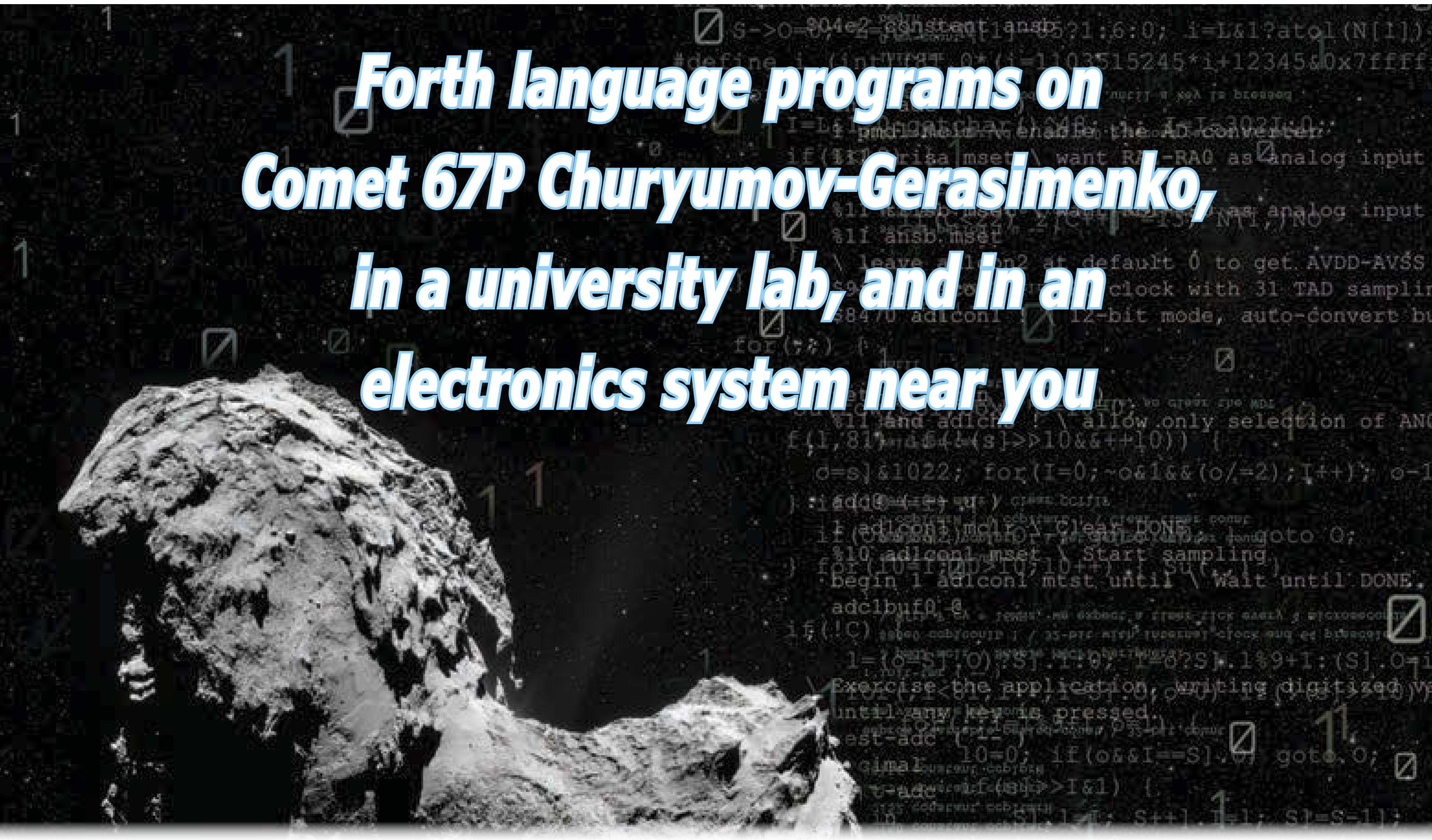
circuitcellar.com

EMBEDDED PROGRAMMING
APRIL 2015
ISSUE 297

circuit cellar

Fast-Forward with FlashForth



Forth language programs on Comet 67P Churyumov-Gerasimenko, in a university lab, and in an electronics system near you

■ Q&A: The IoT as a Disruptive Force | CC Picks: Embedded Programming
■ FlashForth in the Lab | DIY LCDTV Server | Prototyping with SuperSpeed USB | CC Archive Article: Quad Bench Power Supply ■ SuperSpeed for FPGAs | Software Dev Predictability | Estimate Software Costs | ESD Protection Tips | Intro to Programmable Logic Controllers ■ The Future of Embedded Security

$9.00US $10.00CAN
04
0 74470 75349 0

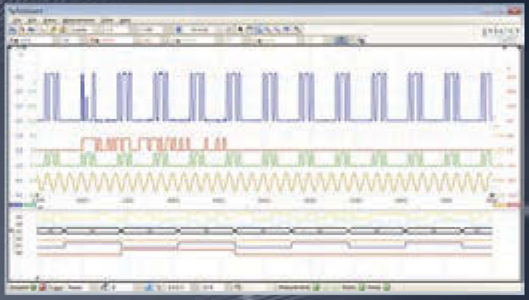
pico Technology

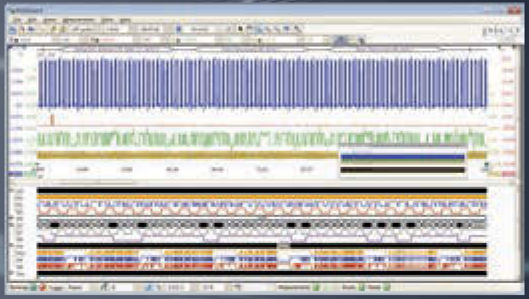
MIXED SIGNAL OSCILLOSCOPES

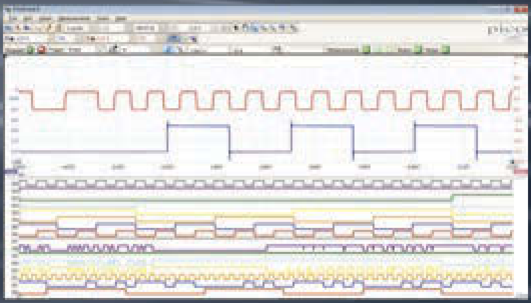
4 ANALOG + 16 DIGITAL CHANNELS

RAPIDLY DEBUG COMPLEX MIXED SIGNAL DESIGNS

- USB 3.0
- ULTRA DEEP MEMORY
- SEGMENTED MEMORY
- RAPID TRIGGERS







INCLUDES AUTOMATIC MEASUREMENTS, SPECTRUM ANALYZER, SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, I²S, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD, WITH FREE UPDATES . 5 YEAR WARRANTY

	3204D MSO	3205D MSO	3206D MSO	3404D MSO	3405D MSO	3406D MSO
Channels	2 analog, 16 digital			4 analog, 16 digital		
Bandwidth	60 MHz	100 MHz	200 MHz	60 MHz	100 MHz	200 MHz
Buffer memory	128 MS	256 MS	512 MS	128 MS	256 MS	512 MS
Max. sampling rate	1 GS/s					
Signal generator	Function generator + Arbitrary waveform generator					
Digital inputs	100 MHz max. frequency, 500 MS/s max. sampling rate					

www.picotech.com/pco539

circuit cellar

Sponsored by NetBurner

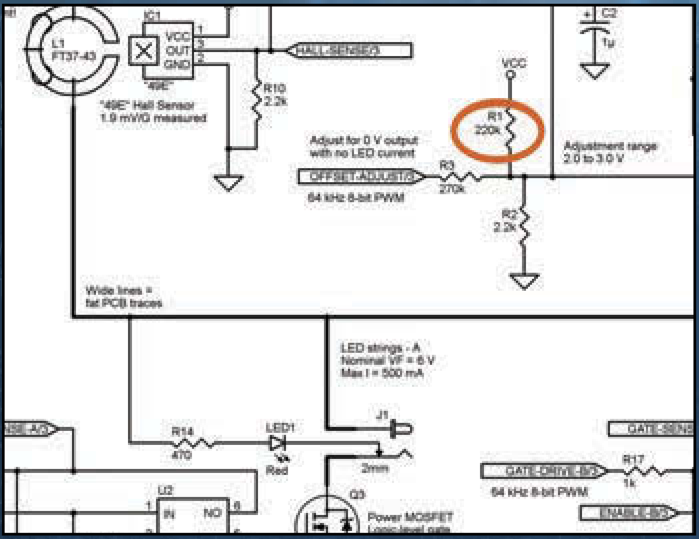
NetBurner

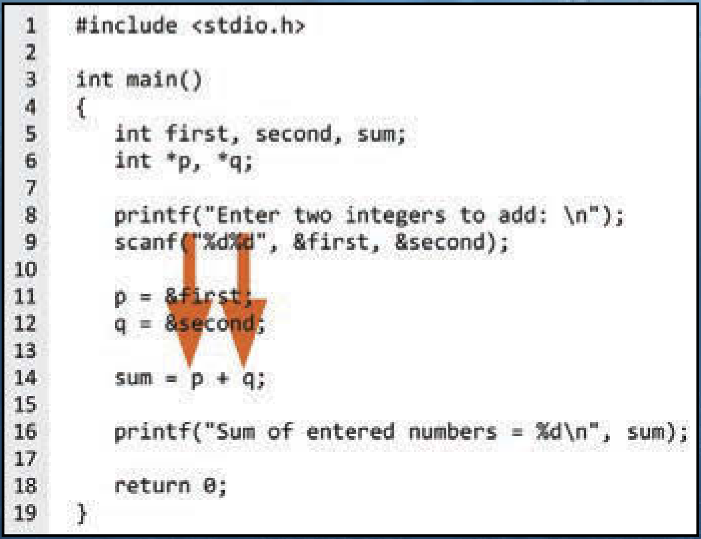
MONTHLY

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.





```
#include <stdio.h>

int main()
{
    int first, second, sum;
    int *p, *q;

    printf("Enter two integers to add: \n");
    scanf("%d%d", &first, &second);

    p = &first;
    q = &second;

    sum = p + q;

    printf("Sum of entered numbers = %d\n", sum);

    return 0;
}
```

Participate: circuitcellar.com/engineering-challenge-netburner
Launch: 1st of each month
Deadline: 20th of each month

TOPICS IN EMBEDDED PROGRAMMING

Circuit Cellar articles on embedded programming have become increasingly popular since 2011, which is when we actively set out to seek and publish more content on the subject. That year, we ran a variety of memorable, well-received articles, such as George Novacek's "Reliable Programming Work Toward Fault-Free Software." We also contracted Bob Japenga to write regular column on embedded Linux.

This month, we continue the tradition with insightful embedded programming-related articles on Flash language, software development predictability, and tips for embedded software development. Let's begin by reviewing what you'll find in each article.

In "FlashForth in the Laboratory," Peter Jacobs first provides some basic background information on the Forth language, which is well suited for real-time machine control (p. 18). He then covers FlashForth and explains how Forth is used in a university lab to enable convenient real-time interaction with hardware.

On page 46, George Novacek tackles the topic of software failure modes effects analysis. He takes a close look at software FMECA (SWFMECA) and its potential for making software development more predictable.

Turn to page 50 for Bob Japenga's new article on estimating the costs of an embedded systems project. In particular, he presents four heuristics for embedded software development.

Once you've had your fill of programming content, take a look the rest of the issue. We present project articles as well as some handy electrical engineering tips.

On page 28, Lindsay Meek describes his WIZnet WIZ550io-based LCDTV Server project. The design enables an LCD TV equipped with a USB port to stream media across a LAN.

This month, we have two articles on the topic of SuperSpeed USB. First, John Hyde presents a guide to the end-to-end development of a SuperSpeed USB device (p. 34). Then, on page 40, Colin O'Flynn covers the use value of SuperSpeed USB for FPGAs.

Flip to page 54 for Robert Lacoste's article on electrostatic discharge (ESD), which can create problems for you electronics projects. Follow his advice to protect your designs.

If you're interested in learning about programmable logic controllers (PLC), turn to Jeff Bachiochi's article, "Ladder Logic," on page 60. After covering the basics, he explains how PLCs operate.

We wrap up the issue with an essay about the future of embedded security. In "Security Agents for Embedded Intrusion Detection," Syed Kamran Haider, Devu Manikantan Shila, and Marten van Dijk cover the idea of an additional layer of security for embedded devices (p. 80).

C. J. Abate
cabate@circuitcellar.com

Issue 297 April 2015 | ISSN 1528-0608

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

Circuit Cellar, Inc.
111 Founders Plaza, Suite 904
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 904
East Hartford, CT 06108, E-mail: newproducts@circuitcellar.com

HEAD OFFICE

Circuit Cellar, Inc. 111 Founders Plaza, Suite 904
East Hartford, CT 06108
Phone: 860.289.0800

COVER PHOTOGRAPHY

Chris Rakoczy, www.rakoczyphoto.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 Plance), 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
Debbie Lavoie

OUR NETWORK

circuit cellar

audioxpress

VOICE COIL

SUPPORTING COMPANIES

Accutrace	7	Ironwood Electronics	79
All Electronics Corp.	79	Jeffery Kerr, LLC	79
Custom Computer Services	79	Lauterbach GmbH	25
Elprotronic, Inc.	31	MaxBotix, Inc.	79
EMAC, Inc.	39	MyRO Electronic Control Devices, Inc.	79
ESC 2015 - Boston	65	NetBurner, Inc.	1, 15
ESC 2015 - Silicon Valley	49	Pico Technology, Ltd.	C2
HuMANDATA, Ltd.	33	Saelig Co., Inc.	31
IAR Systems	21	Scidyne Corp.	79
Imagineering, Inc.	C4	Technologic Systems	27, 45

NOT A SUPPORTING COMPANY YET?

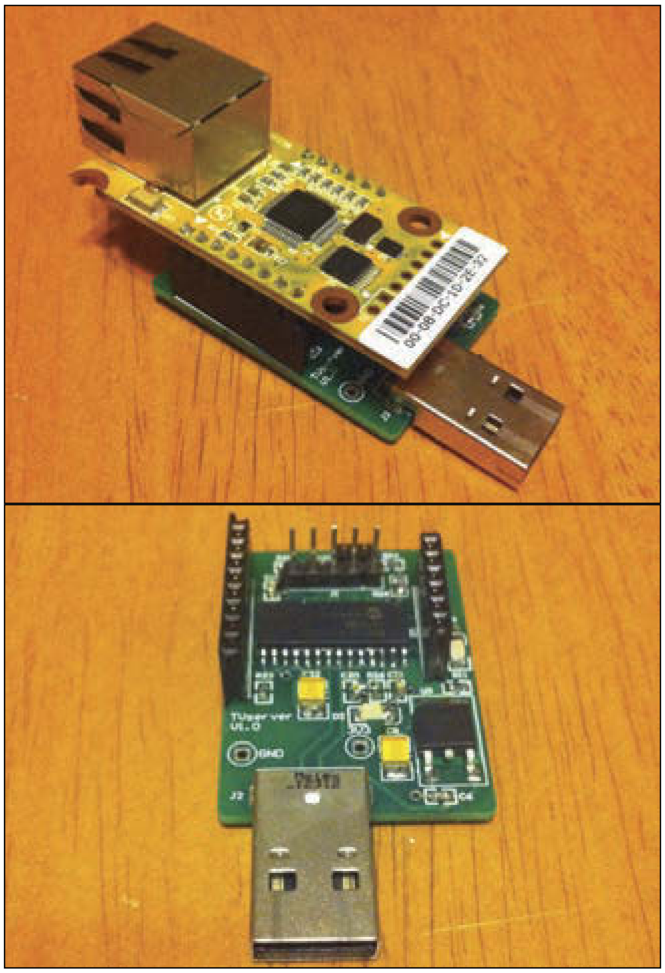
Contact Peter Wostrel (circuitcellar@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706) to reserve your own space for the next edition of our members' magazine.

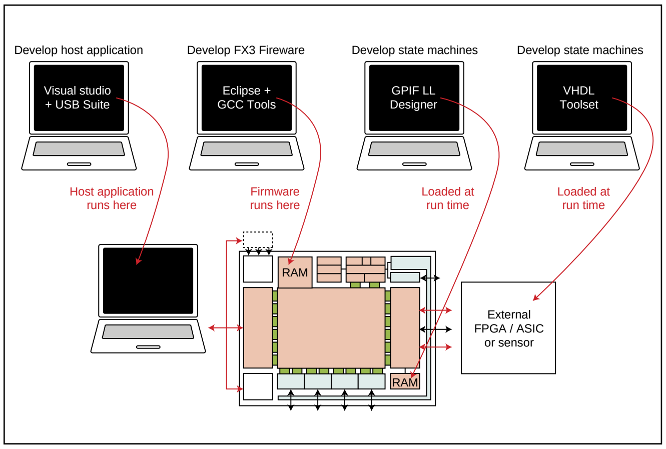
CONTENTS

APRIL 2015 • ISSUE 297

EMBEDDED PROGRAMMING

circuit cellar

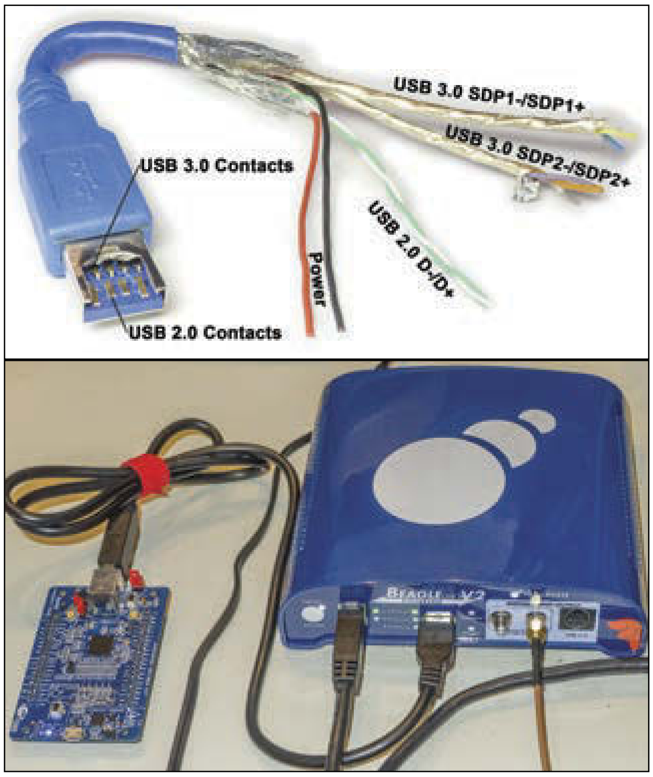

LCDTV SERVER PROJECT



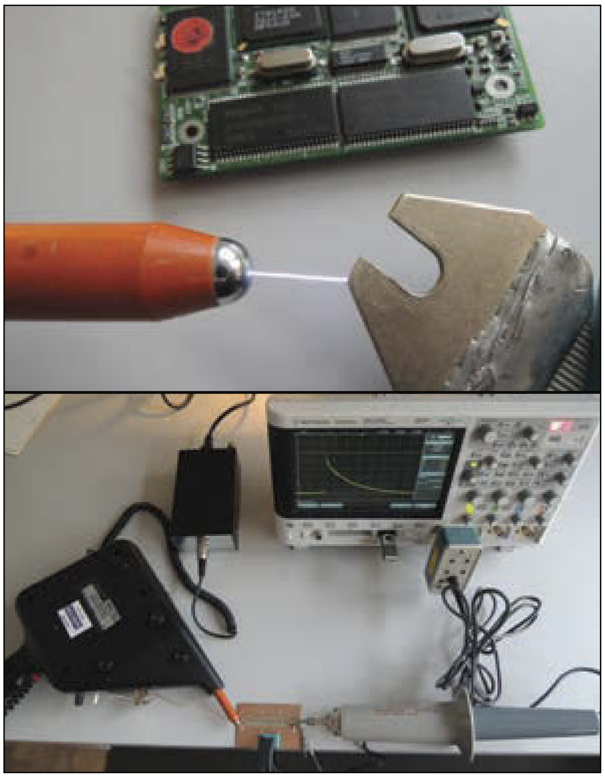
DEVELOPMENT OF A SUPERSPEED DEVICE

BUILD A QUAD BENCH POWER SUPPLY

CONTENTS



WORKING WITH SUPERSPEED USB



HOW TO MANAGE ELECTROSTATIC DISCHARGE

@editor_cc
@circuitcellar
circuitcellar

QUESTIONS & ANSWERS

The Internet of Things
A Very Disruptive Force
An Interview with Geoff Lees

Secure, embedded processing solutions
for the Internet of Tomorrow
Health & Wearables

We met with Geoff Lees (Senior Vice President & General Manager of Microcontrollers, Freescale) at the 2015 Embedded World Show in Nuremberg, Germany. We asked him about the Internet of Things, the big changes on the embedded systems horizon, and what it takes to be a successful engineer.

CIRCUIT CELLAR: A bit of business first. Could first comment on Freescale's success in the market? The company's stock is doing pretty well.

GEOFF: Yes, we don't know more than everyone else. I believe the high stock price is a reward for the quality of our services, products and commitment to our clients. We also have shown excellent results over the last seven or eight quarters and have been successful in addressing the volume markets more efficiently.

CIRCUIT CELLAR: The Embedded World Show is one of the biggest that specifically focuses on embedded technologies, new products, and design. What makes this show special?

GEOFF: In Europe we go to the Electronica in Munich and also to this show. At Electronica, we meet up with our clients and distributors. This Embedded show has a much more technical focus. Here we meet with the individual designers and technical teams of our clients and see most in-depth technical discussions. At the Electronica show we

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
2 Layers
$3 ea.
Accutrace
?
?

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), Polymide, Rogers, Isola, etc.
- HDI Capabilities: Blind/Buried Mocrovias, 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 sales@PCB4u.com

"The IoT is a very disruptive force. It started out as a buzz, but it is in the 'nature' of microcontrollers to connect and to communicate."

talk business; here we talk more technology and what it can do for the client.

CIRCUIT CELLAR: Talking about individual engineers, we remember last year in your press conference you mentioned a focus on hobbyists. That's quite remarkable for a company like Freescale. We also see there is a small "maker lab" in your booth at the show.

GEOFF: It is important to address makers and hobbyists for two reasons. First, there are the sheer numbers. At Maker Show in New York, you see there a 100,000 people showing up. At a show like this, it is 20,000 to 25,000. Here we see the engineering teams of companies. But what is interesting about the maker community is that individuals can have an idea or innovation, create and build the prototypes, but instead of having a company making this, they have the community and can even go to market.

CIRCUIT CELLAR: Sometimes we get the idea that the bigger companies are looking at the crowd-funding communities as part of—or as a replacement for—their own R&D activities. How does that work for a company like Freescale?

GEOFF: One thing that's very clear in today's world is speed. Sometimes an individual, with very little obstruction, can have speed that cannot be matched by companies—and someone who can respond or react to the requirements almost instantaneously has an advantage. There are so many of these. Finding and communicating with them is almost an impossible task. You really have to watch carefully. It is almost impossible to know where the next innovation is coming from.

CIRCUIT CELLAR: You call the Internet of Things, the Internet of Tomorrow?

GEOFF: The IoT is a very disruptive force. It started out as a buzz, but it is in the "nature" of microcontrollers to connect and to communicate. With new Wi-Fi concepts, low-power and IPv6 the road is clear for many new applications. To demonstrate the new technologies we have a "bigger than big" truck driving through the US. We put it in the parking lot of companies and demo not only our own products, but also their products as well as the solutions of their competitors. With a show you get the designers or marketing people. With the truck we also have CEOs and CTOs for a coffee—the guys who would not even consider visiting our website!

CIRCUIT CELLAR: But how will the IoT affect us?

GEOFF: I currently have eight apps on my phone that are all IoT controls, monitoring my house, solar panels, and vehicle. I expect that number will grow. Also, devices will talk to devices and create new independent controls. "Big Ass Fans" is a nice example of that. That company is making fans but is also playing a role in home automation. Their latest model fan talks to the NEST. Only a small difference in temperature can set the fan to work rather than your air conditioning, either by cooling down or circulating the hotter air downwards.

CIRCUIT CELLAR: Everyone knows that standards are key to making the IoT really happen. What role does Freescale have in this?

