID with a PIC®

**Starting at \$90**

**RFID Development Kit**  
Exercises, Tutorials & Boards  
Create RFID Systems Today!

**C Compilers available  
for all PIC® MCU Devices**

Start Programming now!  
**www.ccsinfo.com/CC215**

sales@ccsinfo.com **CCS**  
262-522-6500 x 35

PIC® MCU is a registered trademark of Microchip Technology Inc.

**PLCC68 Series** **HUMANDATA**  
Stamp size FPGA/CPLD Module **from JAPAN**

- Designed for 68-pin PLCC socket
- Very small size (25.3 x 25.3 [mm])
- 50 I/Os (External clock inputs available)
- 3.3V single power supply operation

All series same pin assignment

**XILINX Series** RoHS compliant  
FPGA(Spartan-6, Spartan-3AN)  
Oscillator, Configuration Device

**ALTERA Series** RoHS compliant  
FPGA(Cyclone V, Cyclone III,  
MAX V, MAX II, ...)  
Oscillator, Configuration Device,  
FRAM, ...

**All PLCC68 Series are in stock @Amazon!!**

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

# The Future of Embedded Linux

By David Lynch



David Lynch owns DLA Systems ([www.dlasys.net](http://www.dlasys.net)). He is a software consultant and an architect, with projects ranging from automated warehouses to embedded OS ports. When he is not working with computers, he is busy attempting to automate his house and coerce his two children away from screens and into the outdoors to help build their home.

**M**y first computer was a Cosmac Elf. My first “Desktop” was a \$6,500 HeathKit H8. An Arduino today costs \$3 and has more of nearly everything—except cost and size—and even my kids can program it. I became an embedded software developer without knowing it. When that H8 needed bigger floppy disks, a hard disk, or a network, you wrote the drivers yourself—in assembler if you were lucky and machine code if you were not.

Embedded software today is on the cusp of a revolution. The cost of hardware capable of running Linux continues to decline. Raspberry Pi (RPi) can be purchased for \$25. A Beagle Bone Black (BBB) costs \$45. An increasing number of designers are building products such as Cubi, GumStik, and Olinuxino and seeking to replicate the achievements of the RPi and BBB, which are modeled on the LEGO-like success of Arduino.

These are not “embedded Linux systems.” They are full-blown desktops—less peripherals—that are more powerful than what I owned less than a decade ago. This is a big deal. Hardware is inexpensive, and designs like the BBB and RPi are becoming easily modifiable commodities that can be completed quickly. On the other hand, software is expensive and

ship 5,000 units per year, could make the product unviable.

Products have to be inexpensive, high-quality, and fast. They have to be on the shelves yesterday and tomorrow they will be gone. The bare metal embedded model can’t deliver that, and there are only so many software developers out there with the skills needed to breathe life into completely new hardware.

That is where the joy in embedded development is for me—getting completely new hardware to load its first program. Once I get that first LED to blink everything is downhill from there. But increasingly, my work involves Linux systems integration for embedded systems: getting an embedded Linux system to boot faster, integrating MySQL, and recommending an embedded Linux distribution such as Ubuntu or Debian to a client. When I am lucky, I get to set up a GPIO or write a driver—but frequently these tasks are done by the OEM. Today’s embedded ARMs have everything, including the kitchen sink integrated (probably two).

Modern embedded products are being produced with client server architectures by developers writing in Ruby, PHP, Java, or Python using Apache web servers and MySQL databases and an assortment of web clients

communicating over an alphabet soup of protocols to devices they know nothing about. Often, the application developers are working and testing on Linux or even Windows desktops. The time and skills needed to value engineer the software to accommodate small savings in hardware costs do

---

*Once I get that first LED to blink everything is downhill from there. But increasingly, my work involves Linux systems integration for embedded systems: getting an embedded Linux system to boot faster, integrating MySQL, and recommending an embedded Linux distribution such as Ubuntu or Debian to a client.*


---

slow. Time to market is critical. Target markets are increasingly small, with runs of a few thousand units for a specific product and purpose. Consumers are used to computers in everything. They expect computers and assume they will communicate with their smart phones, tablets, and laptops. Each year, consumers expect more.