GEOFF: We joined up with the Thread Group. This initiative started with only eight companies, and that number has grown to 50 in five months, and now we see around 1,000 companies that look for information. If we see a growth from eight to 50 to 1,000, you know that there is a momentum which will result in new standards. The Thread Group uses existing (IEEE 8082.15.4) technologies and standards to build a new wireless mesh protocol that will enable to overcome the current limitations in wireless home automation. The Thread Networks will aim at the simple installation of new nodes and it can scale up to 250 and more devices in a single network. No company—whether you are Cisco or IBM or Oracle—has the power to set the standard on their own, maybe a part of it, but not all. This will go as usual, an initiative will gain critical mass, and then the momentum drives it through. This will all be about momentum.

CIRCUIT CELLAR: Apart from the standards, what do we need on the technology side to make the Internet of Tomorrow?

GEOFF: Let me give an example. In last year's first quarter, Simon Segars, CEO of ARM, announced that they shipped 50 billion ARM cores in their partners' products since the start of the company in 1990. That is fascinating, and even more fascinating if you realize that 10 billion were shipped in the last 10 months. The ARM technology is one of the big driving forces. I hear everyone is explaining that the microcontroller market has matured. I would say the next five years we will see many things happening. ARM, integrated Wi-Fi, and IPv6 will play key roles in this.

CIRCUIT CELLAR: Are you an engineer?

GEOFF: Yes. I started out with programming Motorola architectures almost 35 years ago. I was using the first CMOS 6805 controllers and also worked with other architectures from Intel and Zilog. I ended up in distribution as an application engineer and in applications management. By then microcontrollers had become a phenomenal business success and the applications were becoming more and more interesting.

CIRCUIT CELLAR: You've worked for several top companies: Siemens, Philips, NXP, and now for Freescale. What were the biggest lessons you learned? And what makes an engineer stand out these days?

GEOFF: I think the emergence of the ARM architecture and the low-power architecture really was a trendsetter. I was involved in bringing the ARM7TDMI with Integrated Flash to the market, and that really changed everything. But also after that, being able to explore the Thumb Instruction set and work with ARM on that. The other big change I noticed is the miniaturization. The designs I worked on in the 1980s did not fit on one board. It took several boards and additional analog boards. I worked on projects with seven processor boards, and now we have all the functionality in only one device! So miniaturization and low power have been the most remarkable for me as an engineer. You ask me what makes an engineer stand out? Never stand still, always follow the path of technology, always look for effective use and how to get a successful product. As soon you stop doing this, the technology stagnates, and for me that is something I never want to see.

THE MAKING OF A SILICON POWERHOUSE

Soon after this interview was finished, NXP Semiconductors announced it would acquire Freescale. The combination of these two companies will create a new silicon powerhouse. Both companies have strong focuses on the mobile and automotive market, which are two areas where we can expect significant growth in the coming years.

EDITORS' PICKS

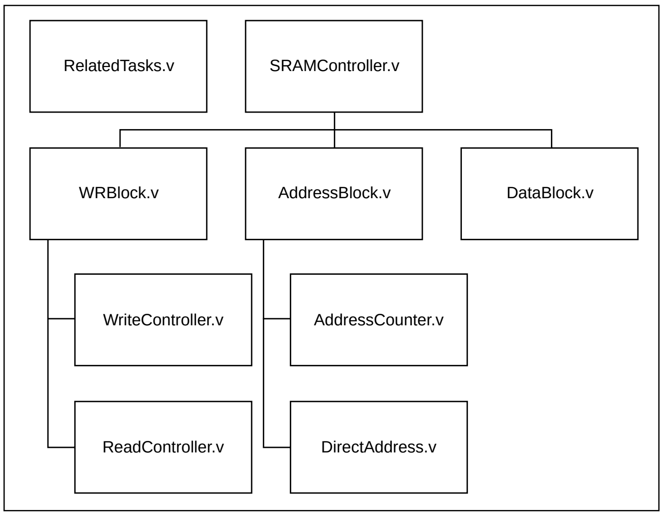
Embedded Programming

An Introduction to Verilog

By Kareem Matariyeh (Circuit Cellar 221, 2008)

If you are new to programming FPGAs and CPLDs or looking for a new design language, Kareem Matariyeh has the solution for you. In this article, he introduces you to Verilog. Although the hardware description language has been used in the ASIC industry for years, it has all the tools to help you implement complex designs, such as a creating a VGA interface or writing to an Ethernet controller.

Matariyeh writes: "This article is mostly tailored to engineers who need to learn Verilog and do not know or know little about the language. Those who know VHDL will benefit from reading this article as well and should be able to pick up Verilog fairly quickly after reviewing the example listings and reading content from the Resources at the end of the article. This article does not go over hardware, but I have included some links that will help you learn more about how the hardware interacts with this language at the end."



This is how Verilog projects are depicted from a top-level design view.

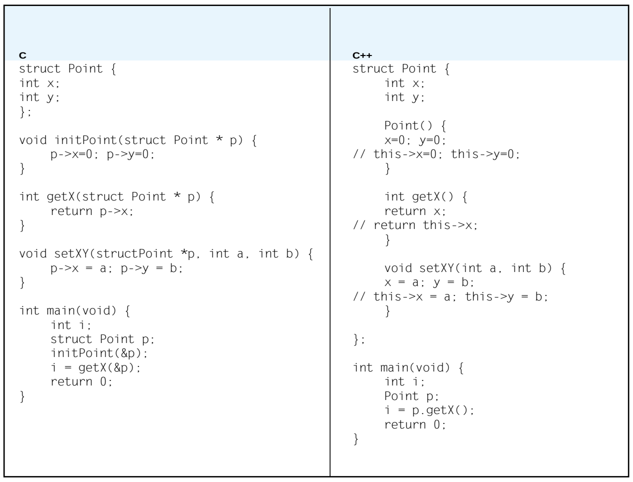
These articles and others on topics relating to Embedded Programming are available at www.cc-webshop.com.

Embedded Object-Oriented Programming

By Chris Cantrell (Circuit Cellar 187, 2006)

To be an effective software engineer, you must have the right tools on hand for generating top-notch code. Object-oriented programming is one such tool. Chris Cantrell explains how to use object-oriented programming to take your embedded designs to the next level.

Cantrell writes: "Object-oriented programming isn't one big tool. It's actually a collection of smaller tools that work together. You're free to pick and choose your object-oriented features and your degree of commitment. In this article, I'll cover five major themes and describe how they have evolved from styles in C to automatic features in C++. Each feature has its benefits and costs. I'll discuss the pros and cons of each so you can clearly decide when to use each. Where possible, I will measure the runtime performance cost of a feature by timing code running on a Game Boy Advance. After familiarizing you with these object-oriented features, I'll explain how they're the embedded world."



C

```
struct Point {
int x;
int y;
};

void initPoint(struct Point * p) {
    p->x=0; p->y=0;
}

int getX(struct Point * p) {
    return p->x;
}

void setXY(structPoint *p, int a, int b) {
    p->x = a; p->y = b;
}

int main(void) {
    int i;
    struct Point p;
    initPoint(&p);
    i = getX(&p);
    return 0;
}
```

C++

```
struct Point {
    int x;
    int y;

    Point() {
    x=0; y=0;
// this->x=0; this->y=0;
    }

    int getX() {
    return x;
// return this->x;
    }

    void setXY(int a, int b) {
    x = a; y = b;
// this->x = a; this->y = b;
    }

};

int main(void) {
    int i;
    Point p;
    i = p.getX();
    return 0;
}
```

Functions and data go hand in hand. In C language (on the left), functions that manipulate a structure's data must be passed a pointer to the target structure. In C++ (on the right), you forge this relationship automatically by putting the functions inside the structure's definition. The resulting binary code is identical..

Microprocessor Glue Logic with Verilog HDL

By Mark Balch (Circuit Cellar 158, 2003)

For reasons of availability and practicality, you may soon find it necessary to design custom logic for your digital projects. There are various design techniques to choose from, but you'll want one that suits your specific needs. Mark Balch explains how Verilog HDL may prove to be the perfect solution for your more complex digital designs.

Balch writes: "After you decide to implement logic within a PLD, you'll need a design methodology to move ahead and solve the problem at hand. It is possible to use the same design techniques as those used for discrete 7400 logic implementations. The trouble with graphical logic representations is that they are bulky and prone to human error. Hardware description languages (HDL) were developed to ease the implementation of more complex digital designs by representing logic with high-level semantic constructs found in mainstream computer programming languages. One of the major HDLs in use today is Verilog, which began as a proprietary product and was eventually transformed into an open standard."

```
module my_logic (
  A, B, C, Y
);

input A, B, C;
output Y;

wire and1_out, and2_out,
notA;

and_gate u_and1 (
  .in1 (A),
  .in2 (B),
  .out (and1_out)
);

not_gate u_not (
  .in  (A),
  .out (notA)
);

and_gate u_and2 (
  .in1 (notA),
  .in2 (C),
  .out (and2_out)
);

or_gate u_or (
  .in1 (and1_out),
  .in2 (and2_out),
  .out (Y)
);

endmodule
```

Verilog gate, or instance, level design entails the manual connection of logical entities in a netlist-like form. A typical design uses this style for interconnecting hierarchal blocks rather than actually creating Boolean equations.

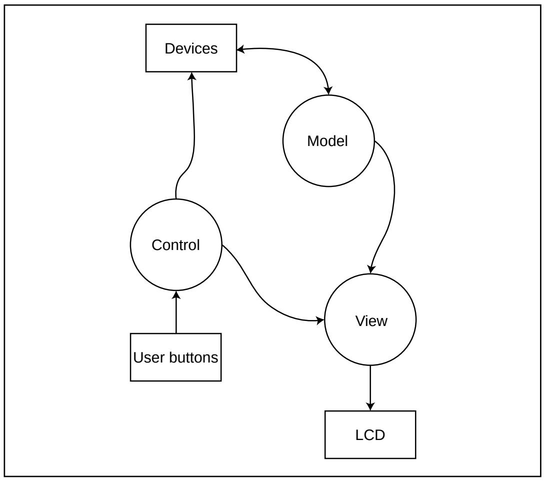
Control Center Software Design

By Scott Weber (Circuit Cellar 273, 2013)

A collection of nearly autonomous microcontroller-based, RS-485 interconnected devices was a useful addition to this designer's home. But it also came with inconveniences. What if the devices needed updates or quality checks? A simple control panel solved the dilemma. This article describes the control panel's basic structure and its Model View-Controller (MVC) paradigm, which is used to organize the software functionality.

Weber writes: "I built a simple control panel that displays to a four-line LCD module, showing me the information I want to see. The control panel also enables me or anyone else to control the devices. In this article, I will describe the control panel's basic structure and go into depth about the software paradigm referred to as the 'Model-View-Controller (MVC).' This paradigm is used to organize software functionality into a structure that is easier to develop, maintain, and enhance. It takes more ROM to handle it, but in some circumstances, the trade-off is worth it. I figured this is one of those cases."

The completed control panel includes a three-gang wall box. Scott used a nibbling tool to cut out the display's bezel.



In this MVC interaction, the user input is read by the controller and used to send commands into the network and change the view. The model information is passed to the view so it can display selected information to the user.

These articles are available at www.cc-webshop.com.

PRODUCT NEWS

NEW MOTION MODULE FOR EASY MOTION MONITORING

Microchip Technology announced at the Embedded World conference in Germany the MM7150 Motion Module, which combines Microchip's SSC7150 motion co-processor combined with nine-axis sensors. Included in compact form factor are an accelerometer, magnetometer, and gyroscope. With a simple I2C connection to most MCUs/MPUs, embedded applications and Internet of Things (IoT) systems can easily tap into the module's advanced motion and position data.

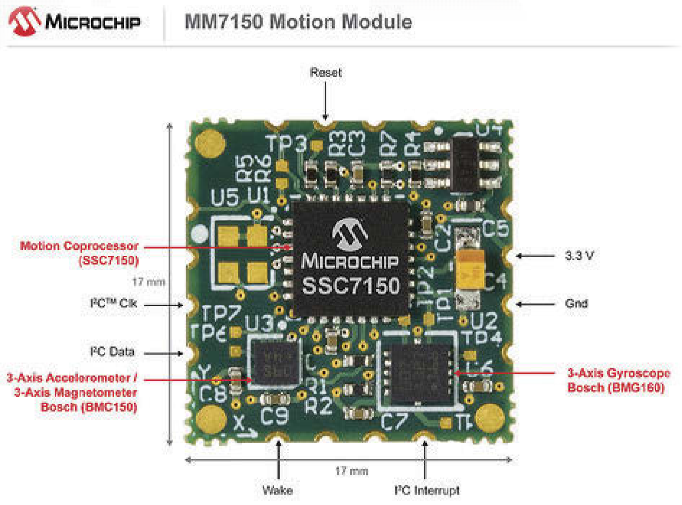
The SSC7150 motion co-processor is preprogrammed with sensor fusion algorithms that intelligently filter, compensate, and combine the raw sensor data to provide highly accurate position and orientation information. The small module self-calibrates during operation utilizing data from the prepopulated sensors—Bosch BMC150 (six-axis digital compass) and the BMG160 (three-axis gyroscope).

The single-sided MM7150 motion module is easily soldered down during the manufacturing process. You can develop motion applications for a variety of products with Microchip's MM7150 PICtail Plus Daughter Board. The MM7150 Motion Module is well suited for a wide range of applications: embedded (e.g., portable devices and robotics), industrial (e.g., commercial trucks, industrial automation, and patient tracking), and consumer electronics (e.g., IoT, remote controls, and wearable devices).

The MM7150 is supported by the MM7150 PICtail Plus Daughter Board (AC243007, $50) that plugs directly into Microchip's Explorer 16 Development Board (DM240001, $129) to enable quick and easy prototyping utilizing Microchip's extensive installed base of PIC microcontrollers.

The 17 mm × 17 mm MM7150 is priced at $26.76 each in 1,000-unit quantities.

Microchip Technology
www.microchip.com


MICROCHIP
MM7150 Motion Module
Reset
Motion Coprocessor (SSC7150)
17 mm
I²C™ Clk
I²C Data
3-Axis Accelerometer / 3-Axis Magnetometer Bosch (BMC150)
3.3 V
Gnd
3-Axis Gyroscope Bosch (BMG160)
17 mm
Wake
I²C Interrupt

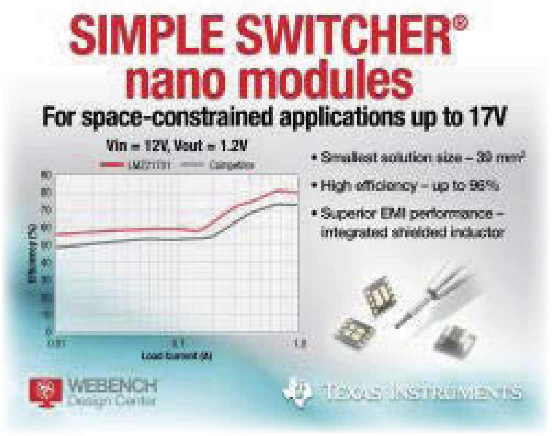
SIMPLE SWITCHER NANO MODULES

Texas Instruments has introduced four new SIMPLE SWITCHER nano power modules for space-constrained applications. The compact 17- and 5-V modules expand TI's SIMPLE SWITCHER module portfolio to address 100-mA to 2-A industrial designs, such as servers, factory automation, test and measurement, and network security cameras.

TI's 17-V, 0.65-A LMZ21700 and 1-A LMZ21701—as well as the 5-V, 1-A LMZ20501 and 2-A LMZ20502 DC/DC power modules—achieve an overall solution size of up to 40% smaller than a discrete implementation. The modules combine high efficiency with high density and reduce EMI, even while operating at low power. All four modules enable designers to easily add more features and functionality to their systems in a smaller form factor, while speeding time to market.Watch a demonstration on how to create a high-density, multi-output design.

Key features and benefits:

- Small solution sizes reduces board space by 40% when compared to discrete solutions.
- Low component count simplifies design and increases system reliability.
- Modules provide effective power management over the entire operating range.
- Low output ripple at less than 10 mVPP for noise sensitive rails.
- Low EMI complies with the CISPR 22 (Class B) radiated and conducted electromagnetic interference standard.
- Modules enable easy implementation of multiple power rail sequencing using Power Good pin.



The four nano modules are available now in volume production. The LMZ21700 and LMZ21701 cost $1.55 and $1.75, respectively, in 1,000-unit quantities. The LMZ20501 and LMZ20502 cost $1.55 and $1.90, respectively, in 1,000-unit quantities.

Texas Instruments
www.ti.com

TWO SOURCE/MEASURE UNITS FOR N6700 MODULAR POWER SYSTEMS

Keysight Technologies recently added two source/measure units (SMUs) to its N6700 Series modular power systems. The N6785A two-quadrant SMU is for battery drain analysis. The N6786A two-quadrant SMU is for functional test. Both SMUs provide power output up to 80 W.

The two new SMUs expand the popular N6780A Series SMU family by offering up to 4× more power than the previous models. The new models offer superior sourcing, measurement, and analysis so engineers can deliver the best possible battery life in their devices. The N6785A and N6786A SMUs allow engineers to test devices that require current up to 8 A, such as tablets, large smartphones, police/military hand-held radios, and components of these devices.

The N6780A Series SMUs eliminate the challenges of measuring dynamic currents with a feature called seamless measurement ranging. With seamless measurement ranging, engineers can precisely measure dynamic currents without any glitches or disruptions to the measurement. As the current drawn by the device under test (DUT) changes, the SMU automatically detects the change and switches to the current measurement range that will return the most precise measurement.

When combined with the SMU's built-in 18-bit digitizer, seamless measurement ranging enables unprecedented effective vertical resolution of ~28-bits. This capability lets users visualize current drain from nA to A in one pass. All data needed is presented in a single picture, which helps users unlock insights to deliver exceptional battery life.

The new SMUs are a part of the N6700 modular power system, which consists of the N6700 low-profile mainframes for ATE applications and the N6705B DC power analyzer mainframe for R&D. The product family has four mainframes and more than 30 DC power modules, providing a complete spectrum of solutions, from R&D through design validation and manufacturing.

Keysight Technologies
www.keysight.com

NEW IoT-ENABLED PRODCUCT PORTFOLIO &SEVERCES

Wind River recently announced that it has enhanced and expanded its Wind River Helix product portfolio to address the system-level opportunities and challenges of the Internet of Things (IoT). In addition, the company has created an IoT professional services offering to assist customers with the creation and deployment of IoT apps.

Wind River has added application and data services in the cloud to its industry-leading operating-based technology stack that is an integral part of the Intel IoT Platform. The Edge Management System agent has been integrated with VxWorks real-time operating system (RTOS), systems and IoT software platform via Wind River Edge Management System, its recently launched cloud Wind River Linux, and Wind River Intelligent Device Platform.

WIND RIVER
HELIX
DEVICES
NETWORK INFRASTRUCTURE
DATA CENTER/ CLOUD

The agents bring secure cloud connectivity to Wind River products to facilitate data capture, rules-based data analysis and response, configuration, and file transfer. Specifically, these integrations provide device-level execution capabilities, remote management and provisioning capabilities at the gateway, as well as cloud-based delivery of software updates. This allows for seamless interaction with edge devices and simplified device-side application development.

To complement its new IoT-enabled product portfolio, Wind River now has an IoT professional services offering to bring IoT concepts to critical infrastructure and other markets where safety and security are imperatives. The new offering will assist customers in configuring IoT systems and getting them to market faster with reduced risk and lower cost of ownership. Services include an IoT startup package, device agent configuration, application/agent interfacing, cloud applications development, and IoT safety and security requirements support.

Further expanding its operating system suite, Wind River has also announced the availability of Microkernel Profile for VxWorks. The microkernel profile is a tiny-footprint RTOS to facilitate the creation of IoT-ready differentiated devices, such as sensor hubs, microcontrollers, and wearables, as well as High Performance Embedded Computing (HPEC) platforms to address intensive data processing. It is based on proven digital signal processing RTOS technology deployed in countless applications.

These product additions and enhancements are the latest in a series of IoT-related updates to the company's operating system suite, which include Security Profile for VxWorks, Virtualization Profile for VxWorks, and Security Profile for Wind River Linux.

Wind River
www.windriver.com

PRODUCT NEWS

4-PLL GENERATORS FOR NEXT-GEN CONSUMER & NETWORKING PRODUCTS

Cypress Semiconductor Corp. has announced a new high-performance programmable clock generator family that's intended to simplify the design of consumer and networking systems. The new CY27410 4-PLL (phase-locked loop) clock generator can generate up to 12 programmable output frequencies on a single chip with superior jitter performance. The clock reduces both board space and BOM costs by consolidating system components to provide a flexible, low-cost solution.

With support for frequencies up to 700 MHz and RMS phase jitter of 0.7 ps, the CY27410 family supports reference clocks for PCIe 1.0/2.0/3.0, SATA1.0/2.0, 10GbE, and USB1.0/2.0/3.0 peripherals. The devices support on-board programming using I2C interface, adding to design flexibility. They can also store up to eight different configuration settings that are selectable using external digital control pins.

The family supports 12 single-ended clock outputs, as well as eight differential output pairs that can be configured as HCSL, LVPECL, LVDS, CML, or LVCMOS outputs. The devices also integrate a unique combination of value-added features that simplify design, including VCXO, glitch-free outputs, EMI-reduction, configurability as a zero or nonzero delay buffer, early/late clocks and PLL cascading.

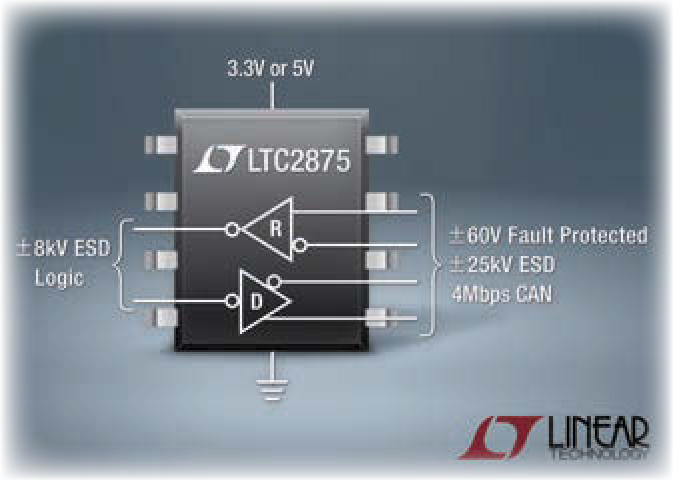
The CY27410 clocks come with the CY3679 evaluation kit and CyClockWizard 2.0 programming software to help designers create their desired frequencies and to easily check device performance.

The CY27410 clock generators are currently sampling. Production expected in Q2 2015. The devices are available in a 48-pin QFN package.

Cypress Semiconductor Corp.
www.cypress.com

3.3-V/5-V 4-Mbps CAN TRANSCEIVER

Linear Technology Corp. recently introduced the LTC2875, an exceptionally rugged, high-voltage-tolerant controller area network (CAN) transceiver to greatly reduce field failures without the need of costly external protection devices. In practical CAN systems, installation cross-wiring faults, ground voltage faults or lightning induced surge voltages can cause overvoltage conditions that exceed absolute maximum ratings of typical transceivers. The LTC2875 features ±60-V overvoltage fault and ±25-kV HBM ESD protection on the data transmission lines, protecting bus pins during operation and shutdown. Whether a circuit is transmitting, receiving or powered off, the LTC2875 tolerates any voltage within ±60 V without damage, increasing the robustness of typical CAN networks.



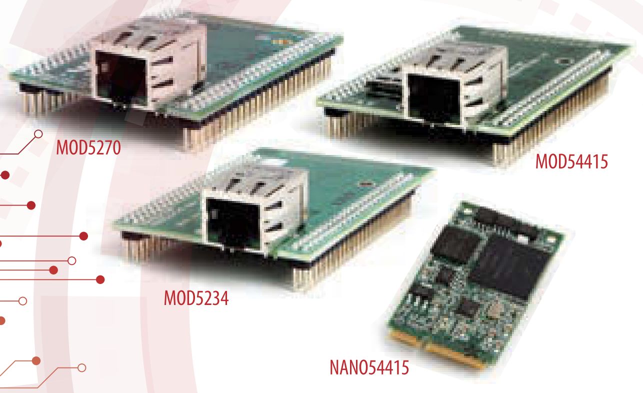
CAN bus systems are becoming increasingly popular in industrial controls, instrumentation networks and automotive electronics. The CAN bus has a well defined protocol stack, with support for standalone controllers, FPGAs and ASICs, making implementation easier over alternative interfaces, such as RS-485. The LTC2875 provides the flexibility to be powered from a 3.3-V or 5-V rail, which is very useful in industrial applications where a 5V rail may not be present. In addition to the high fault and ESD protection, the device features a low electromagnetic emission (EME) driver with a transmit data (TXD) dominant timer to prevent faulty controllers from clamping the bus, as well as a high electromagnetic immunity (EMI) receiver with an extended ±36-V common mode range to enable operation in electrically noisy environments and in the presence of ground loops. The LTC2875 features a high speed data rate of 4 Mbps with an adjustable slew rate for data rates as low as 1 kbps. A shutdown mode brings all of the LTC2875's outputs to high impedance and reduces power consumption to 1 µA.

The LTC2875 is offered in commercial, industrial, automotive and military (–55°C to 125°C) temperature grades and is available in 3 mm × 3 mm DFN-8 and SO-8 packages, with industry-standard pinouts.

Pricing starts at $1.72 each in 1,000-piece quantities.

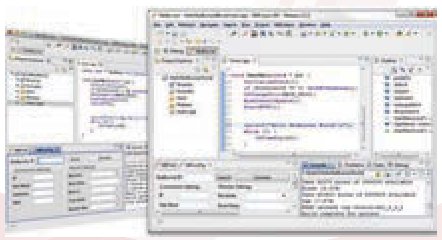
Linear Technology
www.linear.com

Ethernet Core Modules with High-Performance Connectivity Options



- **MOD5270**
  147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO
  3 UARTs · I²C · SPI
- **MOD5234**
  147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs
  I²C · 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 I²C · 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 I²C · 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

NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™ Alliance Member

GECKO BLUETOOTH SMART SOLUTIONS FOR LOW-POWER WIRELESS CONNECTIVITY

Silicon Labs today has launched a Bluetooth Smart solutions portfolio intended to minimize the energy consumption, cost, and complexity of wireless Internet of Things (IoT) designs. Silicon Labs's new Blue Gecko solutions include ultra-low-power wireless system-on-chip (SoC) devices, embedded modules, and Bluegiga's software development kit (SDK) and Bluetooth Smart software stack. Blue Gecko wireless SoCs and modules help you simplify design and speed time to market for a wide range of applications (e.g., connected home, wearable, and automotive).

Blue Gecko Solutions: Fastest On-Ramp to Ultra-Low-Power Wireless Connectivity
bluegiga
A Silicon Labs Company
SILICON LABS
Bluetooth SMART

The Blue Gecko portfolio addresses the largest, fastest-growing low-power wireless connectivity opportunity in the IoT market. It provides developers with the flexibility to begin development with modules and transition to SoCs when needed with little to no system redesign.

The first in a family of wireless SoCs optimized for IoT applications, Blue Gecko SoCs combine Silicon Labs' energy-friendly EFM32 Gecko MCU technology with an ultra-low-power Bluetooth Smart transceiver. This innovative, single-die solution provides industry-leading energy efficiency, the fastest wake-up times, superior RF sensitivity and no-compromise MCU features combined with the Bluegiga Bluetooth Smart software stack to help developers reduce system power, cost and time to market. Unlike other Bluetooth Smart IC alternatives, a Blue Gecko SoC can transmit +10 dBm or higher output power with its fully integrated power amplifier and balun, further reducing design complexity.

Blue Gecko SoCs are based on the ARM Cortex-M3 and M4 cores and offer 128- to 256-KB flash sizes and 16- to 32-KB RAM sizes. The SoCs integrate an array of low-energy peripherals as well as Silicon Labs's Peripheral Reflex System (PRS) for autonomous peripheral operation. The Blue Gecko SoC family also offers a roadmap of enhanced flash and RAM memory sizes and additional package options to meet future application needs.

Bluegiga modules based on Blue Gecko SoCs are designed to help developers accelerate time to market and reduce development costs and compliance risks by providing a precertified, plug-and-play RF design. Bluegiga Bluetooth Smart modules incorporate all features of Blue Gecko SoCs and are certified for use in all key markets including North America, Europe, Japan and South Korea. Bluegiga modules include the Bluegiga Bluetooth Smart software stack and profile toolkit and come with 256 kB flash and 32-KB RAM, providing ample available memory for onboard applications. Flexible hardware interfaces enable easy connection to a variety of peripherals and sensors, and an integrated antenna makes RF operation consistent and straightforward for the design engineer. Bluegiga Bluetooth Smart modules provide very low power operation, enabling wireless system designs to be powered from a standard 3-V coin cell battery or two AAA batteries.

Samples of Bluegiga modules based on Blue Gecko SoCs are scheduled to be available in late Q2 2015. Samples of Blue Gecko wireless SoCs are planned to be available in early Q3 in 5 mm × 5 mm QFN32 and 7 mm × 7 mm QFN48 packages. Pricing for Blue Gecko-based Bluegiga modules starts at $4.99 in 10,000-unit quantities. Blue Gecko SoC start at $0.99 in 100,000-unit quantities. The Bluegiga SDK and Bluetooth Smart software stack will be available to Silicon Labs customers at no charge.

Silicon Laboratories
www.silabs.com

M2M/IoT INTEGRATION PLATFORM

Eurotech recently announced the official release of Everyware Cloud 3.5 (EC), the M2M/IoT Integration Platform. The EC machine-to-machine (M2M) Integration Platform is intended to simplify device and data management by connecting distributed devices over secure and reliable cloud services. It enables you to connect, configure, and manage devices through the lifecycle, from deployment to retirement.

With EC 3.5 a set of new remote features is available: device configuration, device control, device provisioning, and device update. These features enable the Everyware Cloud Web Console to be the single point of administration for all connected devices, and make the Everyware Cloud REST APIs the single programmable interface to remote devices. In addition, security and reliability are enhanced with Two-Factors-Authentication and platform Health Monitoring.

By using EC, you benefit from:

- Short time to market
- Pay-as-you-go
- Open Standard based
- Flexible Deployment from public cloud, to cloud-in-a-box (Everyware Server)
- Device Enablement for effective asset management and asset maintenance
- Data Enablement
- System Enablement for reliable, scalable, secure and user-friendly platform administration
- IoT Application Enablement for simple integration with enterprise business software applications

You can try EC for free for 90 days.

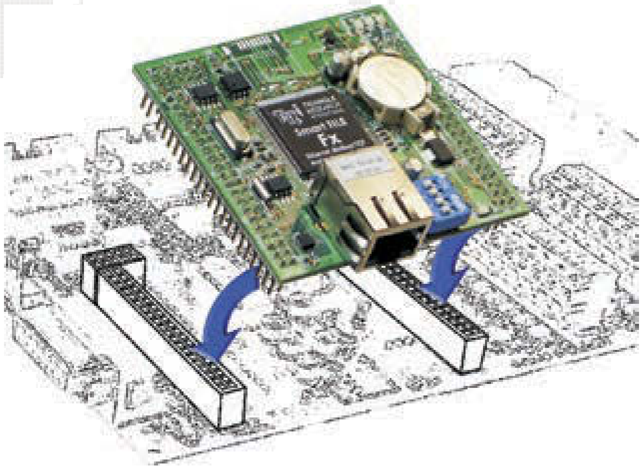
Eurotech
www.eurotech.com

CLIENT PROFILE

Triangle Research International, Inc.

www.triplc.com
11871 Horseshoe Way #2109
Richmond, BC V7A5H5, Canada

CONTACT: Wayne Lye (wayne@triplc.com)

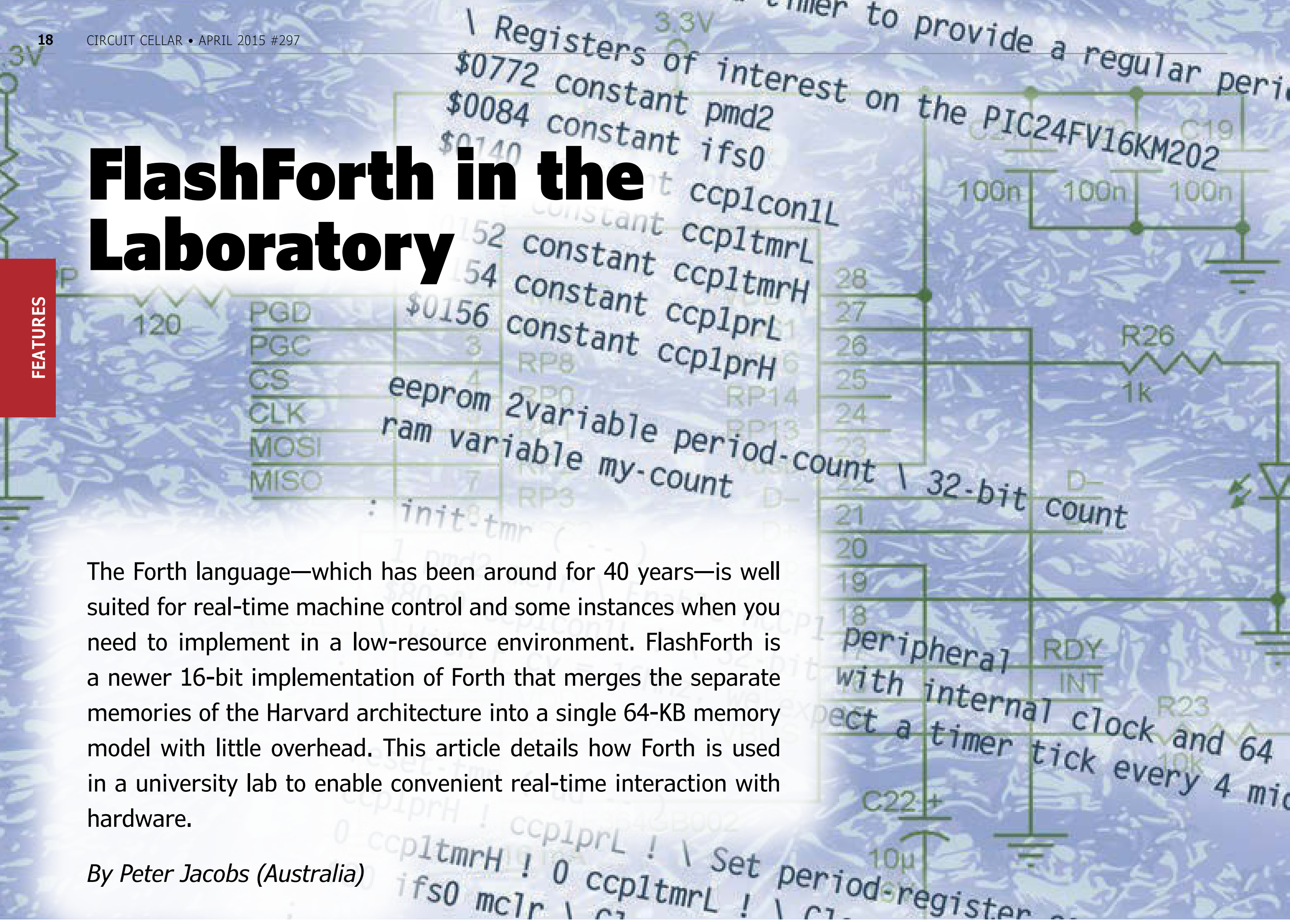


FEATURED PRODUCTS: The **FMD88-10** is one of the most popular Triangle Research Super PLCs with eight DI, eight DO, 10 analog I/Os, integrated Ethernet, RS-485/232, powerful TRiLOGI Ladder+BASIC programming, expandability. Prices start at $229 before OEM quantity discounts (www.triplc.com/fmd88-10.htm). The **SmartTILE-Fx** is for the OEM who needs an in-house custom-build PLC. It is an advanced Fx-series PLC brain-board with integrated Ethernet that easily plugs into a customer-designed carrier I/O board. It supports up to 256 DI/DO and analog I/Os including UART interface. I costs $99 to $195 depending on the quantity (www.triplc.com/smarttile.htm).

WHY SHOULD CC READERS BE INTERESTED? Compared to building a controller from scratch each time, the **SmartTILE-FX** lets OEMs custom-build their own PLC to significantly shorten time to market for all new product line releases, save money from reduced assembly man hours, eliminate unnecessary components that come with off-the-shelf PLCs, and develop a consistent standard for post-sales equipment troubleshooting, maintenance, and related training.

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.*

INDUSTRY & ENTERPRISE



FlashForth in the Laboratory

The Forth language—which has been around for 40 years—is well suited for real-time machine control and some instances when you need to implement in a low-resource environment. FlashForth is a newer 16-bit implementation of Forth that merges the separate memories of the Harvard architecture into a single 64-KB memory model with little overhead. This article details how Forth is used in a university lab to enable convenient real-time interaction with hardware.

By Peter Jacobs (Australia)

The recent arrival of the Rosetta Lander PHILAE at comet 67P/Churyumov-Gerasimenko is a good example of the Forth programming system controlling a low-resource system.[1,2] Forth, which has been in use for more than 40 years, excels at real-time machine control and is particularly convenient for low-resource microcontrollers where it can provide a fully interactive development environment on a single chip.[3] As Mahon Kelly and Nicholas Spies explain in *FORTH: A Text and Reference*, it is also a flexible programming environment, produces fast programs, is miserly with memory, and allows complete control of the microcontroller.[4]

At The University of Queensland's Hypersonics Laboratory, researchers use a number of shock tubes and expansion tubes to produce very high-speed flows for the testing of Scramjet engines and reentry aeroshells. Test flow speeds can reach 15 km per second, but are usually of very short duration, from a couple of hundred microseconds up to several milliseconds. In support of some of the experiments, we build a few bespoke monitoring, signaling, and signal-conditioning devices that have a microcontroller as a central component and FlashForth as the development environment. This combination has proven to be a convenient substrate for building small devices. With command-line access to the hardware peripherals via a serial port, the programmer can sit and interactively manipulate the register bits to experiment on new hardware and, when the peripheral is understood, the same commands may be captured as word definitions for building the final application.

BACKGROUND ON FORTH

Invented by Charles Moore, Forth is an extensible language that provides a basic structure of stacks and a dictionary.[3] It can be augmented for any particular problem. Having the stack brought to the fore in the programmer's thoughts is somewhat unusual for a programming language; however, variables are available for persistent data that needs to be accessed by multiple sections of code. Although arrays and data structures are usually not provided in the base system, it is easy to add the required words via Forth's defining words and use them as normal. Making an array-defining word is a typical text book exercise. Forth has some subtle ideas, but, as noted by Kelly and Spies, you don't need many of them to write useful programs.

Mikael Nordman's FlashForth is a fairly recent 16-bit implementation of Forth for Microchip Technology PIC and Atmel AVR microcontrollers that elegantly unifies the separate memories of the Harvard architecture into a single 64-KB memory model with very little overhead. It is available for download, licensed with the GNU GPL, and is easy to program into your choice of microcontroller. There are variants of FlashForth for PIC18, PIC24, dsPIC33, and Atmel ATmega microcontrollers. A tutorial guide for the use of FlashForth on a range of simple hardware devices is also available for download (https://espace.library.uq.edu.au/view/UQ:330707).

A TASTE OF FLASHFORTH

When trying FlashForth, the first task is to program the interpreter provided at the SourceForge project site into your microcontroller using a suitable device programmer. There is assembly source code for all variants, but there is also a preassembled binary file for the ATmega microcontrollers, such as the ATmega328 found on Arduino-style boards. Once the FlashForth interpreter is resident on your microcontroller, your principal development environment can be as simple as a terminal program running on a PC, connected via a convenient serial port or USB-to-serial bridge. For extra convenience, the terminal program should have the ability to send the content of files down the serial connection.

Forth is an interactive environment, with an input buffer that accepts text from the user console and a text interpreter that is a mix of classic interpreter and compiler, depending on its state. FlashForth starts in interpret mode and certain words can be used to change into compile mode and back again. A simplified view of the process of interpreting the text from the input buffer is shown in **Figure 1**. Words within the text are delimited by spaces. Given a new word, the interpreter searches the dictionary for its definition. If a matching definition is found, it is used, either compiled into the current

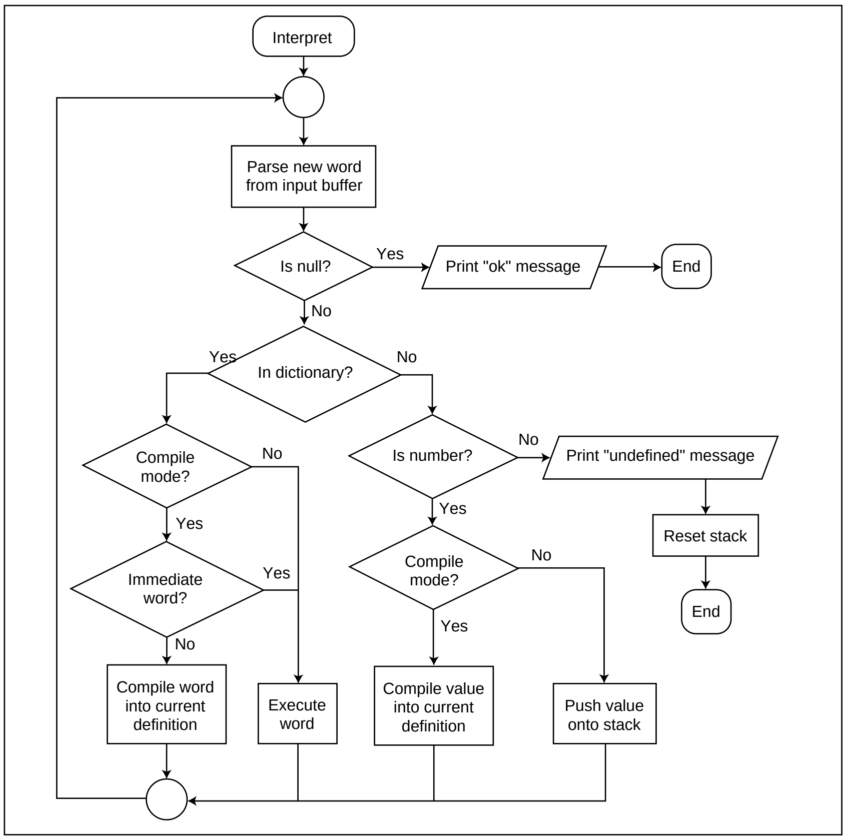


FIGURE 1
Flowchart for the FlashForth outer interpreter.

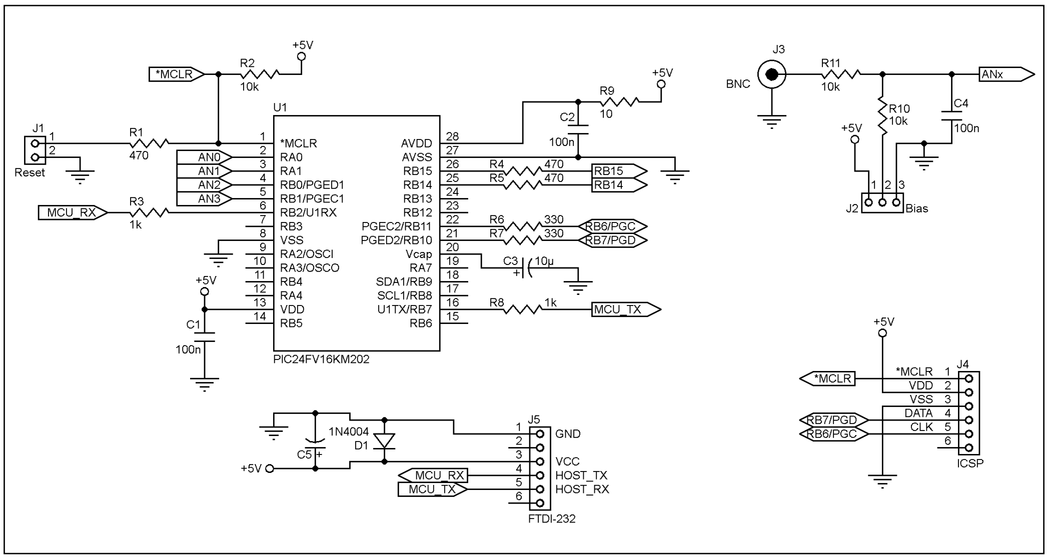


FIGURE 2
A schematic diagram for the converter board

```
\ Blink LED on pin 13 of Arduino Uno.
-blink
marker -blink

$24 constant ddrb
$25 constant portb
%100000 constant bit5

: setup
  bit5 ddrb mset \ digital output
;

: main
  setup
  begin
    bit5 portb mset \ turn LED on
    #1000 ms
    bit5 portb mclr \ turn LED off
    #1000 ms
  again
;
```

LISTING 1
Source code for the blink example.

definition or executed. If the word was not found in the dictionary, an attempt is made to interpret it as a number. If this fails, an error message is written and the stack reset. Successfully interpreted numbers are either pushed onto the data stack (interpret mode) or compiled into the current definition. For example, entering the following:

```
2 3 *
```

That is followed by a return and results in the message:

```
ok<#,ram>  6
```

This shows the classic reverse Polish notation in action. The 2 is not found in the dictionary but is identified as a number and pushed onto the top of the data stack. The 3 is likewise identified and pushed on top of the 2. Then the * token is looked up and it is found to be defined as the word for multiplication of two numbers. Since FlashForth is in interpret mode, this definition is executed by the inner interpreter, which expects to find appropriate data sitting on the data stack. In this case, the top two elements are consumed by multiply and, on completion of its work, the result 6 is left on the top of the stack. The outer interpreter then indicates that it has done its work successfully by writing the prompt ok, followed by <#,ram>, indicating that numbers are represented in decimal and that the current data memory context is static RAM and, lastly, a listing of the data stack elements.

In Forth programming, your program is written by defining new words. These words use previously defined words until, at the top-level, a single word is defined that is the main word to run your application program. In this way, your application is an extension of Forth. Most words are created as colon definitions (starting with : and ending with ;) that are compiled into the dictionary. The program can be compiled as a series of small pieces that can each be tested interactively. This then allows convenient exploration of new hardware via experimentation, something that we can put to good use in the teaching laboratories, as well.

For example, let's make a word to compute a circle's circumference from its diameter.

```
: circumf ( n1 -- n2 ) 355 113 */ ;
```

Our definition makes use of the fact that 355/113 is a pretty good approximation to pi. It is compiled into the dictionary as the word circumf and is now available for use. The stack-effect comment (between parentheses) indicates that, when executed, circumf will expect the value for diameter to be sitting as the top element of the data stack. When executed, the numbers 355 and 113 will be pushed onto the stack after n1, then the scale operation */ will take all three 16-bit integer values from the stack and compute the 16-bit result. To avoid loss of precision, the scale operator uses a 32-bit intermediate representation as the computation is being done. The stack effect comment also indicates that the 16-bit result will be left on the stack. Note that the spaces around the opening parenthesis of the stack effect comment are significant. Without them, the comment would not be identified as such, the token circumf(n1 would be considered a valid name and we wouldn't have ended up with the definition as we intended. Anyway, exercising our new definition, we might try:

```
301 circumf  ok<#,ram> 945
```

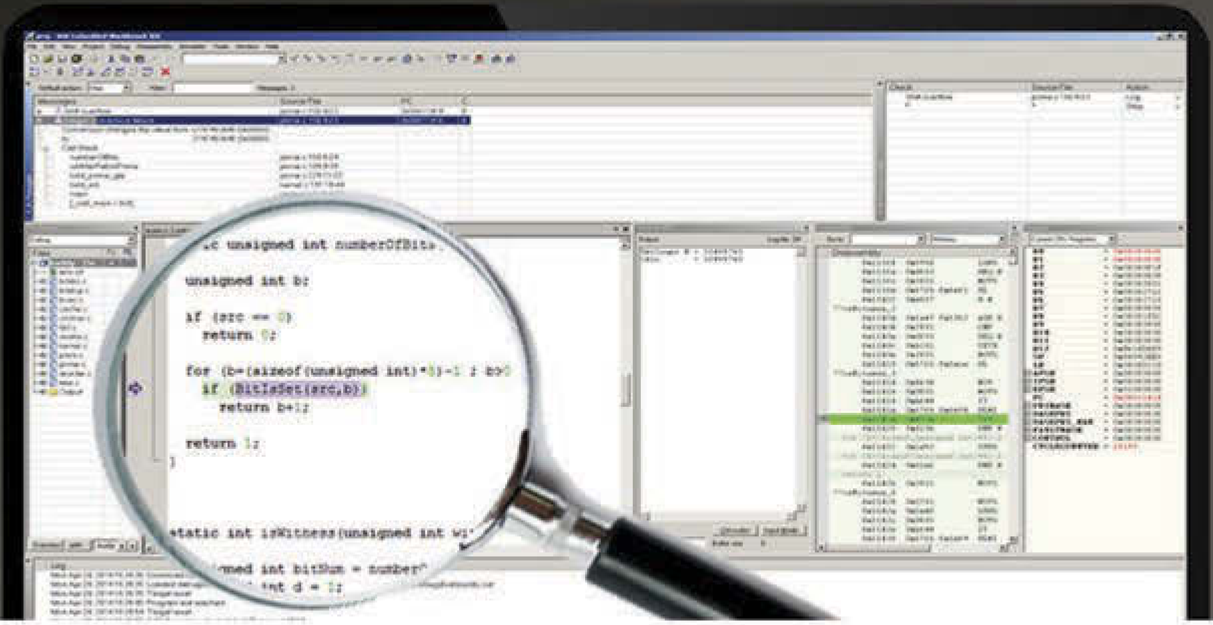
If we were using floating point numbers, we might expect a result of around 945.6 but the integer arithmetic has given us the truncated value of 945. Although all of the

ABOUT THE AUTHOR

Peter Jacobs (peterj@mech.uq.edu.au) is a lecturer in the School of Mechanical and Mining Engineering at The University of Queensland, where he teaches students in the mechanical and mechatronic streams. His interests in computing range from small sensor systems based on microcontrollers to large simulations of high speed gas flows on cluster computers.