There are not enough bare metal software developers to hope to meet the demand, and that will not improve. Worse, we can’t move from concept to product with custom software quickly enough to meet market demands. A gigabyte of RAM adds \$5 to the cost of a product. The cost of an eight-week delay to value engineer software to work in a few megabytes of RAM instead, on a product that may only

not exist. When clients ask for an embedded software consultant, they are more likely after an embedded IT expert, rather than someone who writes device drivers, or develops BSPs.

There will still be a need for those with the skills to write a TCP/IP stack that uses 256 bytes of RAM on an 8-bit processor, but that growing market will still be a shrinking portion of the even faster growing embedded device market.

The future of embedded technology is more of everything. We’ll require larger and more powerful systems, such as embedded devices running full Linux distributions like Ubuntu (even if they are in systems as simple as a pet treadmill) because it’s the easiest, most affordable solution with a fast time to market. 

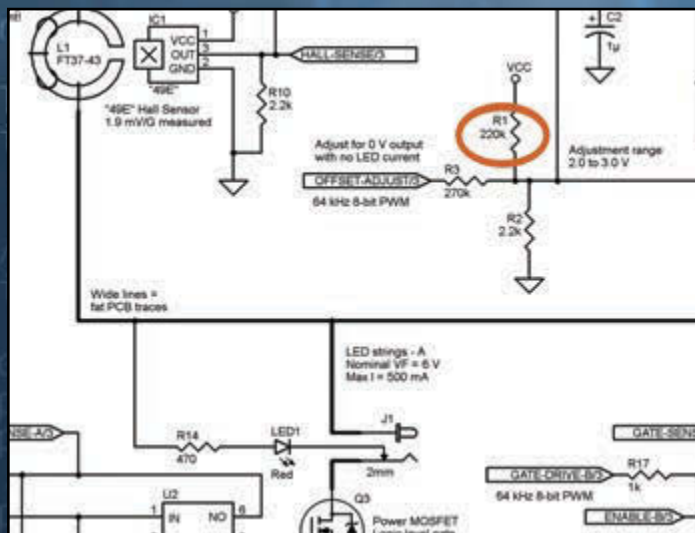




# MONTHLY ENGINEERING CHALLENGE

Each month, you're challenged to find an error in a schematic or in code that's presented on the challenge webpage. Locate the error for a chance to win prizes and recognition in Circuit Cellar magazine!

Prizes such as a NetBurner MOD54415 LC Development kit or a Circuit Cellar subscription will be announced each month.



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int first, second, sum;
6     int *p, *q;
7
8     printf("Enter two integers to add: \n");
9     scanf("%d%d", &first, &second);
10
11     p = &first;
12     q = &second;
13
14     sum = p + q;
15
16     printf("Sum of entered numbers = %d\n", sum);
17
18     return 0;
19 }
```

**Participate:** [circuitcellar.com/engineering-challenge-netburner](http://circuitcellar.com/engineering-challenge-netburner)

**Launch:** 1st of each month

**Deadline:** 20th of each month

No purchase necessary to enter or win. Void where prohibited by law. Registration required. Prizes subject to change based on availability. Review these terms before submitting each Entry. More info: [circuitcellar.com/engineering-challenge-netburner-terms](http://circuitcellar.com/engineering-challenge-netburner-terms)



**We bring the full range of Electronic Contract Manufacturing services to your fingertip!**

**FABRICATION**



**ASSEMBLY**



**KEYPADS**



**ENCLOSURES**



**This is the only place where you would put all your eggs in one basket to get fastest time to market. From concept design to prototype to full turnkey production on all your electronic products.**



**imagineering inc. [www.PCBnet.com](http://www.PCBnet.com)**

**847-806-0003 [sales@PCBnet.com](mailto:sales@PCBnet.com)  
Certified Woman-Owned Small Business**