EXPLORE C-RUN FOR ARM

Runtime Analysis Simplified



```
ic unsigned int numberOfBits

unsigned int b;

if (src == 0)
  return 0;

for (b=(sizeof(unsigned int)*8)-1 ; b>0
  if (BitIsSet(src,b))
    return b+1;

return 1;
}

static int isWitness(unsigned int w

  gned int bitNum = number
  int d = 1;
```

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

IAR SYSTEMS

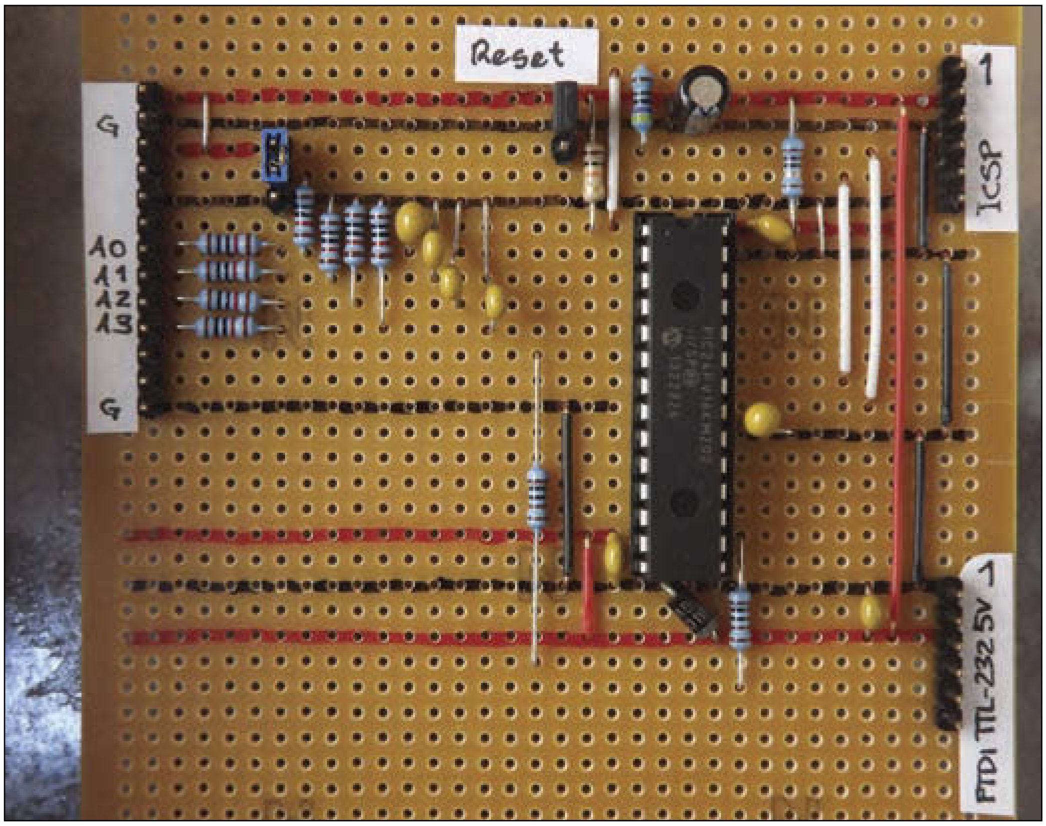


PHOTO 1
PIC24FV16KM202-I/SP microcontroller mounted on strip-board with signal conditioning for channels A0 through A3.

numbers put into and received from the definition have been adequately represented as 16-bit values, this example has benefited from the scale operator using a 32-bit intermediate result.

So far, this is all a little too much like interaction with a standard personal computer. I started off this article by saying that Forth excels in the programming of resource-limited systems, so let's have a look at the classic blink-LED example for Arduino so we get a feel for how FlashForth interacts with hardware. The FlashForth binary, as provided for download on the FlashForth web site, can be easily programmed into the Atmel ATmega328 microcontroller on the SparkFun RedBoard.

Starting with a fresh installation, the code in **Listing 1** can be typed in or sent as a raw file via the serial port connection. Here, the first three lines are just a bit of house-keeping that make our lives a bit easier. The first is a single-line comment, introduced by a backslash character, and this is purely an aid for my memory. We'll get to the second line shortly. The third line marks the state of the dictionary, such that the next time the `-blink` word is executed, the dictionary state and data-memory allocations will be reset to their state before the mark was made. On the first occasion that this file is loaded, `-blink` (to be executed on the second line) is not found in the dictionary, and the interpreter will just say so. However, on subsequent reloads of the same file, the execution of this word will reset the dictionary and memory allocations, so that the following definitions in the file can be remade. FlashForth is a little different from PC-based Forths that run fully in RAM. FlashForth definitions are put into nonvolatile program (flash) memory and cannot be accidentally overwritten. They must be explicitly removed from program memory and the marker mechanism is one way of

PHOTO 2
Boxed converter board with BNC connectors for the analog signals.

doing that. To completely reset the dictionary and memory allocations, back to the default values of a fresh FlashForth install, we can use the word `empty`.

The Arduino-like boards typically have a light-emitting diode attached to pin 13 and the RedBoard is no exception. This pin is controlled by bit 5 on digital port B. To make the rest of the program a little more readable, the next three lines of code define constants `ddrb`, `portb`, and `bit5` for later use. Note the use of `$` and `%` prefixes to indicate hexadecimal and binary number representations, respectively. The `setup` and `main` words have been defined to model the corresponding `setup` and `loop` functions in the original Arduino tutorial code.

The `setup` word in Listing 1 uses the `bit5` bit-mask and ddrb data-direction register address to set the relevant bit for output. The `mset` word expects a bit-mask and a memory address to be sitting on the stack. When executed, it sets the bits at the specified memory location to the corresponding nonzero bits of the bit-mask. Here `bit5` has only one nonzero bit.

The `main` word starts by invoking setup and then enters a loop, starting with `begin` and ending with `again`. The body of the loop first turns the LED on by setting the appropriate bit in the `portb` register, waits 1 second, turns the LED off by clearing the same bit in the `portb` register, and waits another second.

Ignoring the reverse Polish notation of Forth, there is a close correspondence between the FlashForth code and the Arduino code. The advantage of the FlashForth environment is that it is fully interactive and resides completely on the microcontroller. You are free to interact with the bits of just about any register and explore the microcontroller's hardware. Once you understand the hardware, particular actions may be defined into useful words and retained for future use. Further reading on FlashForth the language is available for download (https://espace.library.uq.edu.au/view/UQ:321883). Although I concentrate on the simplest FlashForth variant for the PIC24 and AVR microcontrollers in this article, there is built-in USB capability for various PIC18 microcontrollers.

PERIODIC A/D CONVERSION

For a class laboratory exercise recently conducted by members of the Hypersonics Laboratory, we needed to quickly assemble a three-channel 12-bit ADC as part of a color-ratio pyrometer. The converter needed to report voltages at the rate of a couple of times per second. The period between samples was somewhat arbitrary but needed to be known

```
-my-adc
marker -my-adc
\ Read and report the analog value on AN0 through AN3.

\ Registers of interest on the PIC24FV16KM202
$0770 constant pmd1
$02c0 constant trisa
$02c8 constant trisb
$0300 constant adc1buf0
$0340 constant ad1con1
$0342 constant ad1con2
$0344 constant ad1con3
$0348 constant ad1chs
$04e0 constant ansa
$04e2 constant ansb


: init-adc ( -- )
  1 pmd1 mclr \ enable the AD converter
  %11 trisa mset \ want RA1-RA0 as analog input
  %11 ansa mset
  %11 trisb mset \ want RB1-RB0 as analog input
  %11 ansb mset
  \ leave ad1con2 at default 0 to get AVDD-AVSS range
  $9f00 ad1con3 ! \ RC clock with 31 TAD sampling time
  $8470 ad1con1 ! \ 12-bit mode, auto-convert but
                  \ manually start sampling
;

: set-adc-chan ( u -- )
  %11 and ad1chs ! \ allow only selection of AN0 through AN3
;

: adc@ ( -- u )
  1 ad1con1 mclr \ Clear DONE
  %10 ad1con1 mset \ Start sampling
  begin 1 ad1con1 mtst until \ Wait until DONE
  adc1buf0 @
;

\ Exercise the application, writing digitized values
\ periodically until any key is pressed.
: test-adc ( -- )
  decimal
  init-adc
  begin
    #0 set-adc-chan adc@ u.
    #1 set-adc-chan adc@ u.
    #2 set-adc-chan adc@ u.
    #3 set-adc-chan adc@ u.
    cr
    #500 ms
  key? until
;
```

LISTING 2
Source code for configuring and using the analog-to-digital converter module.

```
-my-timer
marker -my-timer
\ Use MCCP1 as a timer to provide a regular period for
\ the super-loop.

\ Registers of interest on the PIC24FV16KM202
$0772 constant pmd2
$0084 constant ifs0
$0140 constant ccp1con1L
$0150 constant ccp1tmrL
$0152 constant ccp1tmrH
$0154 constant ccp1prL
$0156 constant ccp1prH

eeprom 2variable period-count \ 32-bit count
ram variable my-count

: init-tmr ( -- )
  1 pmd2 mclr \ Enable MCCP1 peripheral
  $80e0 ccp1con1L ! \ 32-bit with internal clock and 64 prescale
 \ With F_cy = 16MHz, we expect a timer tick every 4 microseconds.
;

: reset-tmr ( ud -- )
  ccp1prH ! ccp1prL ! \ Set period-register count
  0 ccp1tmrH ! 0 ccp1tmrL ! \ Clear timer count
  $80 ifs0 mclr \ Clear CCT1IF
;

: wait-for-tmr ( -- )
  begin
    cwd \ We may be here a while, so clear the WDT
    $80 ifs0 mtst \ CCT1IF
  until
;

: stop-tmr ( -- )
  $8000 ccp1con1L mclr
;

: set-ms-period ( u -- )
  #250 um*   \ Scale from milliseconds to ticks.
  period-count 2!
;

#500 set-ms-period  \ Set a 0.5 second period by default.

\ Exercise the periodic tick, until a key is pressed
: test-tmr ( -- )
  init-tmr
  0 my-count !
  begin
    period-count 2@ reset-tmr \ 32-bit value
    my-count @ dup u. 1+ my-count ! \ Print and increment
                                     \ loop count
    wait-for-tmr
  key? until
;
```

LISTING 3
Source code for using the hardware timer and compare-capture module.

accurately.

We chose a Microchip Technology PIC24FV16KM202-I/SP microcontroller to do the conversions and report the data as text because it can conveniently operate at 5 V and has a number of nice analog peripherals, which we happened to have been exploring for other applications in the lab. The microcontroller was tethered to a personal computer running Python for data recording and processing. This connection was provided by a FTDI TTL-232 cable which also powered the microcontroller.

Figure 2 is a schematic diagram for the converter board. **Photo 1** shows the components assembled on to strip-board. Since we were building only three of these and the discrete components were few, this form of construction was convenient. However, because of the laboratory environment, it was important to house the boards in robust cases with BNC connectors for the analog voltages (see **Photo 2**). Discounting labor, this packaging dominated the cost of the build.

With power and serial-port communication coming through the TTL-232 cable, there's not a lot beyond the resistor dividers and capacitors for the A0 through A3 analog channels at the top left of the photograph. This signal conditioning is used to get the 0-10V output from the photodetectors into the 0- to 5-V range of the microcontroller and to provide an adequate input to the ADC. Since the optical system needed to be calibrated every time it was adjusted and we were only interested in the ratio of the photodetector signals, feeding the USB voltage to the microcontroller and using it as the reference voltage for the ADCs was not a problem. Note that, although not indicated in the schematic diagram, the selection of bias voltage was only available on analog channel 0. The others were hard-wired to the zero volt rail.

Development of the firmware first explored the use of the ADC on the microcontroller, then the use of the Timer1 peripheral hardware for regulating the period of the main loop. The final application program made use of the service words defined for controlling the peripheral modules.

Listing 2 shows the FlashForth firmware for interacting with the ADC. It starts with the definitions for the various registers of interest, followed by definitions of service words and ends with a sample application to try out all that has been defined. The `init-adc` word sets the shared port pins appropriately and configures the ADC peripheral hardware. The `set-adc-chan` word masks the channel selection value to limit it to the appropriate range and then writes it into the `ad1chs` register in order to route the selected input

```
-main-app
marker -main-app
\ Sample analog channels AN0 through AN3 in a timed super-loop.
\ This file uses the words defined in my-adc.txt and my-timer-16MHz.txt.
\ Note that the period, in milliseconds can be set with set-ms-period.
\ Run the periodic super-loop, until a key is pressed
: run-app ( -- )
  decimal
  init-adc
  init-tmr
  0 my-count !
  begin
    period-count 2@ reset-tmr
    my-count @ dup u. 1+ my-count ! \ Print and increment
    #0 set-adc-chan adc@ u.
    #1 set-adc-chan adc@ u.
    #2 set-adc-chan adc@ u.
    #3 set-adc-chan adc@ u.
    cr
    wait-for-tmr
  key? until
;
```

LISTING 4
Source code for the top-level application

pin to the ADC. The `adc@` word sets off the ADC sampling process and waits for the conversion to be complete. It then fetches the ADC result value from the `adc1buf0` register and leaves it on the top of the data stack. Finally, the `test-adc` word exercises the previously-defined service words by periodically sampling all four channels and writing the results to the user console. The `u.` word renders the ADC result to the console as an unsigned value.

Since we wanted the reporting period to be precise, we elected to control the period with a hardware timer. Rather than use the `ms` word to get delays, we wait at the end of each pass through the main loop for the expiry of the hardware timer. **Listing 3** shows the firmware for interacting with the capture-compare timer module 1. After defining names for the registers of interest, we make use of FlashForth's elegant handling of memory contexts. The `period-count` variable is mostly kept constant but we would like to be able to adjust it occasionally without having to reprogram the firmware. This is an obvious application for the microcontroller's EEPROM and FlashForth makes it easy to define a double variable (i.e., a 32-bit variable) in EEPROM and then be able to access it as conveniently as any other address in memory. After defining `period-count` in the EEPROM context, the memory context is changed back to static RAM for the `my-count` variable. This particular variable will be changed

frequently, once for each pass through the main loop. Next come the definitions for the service words `init-tmr`, `reset-tmr`, `wait-for-tmr`, and `stop-tmr` whose names are fairly self-explanatory. The `wait-for-timer` word blocks until the timer has reached the previously set compare value. The final service word `set-ms-period` accepts the requested time period in milliseconds and computes the appropriate period count in timer ticks and stores it into the EEPROM variable. We immediately use that word to set a default period of 500 ms and, finally, define the `test-tmr` word to exercise the service words conveniently.

It is worth noting that the code in **Listing 3** was not developed linearly, as it appears now. The firmware started small, with just the register definitions. Some interactive exploration followed, in which register bits were set and cleared (and the datasheet reread many tines) to get an understanding of the timer and its associated output-compare module. As each piece of the puzzle was understood, such as turning-on or turning-off the timer, that piece of code was put into a word definition. The overall code grew nonlinearly and iteratively.

Once the hardware modules were understood and the service words complete, the main application was developed from the text of the two testing words. **Listing 4** shows the definition of the `run-app` top-level word that defines our application code. After initializing the hardware modules and storing zero in the `my-count` variable, the main loop resets the hardware timer with the 32-bit period count, increments and prints the value of the `my-count` variable as a form of time stamp, and then samples and reports the four analog channels. The microcontroller then waits for the timer to expire and goes around the loop again, so long as no character has been received from the attached PC. If a character has been received, the `run-app` word finishes and control returns to the FlashForth interpreter.

Although this example has kept the microcontroller tethered to the personal computer, we often have started with our FlashForth devices tethered to a computer for development and configuration but then "run them stand-alone for most experiments. The final touch when making a stand-alone embedded application is to redirect execution of our main word at start up. This can be done with:

```
‘ run-app is turnkey
```

The single-quote word gets the execution token for the run-app word and this is then saved in the turnkey vector. At power-up, the microcontroller will listen for an incoming escape character and, if one is not received within 2 s, the `run-app` word will be executed automatically.

FLASHFORTH FOR EMBEDDED

In our lab, we have found that Forth enables convenient real-time interaction with hardware. Although we have just provided a simple example here, the laboratory has a range of FlashForth programmed devices for monitoring shock-tunnel processes, reporting pressure measurements, and triggering fuel-injection systems and LED flash lamps. The convenient interactive environment has allowed the development of the firmware through a simple, layered process. As each peripheral was understood and the service words coded, we could then work on the next aspect of the program. You should try FlashForth for your embedded applications.

circuitcellar.com/ccmaterials

REFERENCES

[1] E. Hand and D. Cleary, "Updated: Main mission for Philae comet lander comes to an end," *Science*, 2014, http://news.sciencemag.org/europe/2014/11/updated-main-mission-philae-comet-lander-comes-end.

[2] The CPUSHACK Museum, "Here comes Philae! Powered by an RTX2010," 2014, www.cpushack.com/2014/11/12/here-comes-philae-powered-by-an-rtx2010/.

[3] E. D. Rather, D. R. Colburn, and C. H. Moore, "The Evolution of Forth," ACM SIGPLAN Notices Vol. 28 No. 3, 1993, www.forth.com/resources/evolution/index.html

[4] M. G. Kelly and N. Spies, *FORTH: A Text and Reference,* Prentice-Hall, 1986.

[5] M. Nordman, "FLASHFORTH for the Microchip PIC18, 24, 30, 33 Series and Atmel ATmega (Arduino) Series," 2014, http://flashforth.com/.

RESOURCES

"Blink," www.arduino.cc/en/Tutorial/Blink.

P.A. Jacobs, "A tutorial guide to programming PIC18, PIC24 and Atmega microcontrollers with FlashForth," 2014, https://espace.library.uq.edu.au/view/UQ:330707.

P. Jacobs, P. Zawaski, and M. Nordman, "Elements of FlashForth," 2014, https://espace.library.uq.edu.au/view/UQ:321883.

SOURCES

ATMega328 Microcontroller
Atmel Corp. | www.atmel.com

TTL-232 Cable
FTDI | www.ftdichip.com

PIC24FV16KM202 Microcontroller
Microchip Technology | www.microchip.com

RedBoard
SparkFun | www.sparkfun.com

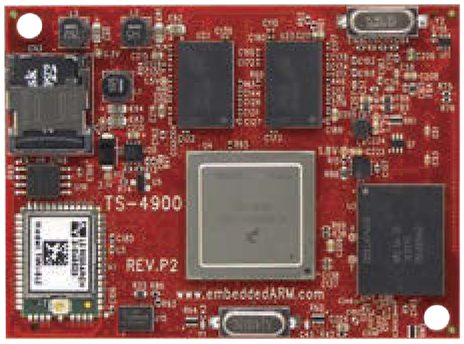
SUPERIOR EMBEDDED SOLUTIONS

Technologic Systems

DESIGN YOUR SOLUTION TODAY
CALL 480-837-5200

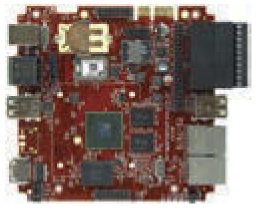
www.embeddedARM.com

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



Pricing Starts At
$99 Qty 100
$134 Qty 1

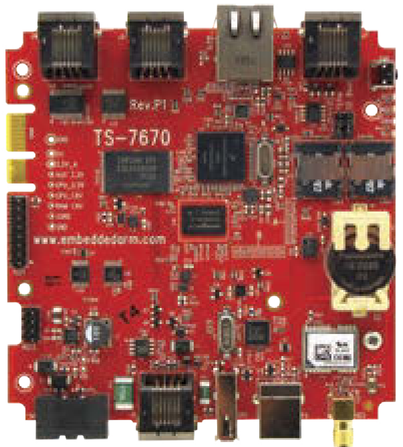
- 1 GHz Single or Quad Core Cortex A9 ARM CPU
- Up to 2 GB DDR3 RAM and 4 GB eMMC Flash
- Wireless 802.11 b/g/n and Bluetooth 4.0
- 4 KLut FPGA, 1x Gigabit Ethernet, 1x PCI Express Bus
- 1x mSD slot, 1x SATA II, 1x USB Host, 1x USB OTG
- 70x DIO, 4x I2C, 1x I2S, 2x SPI, 2x CAN
- 1x LVDS & 1x RGB Display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



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

Available w/ TS-TPC-8390 Touch Panel Computer

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



Pricing starts at
$129 Qty 100 $168 Qty 1

low cost plastic enclosure available

Features:

- 454MHz ARM CPU
- Up to 256MB RAM
- 2GB NAND/eMMC 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

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

LONG LIFE — We've never discontinued a product in 30 years

RUGGED — Embedded systems that are built to endure

OPEN — Support every step of the way with open source vision

ORIGINAL — Unique embedded solutions add value for our customers

www.embeddedARM.com

LCDTV Server

Streaming Media to a TV via the USB Port

Lindsay's LCDTV Server project enables an LCD TV equipped with a USB port to stream media across a LAN. The small adapter converts the mass storage device requests coming from the USB into LAN media requests using a virtual file system. When combined with a power line-to-Ethernet Bridge, the user can watch digital video on an older TV from anywhere in the neighborhood.

By Lindsay Meek (Australia)

My LCDTV Server design is a small adapter intended to plug into a "not so smart" LCD TV or portable DVD player and give it the capability to play streaming media. It does this by attaching itself to the USB port of the TV and emulating a standard USB flash memory. The TV accesses the USB port while searching for media, and the adapter intercepts the read requests and transmits them across a powerline LAN to a media server. The media server is then able to respond with content, which is then reformatted and sent back to the TV. So, the TV still believes it is communicating with a flash memory when in fact it could be coming from any number of online streaming media services. This allows existing players to be retrofitted with a device to upgrade them to be "smarter" and connectible to the Internet, thus prolonging their usefulness and avoiding the curbside garbage collection. Modern media servers such as Tversity also have the ability to automatically transcode content on the fly from newer media formats such as H.264 and MKV to older formats like MPEG.

The LCDTV Server consists of two main system components: the hardware adapter itself that translates the USB to and from Ethernet requests and a server-side process (see **Figure 1**). This process receives the Ethernet requests from the adapter and translates them into DLNA/UPNP accesses suitable for communicating to media servers.

HARDWARE ARCHITECTURE

The system's hardware comprises a WIZnet WIZ550io Ethernet SPI module and a small PCB containing a USB plug, power supply, and microprocessor. The WIZ550io module is a good choice for a high-bandwidth media-streaming application because it comes equipped with 100-MBps Ethernet and uses hardware-accelerated packet and TCP socket processing. It also supports an 80-MBps maximum data rate over its SPI port, so it isn't a significant bottleneck during the transfer of media data. In addition, the WIZ550io is ready-built, so the process of designing the adapter prototype is easy. You can use a simple two-layer PCB for the microcontroller and avoid paying careful attention to the 100-MBps Ethernet track routing.

The WIZ550io module interfaces to the PCB using two single-in-line connectors, which provides the SPI along with its requirement for 3.3-V regulated power and the RDY signaling line, used to indicate when it has initialized. The PCB contains the USB device-class interface, which plugs into the LCD TV. The USB port provides the power output of 5 V at 500 mA, which is then regulated down to 3.3 V for the microprocessor and WIZ55oio module using a simple linear regulator. It is estimated the maximum current draw from the adapter is approximately 156 mA, which is well within the supply capabilities.

A Microchip Technology PIC24FJ64GB002 microcontroller is used to implement the USB mass-storage device and WIZ5500 interfaces (see **Figure 2**). This microprocessor has a relatively small footprint and includes a full-speed, 12-MBps USB peripheral. Another of the microprocessor's advantages is that it can operate from its internal oscillator without violating the USB timing specification, which saves a bit of PCB space and the cost of an external 12-MHz crystal. The microprocessor's SPI peripheral can also operate in FIFO mode, which enables overlapped SPI/CPU processing during data transfers and helps to accelerate the transfer rate.

SOFTWARE ARCHITECTURE

The adapter's software consists of three main system modules: a USB mass-storage device driver for communicating with the LCD TV, an ATA-over-Ethernet (ATAOE) client for communicating with the server, and a driver for communicating with the W5500 chip on the WIZ550io module. The code compiles using Microchip Technology's MPLAB X and the XC16 Compiler-Lite.

The USB mass-storage device driver is initialized when the adapter is plugged into the TV. This device driver is based on the USB framework provided by Microchip Technology in its application libraries. The driver communicates with the TV and provides two internal API functions `FILEIO_NET_MediaDetect` and `FILEIO_NET_SectorRead`, which are used to detect the presence of the media and read a 512-byte sector, respectively. These two functions are used to communicate with the ATAOE client, which uses MediaDetect to broadcast "are you there" messages to the LAN and wait for a valid response from the server. Once the server has been detected, the USB mass-storage device driver is informed that the media is detected. It then relays `SectorRead` requests from the TV, which are translated to ATAOE requests and transmitted to the server.

The ATAOE protocol is a lightweight Ethernet protocol that encapsulates ATA disk drive commands into Ethernet packets. It was

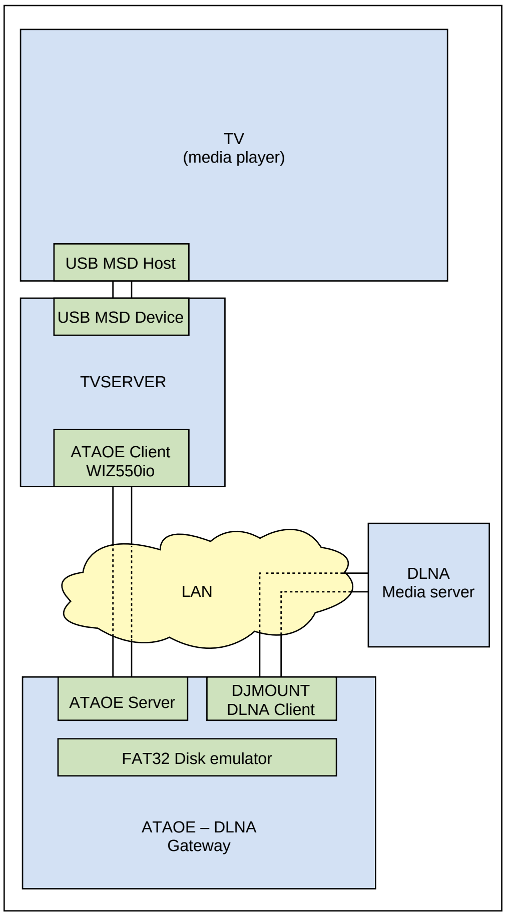


FIGURE 1
An overview of the TV Server System

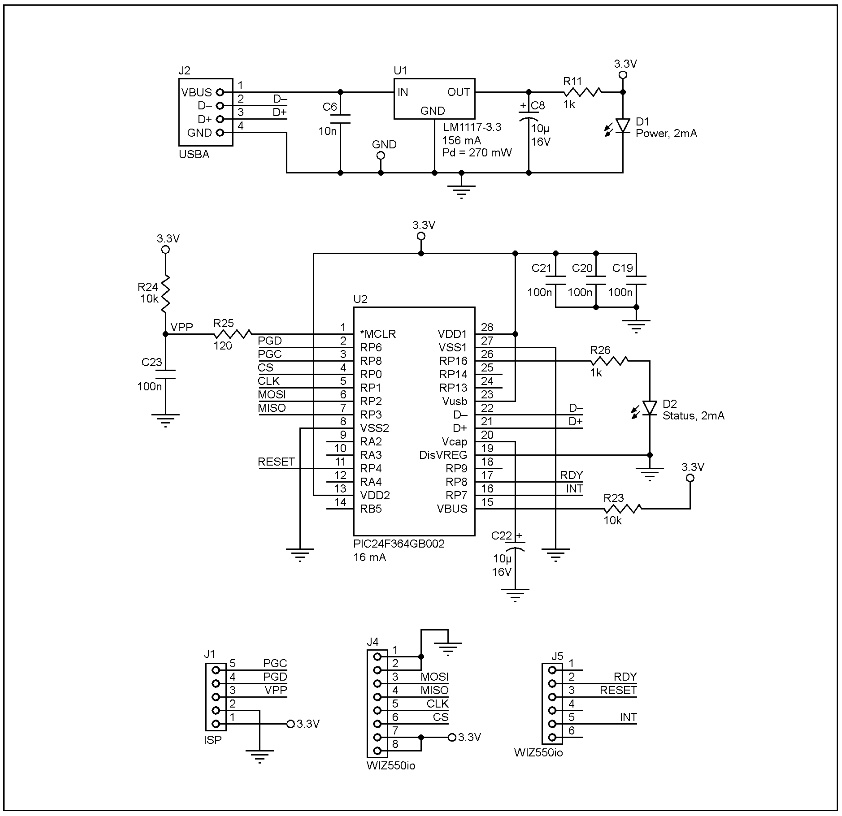


FIGURE 2
The system's complete circuitry.

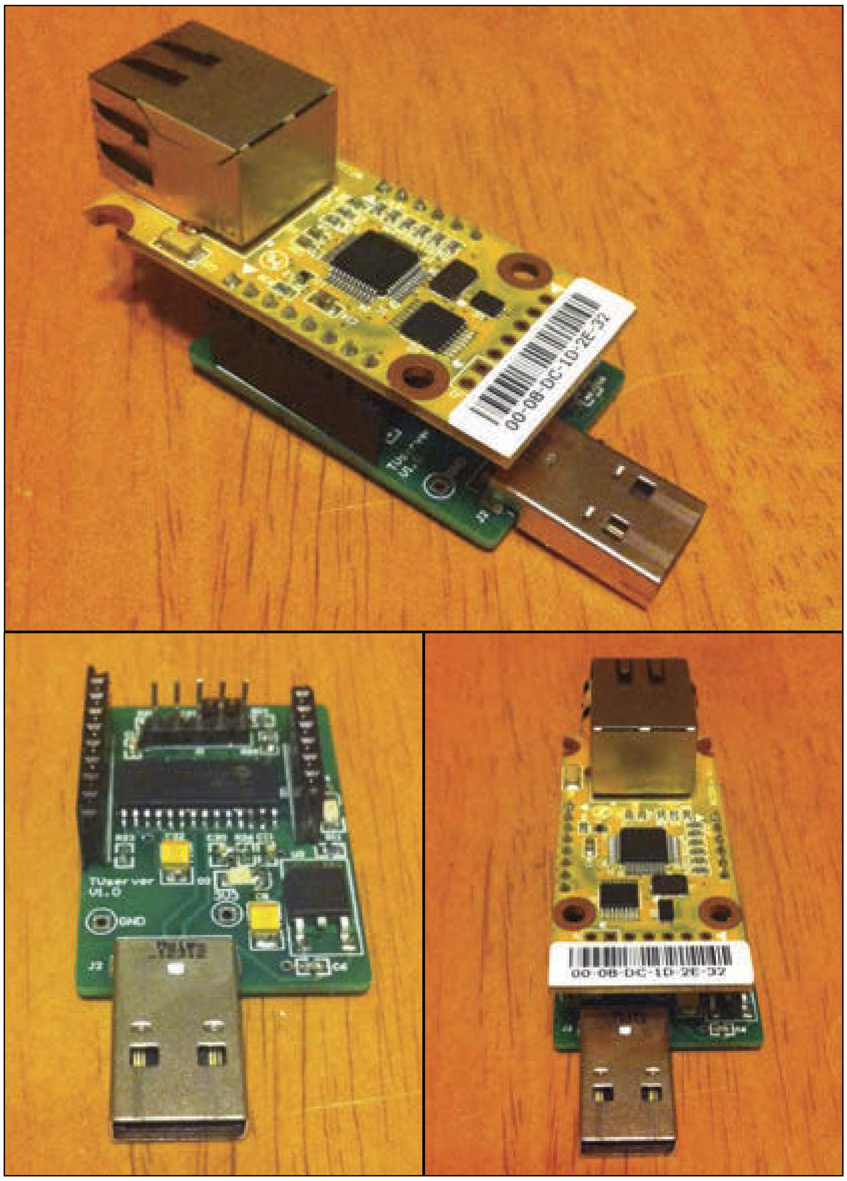


PHOTO 1
The TV Server Adaptor

originally developed for disk driver clusters in server farms, but due to its elegant simplicity, it is suited to high-speed disk I/O over a LAN using a simple embedded protocol stack. It also has a 1:1 correspondence to the USB mass-storage device commands, which perhaps are also based on the ATA disk drive command set. The ATAOE client is only around 400 lines of additional C code to the USB mass storage driver.

The ATAOE client consists of the single function `read_net`, which reads a single sector from the server at a nominated disk logical block address. This function determines if the server has been detected, and if not it will broadcast the ATAOE `config` commands onto the LAN until the server responds. It then transmits the ATA read sector packets to the server, waits for the response and then returns the data to the USB mass storage device. If the server does not respond after several retries, the client will reset and attempt to re-acquire the server.

The `read_net` function also includes several performance optimizations to improve data throughput during the media streaming process. Most are based on the premise that the media is read in a linear sequence and the next data proceeds the current data.

Each ATAOE read sector request contains two sectors instead of one, anticipating that the next `read_net` request will be for this sector. The driver records the location of the data inside the W5500 memory. Thus, if it detects a read to this sector on the next request, it will copy the data directly from the W5500 memory rather than making another request. This optimization improves throughput by avoiding one read request every two, which would invoke a round-trip delay.

In addition, the ATAOE client issues a "speculative" read request after it receives the response from the server to first read request of two sectors. This speculative request is for an additional two sectors ahead of the current request. It is allowed to execute in parallel whilst the USB device is returning the data from the previous request.

The ATAOE client communicates with the W5500 chip via a driver. This driver is a modified version of the Socket API provided by WIZnet. The driver was stripped back and optimized to work with the PIC SPI hardware and communicate efficiently in RAW mode on W5500 Socket 0. Socket 0 is used in RAW mode as ATAOE is a low-level Ethernet Protocol, which uses a different frame identifier to standard IP. The Socket is initialized to use most of the available packet buffer RAM inside the W5500, with a small amount reserved for debugging via a Telnet Server on Port 23/ Socket 1.

The other area of the driver that is optimized is the transfer of data from the W5500 packet buffer to memory of the PIC via the `WIZCHIP_READ_BUF` function. This was recoded in assembler for maximum speed. It has a dedicated subroutine for transferring 512-byte disk sectors in groups of 4 bytes at a time. The assembler algorithm takes advantage of the SPI FIFO capability of the microprocessor by transmitting SPI bytes "ahead" of the responses, allowing overlapped processing to take place between SPI data transfers and storing the incoming data in RAM.

SERVER-SIDE PROCESS

The other main component of the TV Server is a server-side process. This process is used to translate the ATAOE requests coming from the TV server adaptor onto the DLNA Media Network. It does this using three main software components linked to each other: an ATAOE server, a disk drive emulator, and a DLNA network interface via the DJMount package. It runs under Linux in order to use DJMount and also get a good network performance.

The ATAOE server component is responsible for providing the interface to the ATAOE client running on the TV server adapter. The server uses the standard sockets library built into Linux and sets the socket into RAW mode for directly processing Ethernet frames. It responds to ATAOE configuration broadcasts by announcing the network address of its virtual disk drive emulator. The adapter detects the presence of the server and responds by sending read sector requests. The read sector requests are intercepted by the ATAOE server and sent to the disk drive emulator, where virtual disk sectors of media information are formed and transmitted back to the TV for display.

The disk drive emulator implements an virtual 200-GB FAT32 file system. The emulator intercepts the read sector requests coming from TV server adapter and determines an appropriate response based on its virtual disk geometry. So there is a function for "rendering" the Master Boot Record, the File Information Sector, the Boot Record, the File Allocation Tables, the Directories and finally the Media Files themselves.

The emulator maintains a linked list of directory and file entries, which contains the important FAT information such as the start and stop cluster address and the length, as well as the full network name needed for locating the data via the DLNA media network. This list is generated at startup by recursively browsing the entire contents of the local DLNA network.

The virtual clusters are allocated dynamically and to simplify the implementation, the files always use a linear range of clusters, and each directory is limited to a maximum size of one cluster (32 Kb). The FAT32 32-byte directory entries are rendered into cluster memory buffers, which are then linked to the directory entries in order to speed up browsing from the TV.

The media files that are located during the recursive browse of the network are also allocated onto the linked list. They are given a range of cluster addresses based on the file size. The files are not loaded into memory until the individual clusters are accessed by

FlashPro430
FlashPro-CC
FlashPro2000
GangPro430
MSP-GANG
GangPro-CC

MSP-GANG

NEW

FlashPro-CM
FlashPro-LM
FlashPro-STM

USB Flash and Gang Programmers for Texas Instruments' MCUs
MSP430, Chipcon CCxx, C2000, Stellaris LMxx and
ST-Microelectronics MCUs - STM32xx (ARM)

* can assign unique serial number
* up to 64 USB-FPA programmers can be connected to one PC and program target devices simultaneously

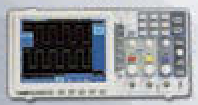
USB-FPA

Reliable and the fastest programmer on the market.
Perfect for production usage.

Elprotronic
Incorporated
www.elprotronic.com

BEST SCOPES, LOWEST PRICES

PASSPORT-SIZE PC SCOPES $149+
Great scopes for field use with laptops. Up tp 200MHz bandwidth with 1GSa/s, high speed data streaming to 1MSa/s, built-in 1GSa/s AWG/function 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 Saelig

ABOUT THE AUTHOR

Lindsay Meek is currently employed in Magellan Power's R&D department. He holds a bachelor's degree in engineering, as well as a master's of philosophy. Lindsay is interested in emerging technologies and enjoys participating in international design contests.

the TV.

The file "renderer" uses a data streaming algorithm that runs in a separate thread to the main ATAOE server, once again in order to improve the system throughput. The streaming process uses double-buffering so it attempts to "read-ahead" the next set of clusters into a separate data buffer before the request is actually received from the TV. This allows the server to overlap the data transfer from the media server and to the TV. Through a process of experimentation, a good cache size was determined to be four clusters (128 Kb). This size is also dynamically reduced to be one cluster to minimize latency during random access file reads, which the TV sometimes does whilst browsing directories and generating thumbnail images. The four cluster read-ahead is only enabled when the server detects the multiple consecutive reads of media playback.

The final software component "DJMount" is responsible for retrieving directories and media files from the DLNA media servers. This is an open-source package which hides the complexity of the DLNA/UPNP media protocols and presents a relatively-simple file system interface. Hence, it provides the POSIX-style functions of `getdir`, `getattr`, and `open/close/read` file.

In order to convert DJMount into a FAT32/ATAOE server, it was modified slightly such that the highest level of the program was re-written. The standard distribution links with the 'Fuse' library which provides a file-system-in-user-space, containing a `fuse_main` function. This was re-written to implement the server algorithms and embedded into an extra source file confuse.c inside the main DJMOUNT build directory. The Makefile.in was also adjusted to compile the target `tvserver` instead of `djmount`.

The last tweak that was made to the server process was the ability to change the location of the initial FAT32 root directory. This was done as it was found some media players could only scan and display a few levels deep into the directory tree.

PERFORMANCE TESTS

Some tests were done on the system, using a Ubuntu server hosting the server-side process and a Tversity DLNA server. Some sample videos were tested at VCD (MPEG-I) and DVD resolutions (MPEG-II and H.264). The playback was smooth and uninterrupted over a 100-MBps LAN cable; however, it could be seen from the almost solid 'busy LED' on the adapter that is was heavily loaded during DVD sequences with a lot of motion in the video frame.

The next test was to introduce a pair of generic 150-MBps Ethernet-over-Power-Line transceivers. I found the transceivers did reduce the playback performance slightly, with frequent "stuttering" on the screen during DVD playback. I traced the source of the problem to my fan-heater and solar inverter, which seemed to be interfering with the power-line signal when they operated.

The upper bandwidth limits of the SPI and USB interfaces also prevented higher definition videos from streaming smoothly. The PIC24's SPI peripheral maximum speed was only capable of 8 MBps despite having a higher internal clock speed, and the USB needs to be "high" speed instead of "full" speed to exceed 12 MBps. Microchip have recently released a new 32-bit MZ series which promises faster peripherals including a high-speed USB.

SUCCESSFUL STREAMING

The experiment with the prototype showed that the concept worked, and it is possible to stream video to a modern "not so smart" TV with minimal additional hardware. And with the addition of Ethernet-over-power adapters, the media can be streamed anywhere in the house, and in my case, beyond the range of my Wi-Fi access point.

circuitcellar.com/ccmaterials

RESOURCES

Djmount 0.71, http://djmount.sourceforge.net/

Microchip Technology, Library for Applications www.microchip.com/pagehandler/en-us/devtools/mla/.

———, PIC24FJ64GB002 Data Sheet, www.microchip.com/TechDoc.aspx?type=datasheet&product=PIC24FJ64GB002.

WIZnet, Socket APIs V1.02, https://github.com/Wiznet/W5500_EVB/tree/master/ioLibrary/Ethernet.

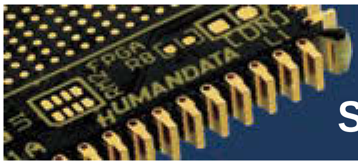
———, "W5500 Data Sheet," www.wiznet.co.kr.

———, "WIZ550io Data Sheet," www.wiznet.co.kr.

SOURCES

PIC24FJ64GB002 Microcontroller
Microchip Technology | www.microchip.com

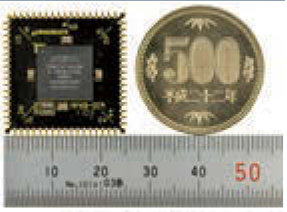
WIZ550io Ethernet module
WIZnet | www.wiznet.co.kr



Easy and Quickly Mountable on 68-pin IC socket

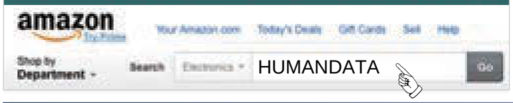
Stamp size FPGA/CPLD Module **PLCC68** Series

PLCC68 series is a very small size FPGA & CPLD Module designed to be used with 68-pin IC socket. It's very convenient to use on universal boards.
PLCC68 series requires only single 3.3V power supply.
The pin assignments are the same for all series. I/O power supplies are separated in two banks (A and B, 25 I/Os each).



- ☐ 50 I/Os(External clock inputs are available)
- ☐ 3.3V single power supply operation
- ☐ Voltage converters for auxiliary power supply
- ☐ Very small size (25.3 x 25.3 [mm])
- ☐ Separated power supply: Core, I/O
- ☐ JTAG Signals (TCK, TMS, TDI, TDO)
- ☐ All series have the same pin assignment
- ☐ Mountable on IC socket

Visit Our Amazon USA Store



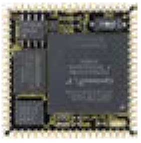
[USAGE EXAMPLE]

ALTERA Series

Cyclone V PLCC68 FPGA Module

AP68-06 Series

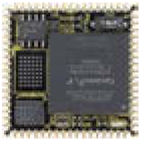
- FPGA : 5CEBA4U15C8N
- FRAM
- 3.3 V single power supply operation
- On-board oscillator, 50MHz
- Configuration Device
- 8 Layer PCB
- Mountable on 68pin IC socket


RoHS compliant

Cyclone V PLCC68 FPGA Module

AP68-06Z Series

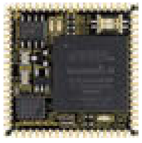
- FPGA : 5CEBA4U15C8N
- 3.3V single power supply operation
- On-board oscillator, 50MHz
- Configuration Device
- 8 Layers PCB
- Board size : 25.3 x 25.3[mm]
- Mountable on 68pin IC socket


RoHS compliant

Cyclone III PLCC68 FPGA Module

AP68-04 Series

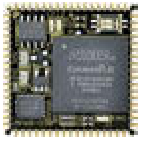
- FPGA : EP3C25U256C8N
- 3.3V single power supply operation
- On-board oscillator, 50MHz
- Configuration Device
- 8 Layers PCB
- Board size : 25.3 x 25.3[mm]
- Mountable on 68pin IC socket


RoHS compliant

Cyclone III PLCC68 FPGA Module

AP68-03 Series

- FPGA : EP3C10U256C8N
- 3.3V single power supply operation
- On-board oscillator, 50MHz
- Configuration Device
- 8 Layers PCB
- Board size : 25.3 x 25.3[mm]
- Mountable on 68pin IC socket


RoHS compliant

MAX V PLCC68 CPLD Module

AP68-02 Series

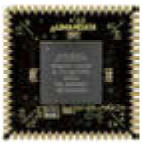
- CPLD : 5M570ZF256C5N
- 3.3V single power supply operation
- Board size : 25.3 x 25.3[mm]
- Mountable on 68pin IC socket
- 4 Layers PCB

RoHS compliant

MAX II PLCC68 CPLD Module

AP68-01 Series

- CPLD : EPM240F100C5N
  EPM570F100C5N
- 3.3V single power supply operation
- Board size : 25.3 x 25.3[mm]
- Mountable on 68pin IC socket
- 4 Layers PCB


RoHS compliant

XILINX Series

Spartan-6 PLCC68 FPGA Module

XP68-03 Series

- FPGA : XC6SLX45-2CSG324C
- 3.3V single power supply operation
- on-board oscillator, 50MHz
- Configuration Device
- Board size : 25.3 x 25.3[mm]
- 6 Layers PCB
- Mountable on 68pin IC socket

RoHS compliant

Spartan-3AN PLCC68 FPGA Module

XP68-02 Series

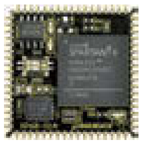
- FPGA : XC3S200AN-4FTG256C
- 3.3V single power supply operation
- on-board oscillator, 50MHz
- Board size : 25.3 x 25.3[mm]
- 6 Layers PCB
- Mountable on 68pin IC socket

RoHS compliant

Spartan-6 PLCC68 FPGA Module

XP68-01 Series

- FPGA : XC6SLX16-2CSG225C
- 3.3V single power supply operation
- on-board oscillator, 50MHz
- Configuration Device
- Board size : 25.3 x 25.3[mm]
- 6 Layers PCB
- Mountable on 68pin IC socket


RoHS compliant

We also have many other products
All stocked items are ready to be shipped immediately

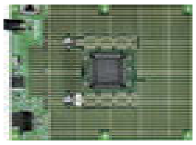
Technical documents are available for free before purchasing

Universal board for AP/XP PLCC68 Module

ZKB-092

RoHS compliant

- JTAG-BUFFER Circuit / JTAG Connector
- Power input :
  DC5V/2.1[mm] Jack / Terminal Block
- 3.3V Power Circuit
- Power Switch / Power LED
- Board size : 4.528"x 6.102" (115 x 155 mm)



HUMANDATA®

HuMANDATA LTD.
TEL：+81-72-620-2002 (Japanese) FAX：+81-72-620-2003 (Japanese/English)
E-Mail : s2@hdl.co.jp URL : http://www2.hdl.co.jp/en/

See all our products, A/D D/A conversion board, boards with USB chip from FTDI and accessories at :
www.hdl.co.jp/CC1501

Rapid Prototyping of SuperSpeed USB Devices

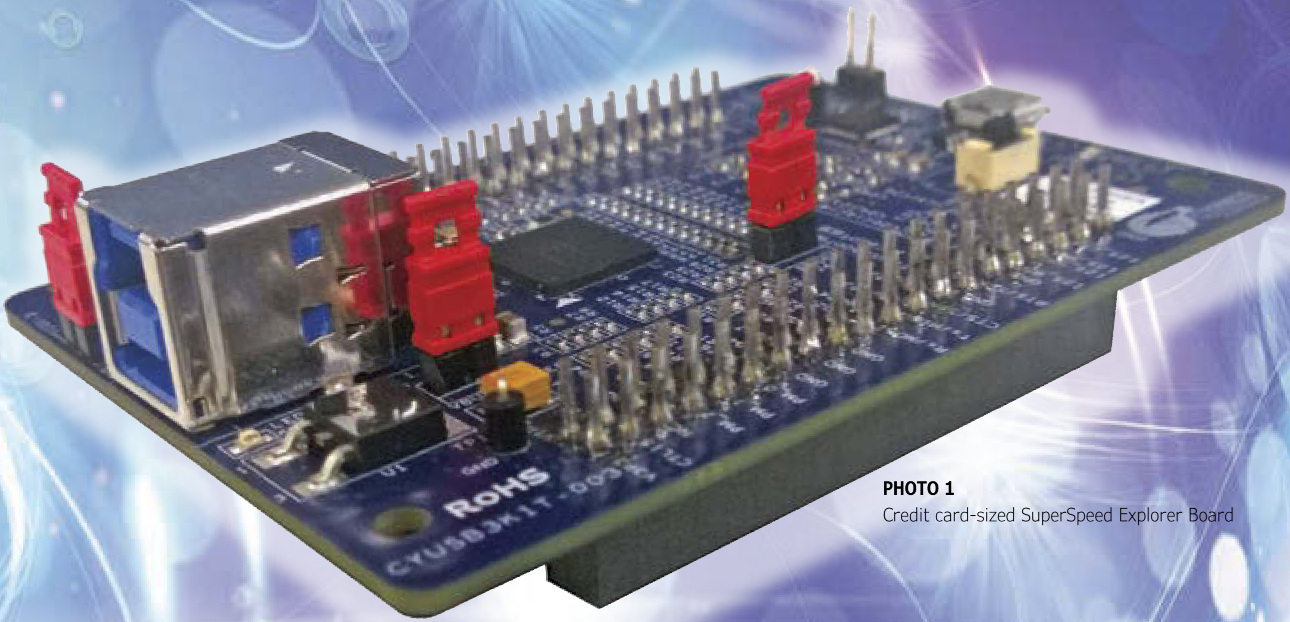


PHOTO 1
Credit card-sized SuperSpeed Explorer Board

SuperSpeed USB is the third extension of the USB Standard, and it was designed with data throughput and lower power levels in mind. Use this article as a guide to the end-to-end development of a SuperSpeed USB device.

By John Hyde (US)

What? Are you crazy? That is much too hard! There are too many things to consider and the development environment is too expensive for my budget! Yes, I've noticed that most PCs available today come with SuperSpeed USB included, and tablets and smart phones are not far behind. But SuperSpeed devices, except for video cameras and hard drives, are hard to find. This must be because they are very difficult to design!

If this is your perception of SuperSpeed device development then this article will be an eye opener since it describes a new set of game changing, development tools and explains why SuperSpeed devices will soon proliferate. This toolset—which includes low-cost development boards, free software tools, and a collection of working examples—is designed for people who need to deal with data that is generated, or consumed, at data rates between 100 MBps and 400 MBps. This is far in excess of the Arduino or similar products usually covered in this magazine but come and look at the "high-end" for a moment, you'll discover that it may well be within your reach. This is what several of my SuperSpeed customers have found. I am not going to promise "easy," but if you know C, then it is certainly straight forward using a methodical approach.

SuperSpeed USB is the third extension of the USB Standard and it was designed with data throughput and lower power levels in mind. The USB 3.0 standard, which supports data transfers of 5 Gbps (400 MBps) has recently been superseded by USB 3.1, which describes data transfers of 10 Gbps (1G Bps). Today's silicon supports 5 Gbps and all USB silicon vendors are working hard to develop 10 Gbps parts; however, since USB 3.1 changed low-level bus signaling, it will be late 2015 or early 2016 before commercial parts become available. Today's silicon will implement these signaling changes such that today's designs are easily upgraded to 10 Gbps. Another focus of SuperSpeed USB was to reduce the overall system power and it accomplished this by eliminating the need for a device to be polled and therefore the USB links can be switched to lower power states more often. The new USB 3.1 Gen 2 silicon will operate the same only faster and using less power.

LOW-COST HARDWARE

Photo 1 shows a credit card-sized development board based upon Cypress Semiconductor's FX3 SuperSpeed device controller. Cypress calls this the SuperSpeed Explorer Kit. **Figure 1** shows a block diagram of the FX3 and its interfaces to the development board connectors. The FX3 is a system-on-a-chip containing a USB 3.1 Gen 1 (5 Gbps) interface including PHY, a 32-bit parallel interface that can run up to 100 MHz, 512 KB of RAM for buffering and program storage, an ARM CPU and a collection of low-speed serial interfaces. Since all USB 3.1 Gen 1 devices are required to operate at high speed (480 Mbps), the FX3 includes a USB 2.0 PHY, and it can also operate as an OTG host at this speed. Alongside the USB block is the EZ-Dtect block that, when enabled by the processor, allows the USB-PHY to detect the presence of a connection to a USB charger. The board contains an integrated debugger, EEPROM for boot code and a user button and LED. All signals from the parallel bus and serial interfaces and power are connected to two 40-pin, 0.1" connectors.

The FX3 was specifically designed with data throughput in mind and it includes a distributed DMA controller that can support simultaneous read and write transfers of up to 800 MBps. It is expected that USB bulk packets will be used to get maximum throughput on SuperSpeed USB since they are 1-KB long and they can be burst up to 16 packets deep. Let's consider the host computer sending data to the FX3. Blocks of data 16-KB long will arrive at the FX3's USB interface. These must be buffered by the FX3, which writes the blocks directly to system RAM at SuperSpeed data rates. (There are no separate packet buffers and the RAM is zero wait states.) The host will want to send multiple 16-KB blocks so the FX3 should have multiple buffers to keep up with back-to-back data transfers. The FX3 can use up to 256 KB of its RAM for data buffering. As soon as the first 16-KB block arrives from USB it is presented to the parallel interface which can start to output it as the second 16-KB block is arriving from USB. This concurrent USB-read and parallel interface-write will boost throughput. The parallel interface is called GPIF II in Figure 1. This is an acronym for General Programmable InterFace Gen 2 (the Cypress FX2 has a GPIF I), and it is a programmable state machine.

The GPIF II subsystem is FPGA-like in nature; it contains some hardware–assist blocks such as counters and comparators but is mainly uncommitted logic that must be programmed on power up. You develop a state machine that implements the protocol required to move these 16-KB DMA buffers to the outside world. Cypress provides examples of industry standard protocols such as master FIFO, slave FIFO and async RAM but you can also develop your own if you have to interface to some custom hardware. There

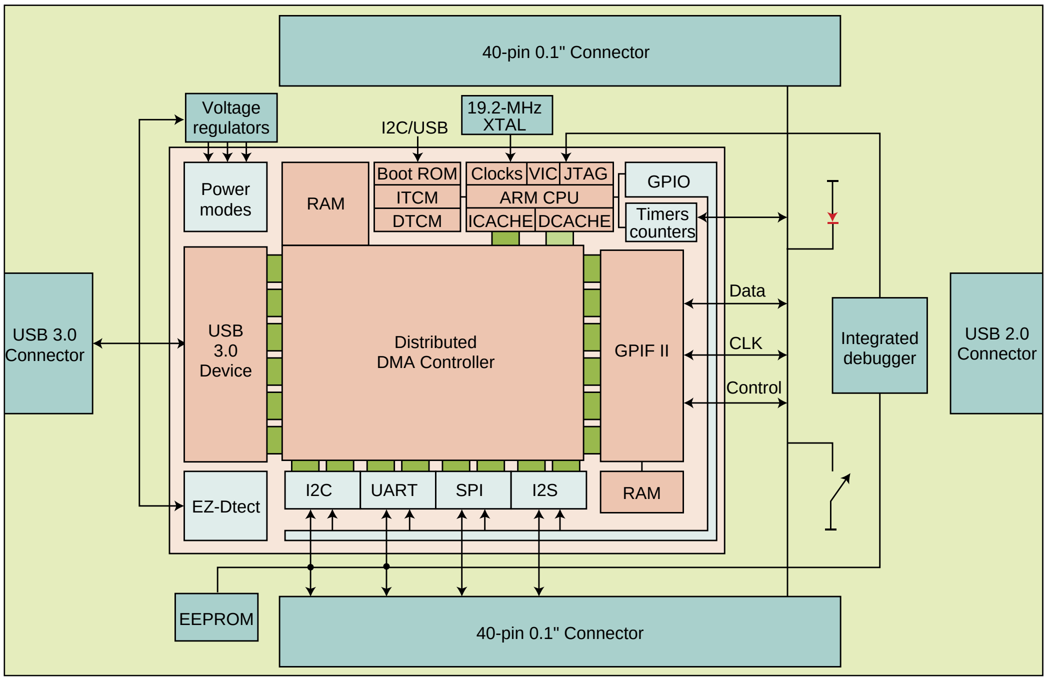


FIGURE 1
Block diagram of FX3 showing connections to the SuperSpeed Explorer Board

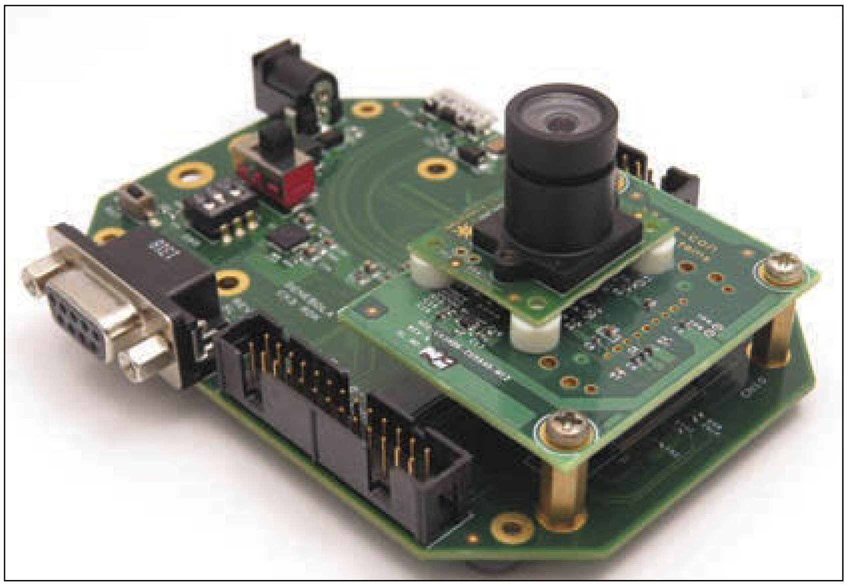


PHOTO 2
Video camera reference design using CX3 (Source: e-con systems (http://www.e-consystems.com/)

are 14 bidirectional control lines available to implement these standard or custom protocols. In the simplest case, the GPIF state machine could output 32 bits with a 100-MHz clock and the external hardware would need to strobe this data. Typically, you would add flow control such as `FIFO_Full` and this will reduce the data rate to something that the external hardware can handle. You would also need `DataValid` in case USB is not keeping up. Note that the FX3 will not be the bottleneck since it can run constantly at 400 MBps.

HOST COMPUTER SOFTWARE

In this simple Byte Mover example, you will need to write a custom program on the host computer since Windows, Mac OS nor Linux support a generic byte mover device. Cypress provides an example FX3 device driver, including source code, for each of these OSes but this is still a daunting task to undertake. There is a better way.

In my first example I used the DMA controller to move data from the USB subsystem to the GPIF II subsystem. This can be done at 400 MBps and does not need any involvement from the CPU except for the initial setup of the connection. I can also move the data via the CPU but this will decrease the throughput. The CPU is a 200-MHz ARM that can inspect, process and pass on modified data. This will allow you, for example, to add a higher level protocol to the data. Cypress provides examples using the USB Video Class (UVC) driver and the Mass Storage Class (MSC) driver. The CPU handles all of the class requests then packs the data into the correct format as expected by the device driver—this means that development of a video capture or recording device, for example, would need no driver development. In fact, Cypress has an accessory board that connects to an Aptina image sensor and an FX3 firmware example enabling an HD video camera to be designed with only modest effort. All of the source code and hardware design files are available and described in an Application Note AN75779, so if your sensor is a little different, then that is the only piece of the project that you need focus upon. This is a single chip sensor to USB UVC solution.

High-end video cameras use MIPI CSI-2 (Camera Serial Interface version 2) links. Cypress have a software compatible variant of the FX3, called the CX3, that uses a customized GPIF interface to collect up to four lanes of 1-Gbps MIPI CSI-2 data to support a wide range of industry-standard image formats and is capable of streaming uncompressed 4K UHD video at 15 fps or 1080P video at 30 fps. A reference design, shown in **Photo 2**, is available to get you off to a running start.

Cypress has another software compatible variant of the FX3 called the FX3S which has a different customized GPIF II interface specifically designed for mass storage devices. The FX3S has two storage ports which each supports SD 3.0, eMMC 4.41 and SDIO 3.0 standards. The FX3S supports multiple configurations including a single-chip RAID1 solution, this implementation is shown in **Photo 3** and is described in an Applications Note (AN89661) with all source code and hardware design files available for your customization.

If you look inside the currently available SuperSpeed camera products, you will discover a Cypress FX3 or derivative 95% of the time.

CONNECTING TO THE REAL WORLD

Coming back to the SuperSpeed Explorer Kit, it is most likely that you will connect an external FPGA to the GPIF II interface using the two 40-pin connectors. If you are an experienced FPGA designer then you

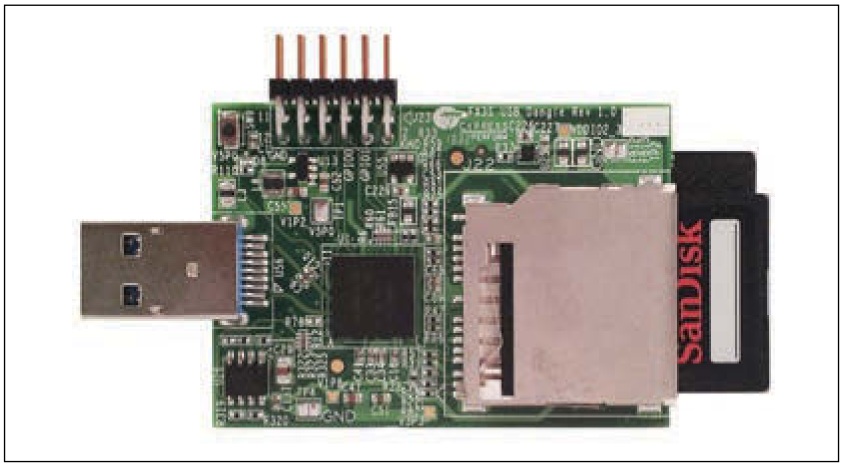


PHOTO 3
Single chip RAID1 reference design using FX3S (Source: Pactron (http://www.pactroninc.com)

are probably using a Xilinx Spartan 601 development board or an Altera Cyclone III development board. Cypress has adapter boards that enable either of the FPGA boards to be connected to the SuperSpeed Explorer kit. These are shown in **Photo 4a** and **Photo 4b**. If you are new to FPGAs then Cypress has a CPLD board, shown in **Photo 5**, that connects directly onto the SuperSpeed Explorer kit such that initial design ideas may be tried. The CPLD board contains a Xilinx CoolRunner XC2C256 CPLD and the WebPack tools are a free download from www.xilinx.com. This site also includes a wide variety of applications notes and training materials if you are new to the world of programmable logic. Also in development by a third party is an analog processing board, designed around a Cypress PSoC 5, which will form the basis of a high-performance analog data acquisition system.

I called the article "Rapid Prototyping" since all of the elements needed to create a working prototype are readily available for your customization. **Figure 2** shows the development of a SuperSpeed device divided into managable pieces. Windows is the primary development platform, but there is also support for Mac OS and Linux. A free Software Development Kit (SDK) is downloadable from www.cypress.com/fx3sdk. This SDK includes software examples for a Windows host, firmware examples for the FX3, and GPIF II Designer examples for the GPIF II interface. It is likely that the building blocks that you need are already available.

PROJECT DEVELOPMENT

Host application software uses the standard Windows Visual Studio environment and examples are provided for native C++ applications and managed C# applications. Cypress provides a device driver that supports fast data transfers using standard APIs and very fast data transfers using standard APIs with Overlapped IO operations. Several utilities are also provided including a USB Control Center that enables FX3 programs to be downloaded to the SuperSpeed Explorer kit for debug. I have provided Cypress with a collection of examples and these are also downloadable from the FX3 web page. One of the examples is a comprehensive Benchmark program that allows you to characterize the performance of your PC platform; this includes host controller, memory and disk performance. You will discover that the FX3 is never the bottleneck and can deliver whatever datarate the PC can support directly to the GPIF II interface and your hardware.

FX3 firmware development involves C programming with an FX3 Program Framework and uses an Eclipse Toolset with integrated GCC cross compiler plugins. The low-level timing details of the FX3's heavily-coupled units, especially the distributed DMA channels, require a lot of set up. Rather than burden the developer with these intricate, low-level details, Cypress provides a Real Time Operating System (RTOS) and a set of device drivers for all of the subsystems shown

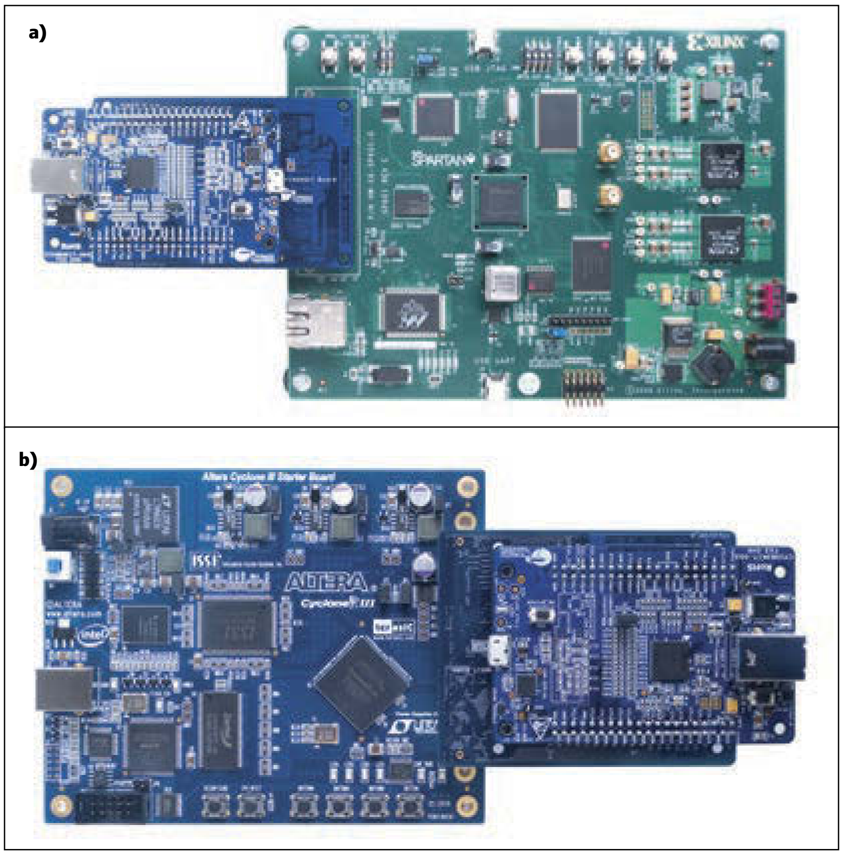


PHOTO 4
Commercial FPGA boards from Xilinx and Altera can be connected to the SuperSpeed Explorer Board (Source: Cypress Semiconductor www.cypress.com/fx3)

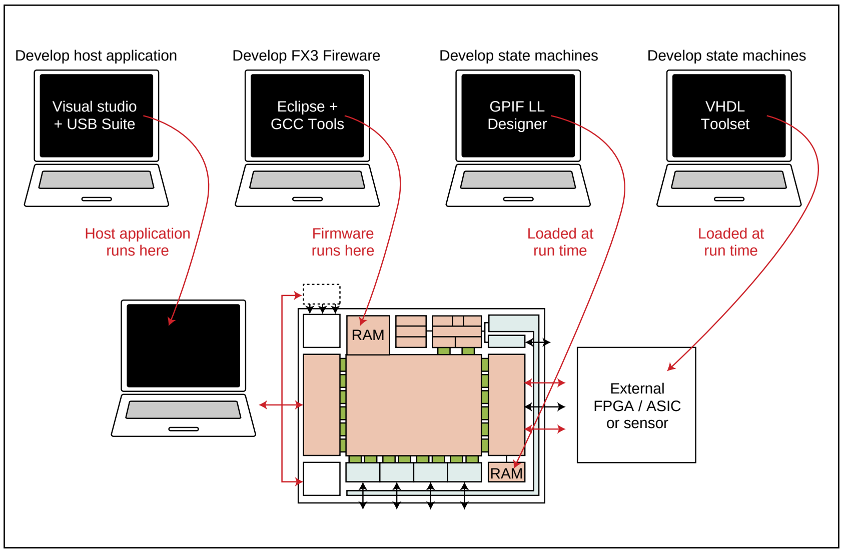


FIGURE 2
Four pieces of software are typically needed for a full system solution

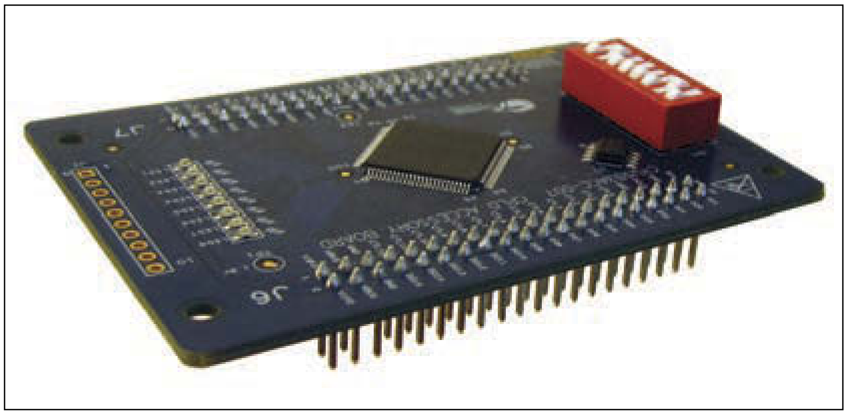


PHOTO 5
The CPLD Accessory Board plugs directly onto the SuperSpeed Explorer Board

in Figure 1. The RTOS is Express Logic's ThreadX (Version 5.1) and all of its features are imported into the FX3 environment. A high-level, block-oriented API Framework is presented to the developer so that focus can be placed on what your application will do and not on how it will do it. You can ignore the RTOS and just write your application as a single threaded user task or you could take full advantage of the real time capabilities and partition your application into several cooperating tasks. The choice is yours and I recommend starting with a single thread then move onto several threads as your applications needs evolve.

The FX3 Program Framework handles all of the USB communications. The RTOS USB driver does all of the necessary enumeration and run-time power management on your behalf. You provide the USB descriptors and set the power management policy. USB endpoints are connected to DMA channels and all data is buffered and delivered automatically by the DMA hardware. A USB class template describes what must be done if you need to develop your own class driver and, of course, several examples are included.

Most FX3 application programs consist of using the Framework APIs to set up the hardware units and then let them do their work. The CPU is notified of errors or completions and typically operates as a "traffic cop" controlling the high level policy and operation of the application. Having an operating system "underneath" the application, which handles all of the low-level details, makes FX3 firmware development more productive. Cypress supplies an integrated JTAG debugger which can set breakpoints and examine registers, variables or memory but I found that I did not use this. Since there is an RTOS scheduling multiple cooperating tasks then the last thing I want to do is to stop the CPU at a breakpoint! I found it more productive to have a debug monitor that runs as a task that is aware of the real time environment and RTOS data constructs. I implemented several monitor commands which query or set variables including I/O and GPIF state. One of the examples I wrote raised I/O lines when certain RTOS events happened, such as a thread starting or stopping, semaphore write etc. This allowed me to observe the operation of the running tasks using a logic analyzer.

A typical SuperSpeed device will need one or more high speed data channels and an additional control channel. A video camera, for example, may need pan and tilt or setup of camera features such as white balance. The FX3 has a "supporting-cast" of low-speed subsystems: UART, I2C, SPI and I2S. Each subsystem has additional hardware and a block-oriented, hardware interface such that the DMA controller can read or write them just like the USB and GPIF subsystems. You need not access data or status registers of these low-speed subsystems. The same DMA channel API is used to communicate with the low speed susbsystems and this reduces programming effort and debug time. Many concurrent DMA transfers are supported and the available bus bandwidth is allocated at 50% for the CPU (if it needs it, in general the CPU will be asleep), then each DMA channel is served with a round-robin priority scheme.

The UART is connected to an on-board UART-to-USB device so that all that is needed to use the debug monitor with a standard terminal emulation program, such as Clear Terminal or TeraTerm, is a USB cable. The FX3 includes a boot loader in ROM and can load its program from an EEPROM included on the SuperSpeed Explorer Kit or it can request a program be downloaded via USB from the computer host.

The CPLD board allows you to develop protocols with the GPIF II subsystem. The examples include basic counters so that the data path can also be verified. The code for the CPLD is written in a hardware descriptiopn language such as VHDL or Verilog. The examples provided by Cypress are in Verilog. The Xilinx toolchain "compiles" the Verilog

SOURCES

FX3 SuperSpeed device controller and GPIF II Master Interface

Cypress Semiconductor | www.cypress.com

ThreadX RTOS Ver. 5.1

Express Logic | www.rtos.com

CoolRunner XC2C256 CPLD

Xilinx | www.xilinx.com

circuitcellar.com/ccmaterials

code into an XSVF binary programming file. One of the Cypress FX3 examples is a CPLD_ Programmer which can take these XSVF files and can program the CPLD so no additional cables or hardware is required.

SYSTEM DEBUG

The SuperSpeed Explorer Kit has extended pins on its two 40-way connectors so that "test-points" for all of the GPIF signals are readily accessible (see Photo 1). A logic analyzer, or oscillscope, can be attached to these test points so that the communication protocol between GPIF and the real world external hardware can be observed and debugged. The GPIF clock can be run up to 100 MHz but it can, if necessary, be divided down to lower frequencies (as low as 100 kHz) so that low-cost logic analyzers, such as the popular USBee DX, can be used to debug the logic of the protocol. Once this logic is working the clock can be raised back to 100 MHz and USB-captured data files are used for final system verification. The examples, that can be downloaded from www.cypress.com/fx3, include a variety of simple data exchange programs, including counters, so that individual elements of the design can be tested and reliable data transfer verified.

This article has discussed the end-to-end development of a SuperSpeed USB device using the FX3-based SuperSpeed Explorer Kit. It is low cost and the software tools are free and I know that this will help SuperSpeed technology be more accessible to peripheral developers. If you have a lot of data that you have to move, or some data that is generated at 400 MBps rates, then it is time to look into the design of a SuperSpeed USB device. This is now readily achievable with the new toolset from Cypress and at power levels that are lower than an equivalent USB 2.0 solution.

Happy developing.

ABOUT THE AUTHOR

John Hyde is Principle at USB Design By Example, a consultancy firm he created after "retiring" from Intel in 2002. John has been involved in USB since its inception and wrote his first *USB Design By Example* book in 1999. He works with a wide variety of USB companies on the architecture, design, and debugging of their embedded products. He has written five "how-to" books and is now working on an update to his *SuperSpeed* book and a new *USB Type-C Design By Example* book. John earned a BSc in Electronics from Southampton University and now lives in Portland, OR.

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.

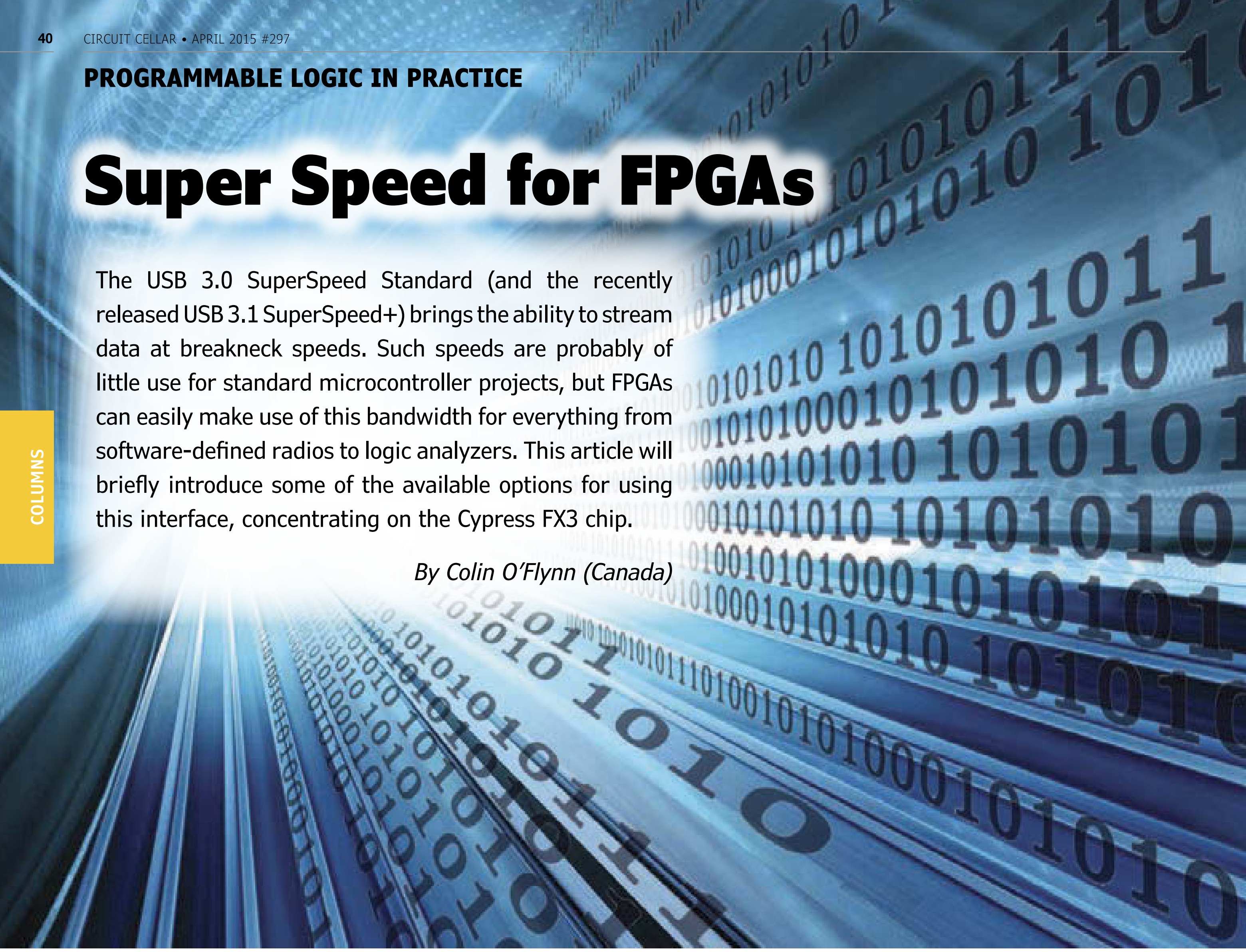
http://www.emacinc.com/products/system_on_module/SoM-A5D36

Since 1985 OVER 30 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • www.emacinc.com

PROGRAMMABLE LOGIC IN PRACTICE



Super Speed for FPGAs

The USB 3.0 SuperSpeed Standard (and the recently released USB 3.1 SuperSpeed+) brings the ability to stream data at breakneck speeds. Such speeds are probably of little use for standard microcontroller projects, but FPGAs can easily make use of this bandwidth for everything from software-defined radios to logic analyzers. This article will briefly introduce some of the available options for using this interface, concentrating on the Cypress FX3 chip.

By Colin O'Flynn (Canada)

Using USB 2.0 in your design is a fairly routine operation now. Not only are there a number of dedicated interface chips (such as made popular by FTDI), numerous microcontrollers are available with full- or high-speed USB 2.0 interfaces, many of which can also be interfaced to an FPGA through external memory support of the microcontroller.

USB 2.0 high speed has a maximum theoretical throughput of 480 Mbps, or 60 MBps. If you are trying to quickly stream data, you might find that inadequate. Streaming two channels of ADC data for software defined radio (SDR) applications can easily use more bandwidth than that, especially if you are trying to use a 10- or 12-bit ADC over a fairly wide bandwidth.

USB 3.0 and 3.1 jump the maximum throughput to 5 and 10 Gbps, respectively (or 625 MBps and 1.25 GBps). At such data rates, you've got to ask what happens on the other side. These rates are much faster than your typical hard drive can deal with—which means you're either just buffering to RAM, computing on the fly with incoming data, or have a much more expensive storage solution!

Of course your application might not need such fast data rates (nor should you necessarily expect to achieve them), but it might require something a little above USB 2.0 high speed. I expect many applications to fall into this category, aiming to achieve around 100 to 400 MBps. In this article I'll briefly introduce what's new in USB 3.0, discuss debugging your design, and outline your options for adding a USB 3.0 interface to your FPGA project.

It's also worth noting that USB 3.0 increases the amount of power that can be provided to 900 mA, up from the 500 mA USB 2.0 would allow. Even if your device doesn't need USB 3.0 speeds, it may require USB 3.0 power.

This article is designed to give you a brief introduction to tools you can use to design your own USB 3.0 device. Some recent product releases have made USB 3.0 development accessible to everyone from novices to full-time engineers, and I'm going to highlight my experiences with them.

SPEEDING AWAY

So what is USB 3.0 SuperSpeed? You might have noticed the blue USB SuperSpeed ports on your new laptop, and if you look closely at them, you would notice another row of contacts "behind" the normal USB contacts, which mate with a cable as shown in **Photo 1**. I'd like to take a moment to appreciate that the protocol designers are sending a 5-Gbps signal over an external cable that might be up to 3 m long—which puts things in perspective next time you are doing a high-speed digital PCB, wondering about running some signals more than a few centimeters!

Physically, you might notice USB 3.0 cable in Photo 1 has both transmit and receive data pairs, allowing the protocol to operate in full-duplex mode. With USB 2.0, this wasn't possible, meaning that if you need to stream data in both directions simultaneously, USB 3.0 contains an even larger speed boost than you might expect.

With USB 3.1 a new connector type called Type C was introduced—consumers will appreciate that the cable is reversible—no more figuring out which way is "up" on the USB connector. This connector is also used for providing up to a 10-Gbps transfer rate, and a special "power delivery" mode allows up to 100 W of power. To limit connector and wire current, the 100-W power transfer occurs at 20 V instead of the more usual 5 V. This article will be strictly limited to USB 3.0, but if you need either the higher power delivery or speed of USB 3.1, I wanted you to be aware of its existence.

If you are interested in more details of USB 3.0, I highly recommend Donovan Anderson and Jay Trodden's book, *USB 3.0 Technology*. But you can also download the full USB specification from the USB Implementers Forum, which includes considerable amount of 'plain English' background and information in addition to the detailed specifications.

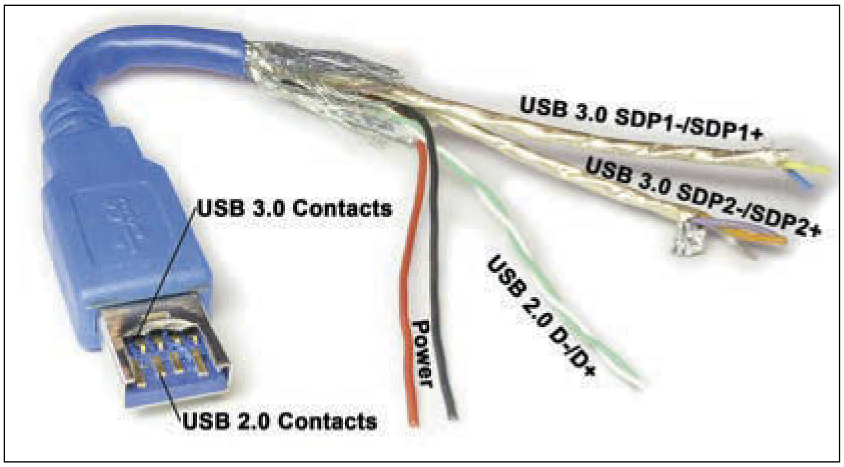


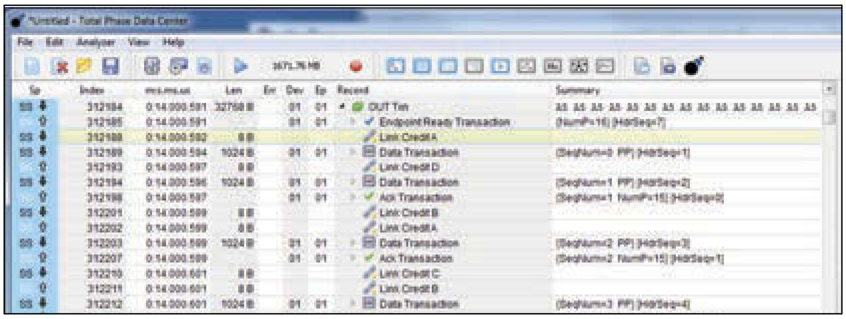
PHOTO 1
I cut away a cheap USB 3.0 cable to show you the second row of connectors used for high-speed data transfer. When plugged into a USB 2.0 port the USB 3.0 contacts won't be used; the existing USB 2.0 contacts allow full compatibility of old devices with new computers and vice-versa. With USB 3.0 two Shielded Data Pairs (SDP) are present, as one is dedicated for transmit and one is dedicated for receive.

DEBUGGING IN SUPERSPEED

As in USB 2.0 designs, I consider having a hardware protocol analyzer a necessity. If you are using well-defined example code you might be able to get away without one, but my experience is that hardware protocol analyzers very quickly pay for themselves in rapid troubleshooting of USB interface problems.

I'm using a TotalPhase Beagle 5000 protocol analyzer (see **Photo 2**), although there are several other vendors of USB protocol analyzers too. A view of typical USB 3.0 data transfer is shown in **Photo 3**, which

PHOTO 2
The hardware setup used here is a Cypress FX3 SuperSpeed Explorer Kit far left, with a TotalPhase Beagle 5000 V2 hardware analyzer used to snoop the USB 3.0 traffic. The SMA connector on the Beagle 5000 is the trigger out which will be used to precisely determine the location of error events.



*Untitled - Total Phase Data Center

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Summary
SS	312184	0:14.000.591	32768 B		01	01	OUT Txn	A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5 A5
	312185	0:14.000.591			01	01	Endpoint Ready Transaction	[NumP=16] [HdrSeq=7]
SS	312188	0:14.000.592	8 B				Link Credit A	
SS	312189	0:14.000.594	1024 B		01	01	Data Transaction	[SeqNum=0 PP] [HdrSeq=1]
	312193	0:14.000.597	8 B				Link Credit D	
SS	312194	0:14.000.596	1024 B		01	01	Data Transaction	[SeqNum=1 PP] [HdrSeq=2]
	312198	0:14.000.597			01	01	Ack Transaction	[SeqNum=1 NumP=15] [HdrSeq=0]
SS	312201	0:14.000.599	8 B				Link Credit B	
	312202	0:14.000.599	8 B				Link Credit A	
SS	312203	0:14.000.599	1024 B		01	01	Data Transaction	[SeqNum=2 PP] [HdrSeq=3]
	312207	0:14.000.599			01	01	Ack Transaction	[SeqNum=2 NumP=15] [HdrSeq=1]
SS	312210	0:14.000.601	8 B				Link Credit C	
	312211	0:14.000.601	8 B				Link Credit B	
SS	312212	0:14.000.601	1024 B		01	01	Data Transaction	[SeqNum=3 PP] [HdrSeq=4]

PHOTO 3
Monitoring a data transfer using the TotalPhase Beagle 5000 shows us some details of the USB 3.0 data transfer process. Significant changes exist at lower layers which you can see here, but at higher layers the same basic ideas are used (such as interfaces and endpoints) as in USB 2.0.

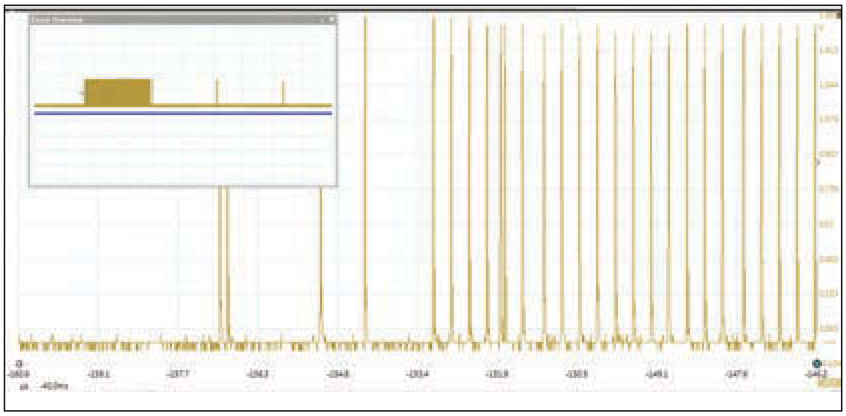


PHOTO 4
As in USB 2.0, having a hardware protocol analyzer is useful when errors could be caused by anything from improper PCB layout to software issues to power supply problems. Here I'm using the 'Trigger Out' feature to mark whenever physical-layer (PHY) errors occur, allowing me correlate this burst of errors with other issues such as a large load switching nearby.

was captured running the streaming example on the FX3 board. In this capture you can see the new link layer in USB 3.0, which has four buffers accessible from the link layer (i.e., before being sent to the protocol layer). The "Link Credit" messages you see on the USB analyzer window is flow control being performed for those buffers, as the chip advertises the availability of buffer space.

One feature to look for on your USB analyzer is a "trigger out." This allows you to correlate events on the USB trace with real life. In **Photo 4**, I've set the trigger out to pulse on physical-layer errors. You can see a huge jump in errors during a short time, and having the view on the oscilloscope helps you determine if these errors come up due to power supply transients, perhaps due to a large load your device is switching. Or if you are debugging your FPGA interface, this "trigger out" can help you trigger other instruments such as the ChipScope Analyzer. For more information, refer to my article, "Using Internal Logic Analyzers for FPGAs" (*Circuit Cellar* 279, 2013). I previously discussed some more in-depth uses of these trigger features in my 2010 article, "Advanced USB Design Debugging" (*Circuit Cellar* 241).

INTERFACE OPTIONS

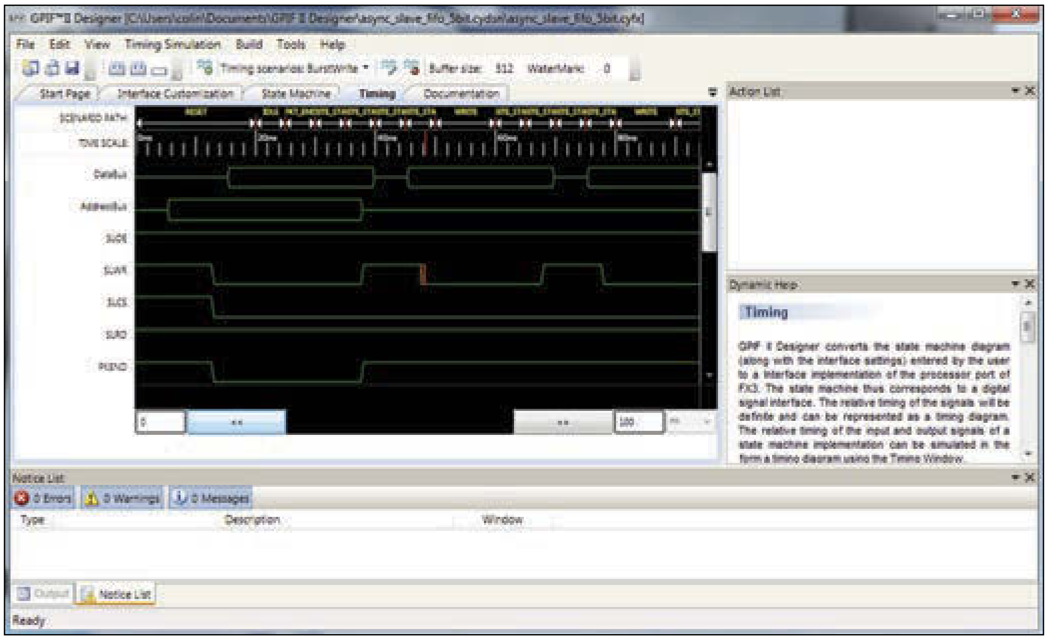
When it comes to USB 3.0 interfaces, you have a few immediate options. One of the most popular is a single-chip solution by Cypress Semiconductor, the EZ-USB FX3 device. If you are familiar with Cypress's line of EZ-USB interface chips for USB 2.0, it will be a familiar idea—a high-speed microcontroller with USB 3.0 interface, which has a flexible high-speed parallel interface that can interface to many different types of parallel busses. The FX3 series upgrades from an 8051 core to an ARM926EJ core running at 200 MHz, with 256 or 512 KB of SRAM.

Of the options I'm presenting, this device is the easiest to interface to your design. The parallel interface can run in synchronous or asynchronous mode, and you can choose between multiple data bus widths. The maximum bandwidth over the external interface is 3.2 Gbps (32 bits at 100 MHz).

It's worth emphasizing that you get a very serious micrcontroller with your USB 3.0 interface—the FX3 device should be considered as a fairly integral part of your design, as you'd hate to waste the available ARM9 processor! The FX3 devices are relatively costly—about $20 in quantities of 100—but you get a lot of silicon for the price (see **Figure 1**).

A more complicated, but cheaper, option is the USB3380 chip from PLX Technology. The USB3380 chip is a PCIe-to-USB 3.0 bridge, which can work in either direction. This means you can use it to drive a PCIe card over USB 3.0, or you can appear as a USB 3.0 peripheral controlled by PCIe. The second option is of particular interest: many FPGAs provide a PCIe interface block, which provides the physical interface required by the USB3380 chip. The supporting documentation and examples are considerably less complete, and you will have much work ahead of you in getting the PCIe interface talking to the USB3380 chip. (This comes from my own experiences using the device, not just reading the press releases.) The upside is the lower cost: the USB3380 is around $12 in 100-piece quantities.

If your FPGA has sufficiently fast transceivers, you can also run the USB 3.0 core directly on the FPGA. Various third-party cores are available at considerable cost; but amazingly, an open-source USB 3.0 core is available as part of Project Daisho, the USB 3.0 core in this project having been written by Marshall H. This targets an Altera Cyclone IV FPGA, and requires a USB 3.0 transceiver


GPIF™II Designer [C:\Users\colin\Documents\GPIF II Designer\async_slave_fifo_5bit.cydsn\async_slave_fifo_5bit.cyfx]
File Edit View Timing Simulation Build Tools Help
Timing scenarios: BurstWrite
Buffer size: 512 WaterMark: 0
Start Page
Interface Customization
State Machine
Timing
Documentation
Action List
Dynamic Help
Notice List
0 Errors
0 Warnings
0 Messages
Type
Description
Window
Output
Notice List
Ready

Timing

GPIF II Designer converts the state machine diagram (along with the interface settings) entered by the user to a interface implementation of the processor port of FX3. The state machine thus corresponds to a digital signal interface. The relative timing of the signals will be definite and can be represented as a timing diagram. The relative timing of the input and output signals of a state machine implementation can be simulated in the form a timing diagram using the Timing Window.

PHOTO 5
While I don't expect you to be able to see the details of the GPIF II Designer software here, it does demonstrate the convenience of having 'live' timing diagrams which would match your specific interface requests, with the ability to switch between read, write, and burst examples.

physical interface, such as the TI TUSB1310A (about $10 USD in Qty 100).

A final option to investigate is the FTDI FT600/FT601. They are a line of USB 3.0 interface chips in a slightly more prototype friendly package compared to the FX3 (QFN instead of BGA), but with a more rigid interface design. The external 16/32-bit FIFO-style interface appears almost identical to their earlier USB 2.0 products, and with a maximum throughput of 400 MBps for the 32-bit bus (same as the FX3). At the time I wrote this article, only preliminary datasheets were available, and no pricing information was available.

The remainder of this article will look only at the Cypress FX3. My assumption is that for many designs the engineering effort is a considerable portion of the overall cost. The slightly higher cost of the FX3 device will be offset by the greatly reduced engineering effort (as will be described), and the fairly simple interface requirements of the FX3 allow you to avoid the need to select an FPGA with Gigabit transceivers. The FTDI device may end up being an interesting contender too (having similar interface requirements), but it was only just being released as I wrote this article, so I cannot comment on real-world usability of this device.

DEVELOPING WITH THE FX3

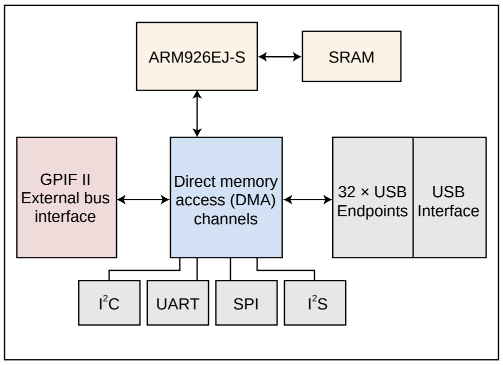
Cypress recently released their USB SuperSpeed Explorer Kit, which is a $49 kit using the CYUSB3014 EZ-USB FX3 device, shown previously in Photo 1. In tandem with this, you can use the book *SuperSpeed Device Design by Example* by John Hyde, which contains extensive and well-documented examples for this development board. This combination means for $80 you can get a complete hardware development kit and printed reference, along with a software development environment, debugger for the microcontroller, and this ecosystem has several expansion boards for connections to FPGAs and CPLDs. While I hate to write a column which simply says "go buy this for more details," this combination of an excellent reference book with low-cost hardware is impossible to resist. I had been meaning to write a column on USB 3.0 for some time now, but having bought this combination myself, it really drove home that development for USB 3.0 doesn't have to be complex!

For connecting the FX3 evaluation board to your FPGA, you can buy interposer boards for some Xilinx and Altera FPGA boards, or spin your own if it doesn't fit the available options. As mentioned previously, the FX3 is designed to work using the DMA to shuffle data between the USB endpoints and your external interface. For FPGA interfacing the

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.

FIGURE 1
The block diagram of the Cypress FX3 shows that it is primarily designed around a high-speed Direct Memory Access (DMA) engine; a necessity to keep up with potential USB 3.0 data flows. External connections are omitted on this diagram for clarity.


ARM926EJ-S
SRAM
GPIF II External bus interface
Direct memory access (DMA) channels
32 × USB Endpoints
USB Interface
I²C
UART
SPI
I²S

```
>>> import usb.core
>>> import usb.util
>>> dev = usb.core.find(idVendor=0x04b4, idProduct=0x00F1)
>>> print dev
<usb.core.Device object at 0x0225EE10>
>>> dev.set_configuration()
>>> dev.read(0x81, 1024)
array('B', [170, 170, 170, 170, 170, 170, 170
, 170, 170, 170, 170, 170, 170, 170, 170, 170
... many more lines ...
, 170, 170, 170, 170, 170, 170, 170, 170, 170
, 170, 170, 170, 170, 170, 170, 170, 170, 170])
```

LISTING 1
Communicating with USB 3.0 devices can use a standard interface library, such as this example using libusb in Python to read data from a specific endpoint.

"address data" interface is fairly standard, and makes it easy to map registers within the FPGA, along with performing bursts of data transfer.

As the FX3 provides other standard peripherals (SPI, I²C, and UART) you also have the option of using the external interface for high-bandwidth communication, and using another interface for low-bandwidth applications such as setting registers. Or if your application requires the configuration of an ADC or DAC over I²C/SPI, this can be done directly from the FX3, requiring you to only design the data conversion blocks in the FPGA. The only caveat here is the device on the explorer kit does not support the SPI interface when a 32-bit external bus is being used.

Cypress provides the GPIF II Designer software which helps you configure the device for any specific external bus setup, such as address-data or FIFO. This includes detailed timing diagrams as shown in **Photo 5**, which simplifies the design of your FPGA interface code. It's also worth noting that *SuperSpeed Device Design by Example* includes Verilog source code for Master and Slave FIFO interfaces, which can help jump-start your FPGA design. This source code is freely available from Cypress on the companion website for the book.

The output of the GPIF II Designer only provides you with configuration settings you can load for the interface; it doesn't tell the chip how data will be sent over the interface. Several examples are provided with the FX3 Software Development Kit (SDK) demonstrating the use of DMA to shuffle data between the USB endpoints and the external interface, and if you need even more examples see the *Device Design by Example* book.

Finally, you'll need the computer interface side. A DLL is provided with high-level interface examples for C++ and C#, which is used by the examples from Cypress and in the *Design by Example* book. If using another language such as Python you can typically still access this DLL—for example, using ctypes on Python. (Refer to my blog post at ProgrammableLogicInPractice.com for more details.) But if you just need to sling data between USB endpoints, you can use generic interfaces such as the open-source libusb. This has the advantage of having existing interfaces for almost any programming languages. **Listing 1** shows an example of using Python to read from the "Streaming" example that comes preprogrammed on the FX3 Explorer Kit. This example simply continuously sends data over the USB 3.0 connection, but you can see it's trivial to read data from these endpoints; in this case, I'm reading 1,024 bytes being sent from the FX3 device.

FINISHING TOUCHES

In this column I can only touch on the basics of USB 3.0 SuperSpeed development. But hopefully, this introduction will demonstrate that this new standard is in reach of almost any embedded engineer, thanks to the release of copious examples and low-cost development tools.

As usual I'll post more detailed links and videos to the companion website at www.ProgrammableLogicInPractice.com, although a few of the critical ones are listed in the Resources section of this article too.

circuitcellar.com/ccmaterials

RESOURCES

D. Anderson and J. Trodden, *USB 3.0 Technology*, MindShare Press, 2013, www.mindshare.com/Learn/USB_3.1/Books.

Cypress Semiconductor, FX3 Resource Page, www.cypress.com/fx3/.

J. Hyde, *SuperSpeed Device Design by Example*, CreateSpace, 2014. Download examples from the book online at www.cypress.com/fx3book.

Project Daisho (includes open-source USB 3.0 controller), https://github.com/mossmann/daisho.

SOURCES

FX3 SuperSpeed device controller
Cypress Semiconductor | www.cypress.com

TotalPhase Beagle 5000 USB SuperSpeed Analyzer
Total Phase | www.totalphase.com

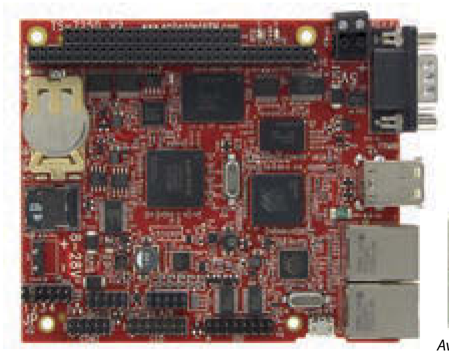
SUPERIOR EMBEDDED SOLUTIONS

Technologic Systems

DESIGN YOUR SOLUTION TODAY
CALL 480-837-5200

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)

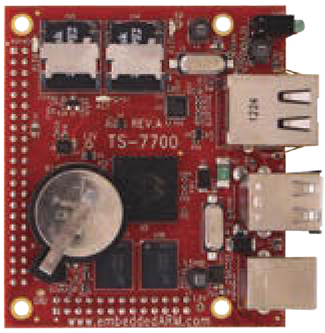
Features:

- 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

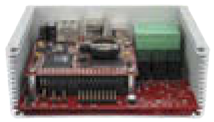
Benefits:

- 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 ºC to 85 ºC industrial temperature range

TS-7700 Embedded Board
High Performance Compact SBC

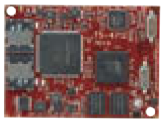


Pricing Starts At
$119 Qty 100
$165 Qty 1



Available with TS-ENC750 enclosure and TS-752 interface board

- 800 MHz or 1 GHz Marvell PXA168 ARM CPU
- 512 MB DDR3 RAM, 128 bytes NVRAM
- 2x microSD Card Sockets for Redundant Storage
- 8 KLut FPGA, 1x 10/100 Ethernet, 2x USB Host
- 55x 3.3V DIO, 6x TTL UARTs, 1x I2C, 1x 12S, 1x CAN
- On board temperature sensor and watchdog timer
- -40 ºC to 85 ºC Industrial Temperature Range
- Runs Linux 2.6.34 or 3.14 with Debian Wheezy



Computer-on-Modules
State of the Art Embedded Design

- TS-4600: 450 MHz low cost w/ 2 Ethernets
- TS-4710: Up to 1 GHz PXA168 w/ video
- TS-4720: Like TS-4710 + 2 GB eMMC Flash & 2 Ethernets
- TS-4200: Atmel ARM9 w/ super low power
- TS-4800: 800 MHz FreeScale iMX515 w/ video

LONG LIFE — We've never discontinued a product in 30 years

RUGGED — Embedded systems that are built to endure

OPEN — Support every step of the way with open source vision

ORIGINAL — Unique embedded solutions add value for our customers

www.embeddedARM.com

THE CONSUMMATE ENGINEER

Software FMEA/FMECA

Toward Better Software Development Predictability

The analytical methods of failure modes effects and criticality analysis (FMECA) and failure modes effects analysis (FMEA) have been around since the 1940s. In recent years, much effort has been spent on bringing hardware related analyses such as FMECA into the realm of software engineering. This month, George takes a close look at software FMECA (SWFMECA) and its potential for making software development more predictable.

By George Novacek (Canada)

The roots of failure modes effects and criticality analysis (FMECA) and failure modes effects analysis (FMEA) date back to World War II. FMEA is a subset of FMECA in which the criticality assessment has been omitted. Therefore, for simplicity, I'll be using the terms FMECA and SWFMECA only in this article. FMECA was developed for identification of potential hardware failures and their mitigation to ensure mission success. During the 1950s, FMECA became indispensable for analyses of equipment in critical applications, such as those occurring in military, aerospace, nuclear, medical, automotive, and other industries.

FMECA is a structured, bottom-up approach considering a failure of each and every component, its impact on the system and how to prevent or mitigate such a failure. FMECA is often combined with fault tree analysis (FTA) or event tree analyses (ETA). The FTA differs from the ETA only in that the former is focused on failures as the top event, the latter on some specific events. Those analyses start with an event and then drill down through the system to their root cause.

In recent years, much effort has been spent on bringing hardware related analyses, such as reliability prediction, FTA, and FMECA into the realm of software engineering.[1, 2, 3] Software failure modes and effects analysis (SWFMEA) and software failure modes, effects, and criticality analysis (SWFMECA) are intended to be software analyses analogous to the hardware ones. In this article I'll cover SWFMECA as it specifically relates to embedded controllers.

Unlike the classic hardware FMECA based on statistically determined failure rates of hardware components, software analyses assume that the software design is never perfect because it contains faults introduced unintentionally by software developers. It is further assumed that in any complicated software there will always be latent faults, regardless of development techniques, languages, and quality procedures used. This is likely true, but can it be quantified?

SOFTWARE ANALYSIS

SWFMECA should consider the likelihood of latent faults in a product and/or system, which may become patent during operational use and cause the product or the system to fail. The goal is to assess severity of the potential faults, their likelihood of occurrence, and the likelihood of their escaping to the customer. SWFMECA should assess the probability of mistakes being made during the development process, including integration, verification and validation (V&V), and the severity of these faults on the resulting failures. SWFMECA is also intended to determine the faults' criticality by combining fault likelihood with the consequent failure severity. This should help to determine the risk arising from software in a system. SWFMECA should examine the development process and the product behavior in two separate analyses.

First, Development SWFMECA should address the development, testing and V&V process. This requires understanding of the software development process, the V&V techniques and quality control during that process. It should establish what types of faults may occur when using a particular design technique, programming language and the fault coverage of the verification and validation techniques. Second, Product SWFMECA should analyze the design and its implementation and establish the probability of the failure modes. It must also be based on thorough understanding of the processes as well as the product and its use.

In my opinion, SWFMECA is a bit of a misnomer with little resemblance to the hardware FMECA. Speculations what faults might be hidden in every line of code or every activity during software development is hardly realistic. However, there is resemblance with the functional level FMECA. There, system level effects of failures of functions can be established and addressed accordingly. Establishing the probability of those failures is another matter.

The data needed for such considerations are mostly subjective, their sources esoteric and their reliability debatable. The data are developed statistically, based on history, experience and long term fault data collection. Some data may be available from polling numerous industries, but how applicable they are to a specific developer is difficult to determine. Plausible data may perhaps be developed by long established software developers producing a specific type of software (e.g., Windows applications), but development of embedded controllers with their high mix of hardware/software architectures and relatively low-volume production doesn't seem to fit the mold.

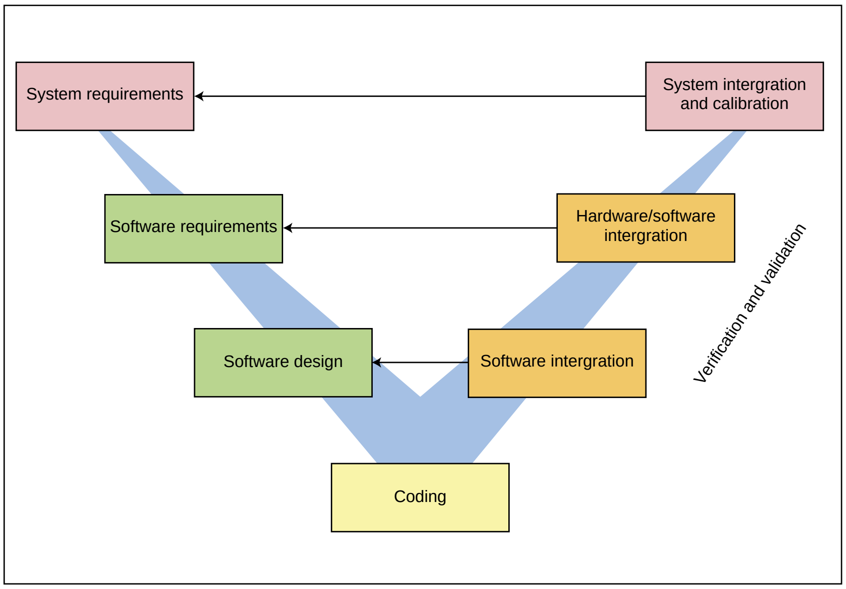
Engineers understand that hardware has limited life and customers have no problem accepting mean time between failures (MTBF) as a reality. But software does not fail due to age or fatigue. It's all in the workmanship. I have never seen an embedded software specification requiring software to have some minimum probability of faults. Zero seems always implied.

SCORING & ANALYSIS

In the course of SWFMECA preparation, scores for potential faults should be determined: severity, likelihood of occurrence, and potential for escaping to the finished product.[4] The scores between 1 to 10 are multiplied and thus the risk priority number (RPN) is obtained. An RPN larger than 200 should warrant prevention and mitigation planning. Yet the scores are very much subjective—that is, they're dependent on the software complexity, the people, and other impossible to accurately predict factors. For embedded controllers the determination of the RPN appears to be just an analysis for the sake of analysis.

Statistical analyses are used every day from science to business management. Their usefulness depends on the number of samples and even with an abundance of samples there are no guarantees. SWFMECA can be instrumental for fine-tuning the software development process. In embedded controllers, however, software related failures are addressed by FMECA. SWFMECA alone cannot justify the release of a product.

FIGURE 1
Software development "V" model


System requirements
System intergration and calibration
Software requirements
Hardware/software intergration
Software design
Software intergration
Coding
Verification and validation

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@nexicom.net with "Circuit Cellar"in the subject line.

EMBEDDED SOFTWARE

In embedded controllers, causes of software failures are often hardware related and exact outcomes are difficult to predict. Software faults need to be addressed by testing, code analyses, and, most important, mitigated by the architecture. Redundancy, hardware monitors, and others are time proven methods.

Software begins as an idea expressed in requirements. Design of the system architecture, including hardware/software partitioning is next, followed by software requirements, usually presented as flow charts, state diagrams, pseudo code, and so forth. High and low levels of design follow, until a code is compiled. Integration and testing come next. This is shown in the ubiquitous chart in **Figure 1**.[5]

During an embedded controller design, I would not consider performing the RPN calculation, just as I would not try to calculate software reliability. I consider those purely statistical calculations to be of little practical use. However, SWFMECA activity with software ETA and FTA based on functions should be performed as a part of the system FMECA. The software review can be to a large degree automated by tools, such as Software Call Tree and many others. Automation notwithstanding, one should always check the results for plausibility.

TOOLS

Software Call Tree tells us how different modules interface and how a fault or an event would propagate through the system. Similarly, Object Relational Diagram shows how objects' internal states affect each other. And then there are Control Flow Diagram, Entity Relationship Diagram, Data Flow Diagram, McCabe Logical Path, State Transition Diagram, and others. Those tools are not inexpensive, but they do generate data which make it possible to produce high-quality software. However, it is important to plan all the tests and analyses ahead of the time. It is easy to get mired in so many evaluations that the project's cost and schedule suffer with little benefit to software quality.

The assumed probability of a software fault becomes a moot point. We should never plunge ahead releasing a code just because we're satisfied that our statistical development model renders what we think is an acceptable probability of a failure. Instead, we must assume that every function may fail for whatever reason and take steps to ensure those failures are mitigated by the system architecture.

System architecture and software analyses can only be started upon determination that the requirements for the system are sufficiently robust. It is not unusual for a customer to insist on beginning development before signing the specification, which is often full of TBDs (i.e., "to be defined"). This may be leaving so many open issues that the design cannot and should not be started in earnest. Besides, development at such a stage is a violation of certification rules and will likely result in exceeding the budget and the schedule. Unfortunately, customers can't or don't always want to understand this and their pressure often prevails.

The ongoing desire to introduce software into the hardware paradigm is understandable. It could bring software development into a fully predictable scientific realm. So far it has been resisting those attempts, remaining to a large degree an art. Whether it can ever become a fully deterministic process, in my view, is doubtful. After all, every creative process is an art. But great strides have been made in development of tools, especially those for analyses, helping to make the process increasingly more predictable.

circuitcellar.com/ccmaterials

RESOURCES

G. Novacek, "Failure Mode and Criticality Analysis," *Circuit Cellar* 270, 2013.

———, "Product Reliability (Parts 1 and 2)," *Circuit Cellar* 268–269, 2012.

———, "Software Reliability," *Circuit Cellar* 273, 2013.

———, "Software Safety," *Circuit Cellar* 285, 2014.

R. W. Stoddard, ASQ Reliability Division Webinar, http://media.asq.org/113401/web.mp4.

Experience extraordinary training for embedded systems professionals at ESC Conference Series 2015!

The **Embedded Systems Conference (ESC)** is the industry's largest, most comprehensive technical conference for embedded systems professionals in the U.S. ESC is excited to announce an expanded 2015 USA Conference Series, taking place in **Boston (May 6-7), Silicon Valley (July 20-22)**, and **Minneapolis (November 4-5)**.

esc

EMBEDDED SYSTEMS CONFERENCE '15

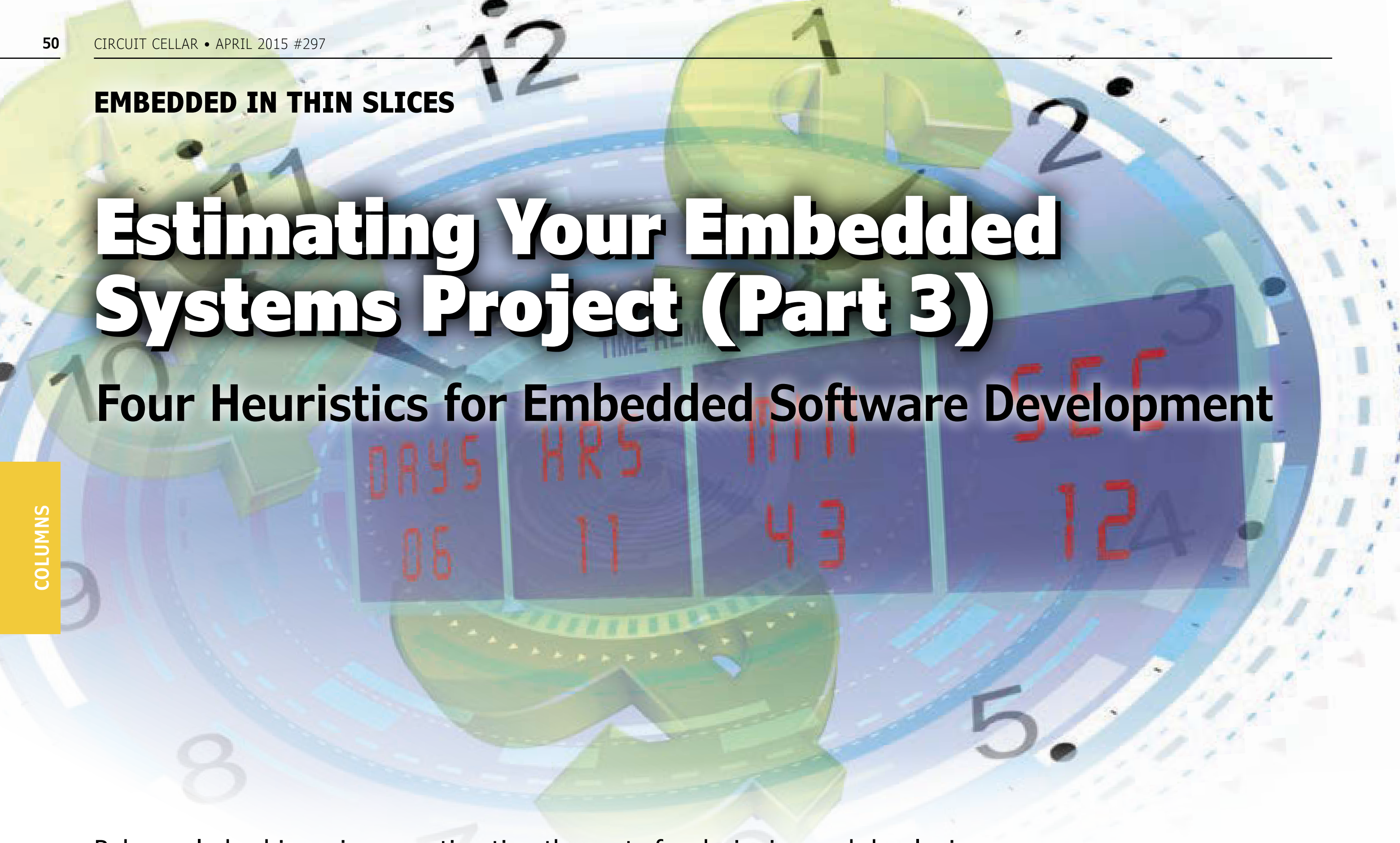
SILICON VALLEY

The ESC series continues in Silicon Valley, a leading hub for high-tech innovation and development; its home to many of the world's largest high tech corporations, as well as thousands of high-tech startup companies. Silicon Valley boasts more engineers per square foot than anywhere else on earth.

Come join us at the Santa Clara Marriot and check out some of our sponsors for the event:

Rohde & Schwarz | **All Quallity & Service, Inc. (AQS)** | **AdaCore** | **Altium, Inc.** **AMP Display** | **Azul Systems, Inc.** | **Chefree Technology Corp** | **Code** | **Data Image Corporation** | **EBS Net Inc.** | **EMA Design Automation** | **STMicroelectronics** | **Teledyne LeCroy** | **Toradex Inc.** | **WolfSSL** | **Embedded Works** | **Green Hills Software, Inc.** | **IEEE** **Ironwood Electronics** | **Lauterbach** | **Micro Computer Control Corp.** | **Pico Technology** **Rigol Technologies** | **Gemalto** | **Silex Technology America** | **Symmetry Electronics** **The QT Company** | **Trusted Computing Group**

Register today at www.embeddedconf.com/silicon_valley with promo code **CircuitCellar15SV** and get 15% off an All Access Pass. The offer ends July 15, 2015.



Estimating Your Embedded Systems Project (Part 3)

Four Heuristics for Embedded Software Development

Bob concludes his series on estimating the costs for designing and developing your embedded systems project. He looks at four heuristics and how by knowing them you can get better at this task.

By Bob Japenga (US)

"People who spend their time, and earn their living, studying a particular topic produce poorer predictions than dart-throwing monkeys." This quote from a Nobel prize winner who knows his stuff is a jarring introduction to our final installment in our article series dealing with estimating the cost and schedule of our embedded software systems. Is this whole process of software estimation no better than what dart-throwing monkeys could come up with? In the opening article, I said that accurate estimating is extremely difficult. But I also said that there is some hope. This month I would like to provide some thin slices of help for anyone who is asked to estimate how many man hours it will take to create an embedded system or even a part of an embedded system. And the help will come from the author of that quote.

Daniel Kahneman is a psychologist who won the Nobel Prize in Economics in 2002. The above quote is taken from his 2012 book entitled *Thinking, Fast and Slow*, which summarizes his decades of research on the psychology of judgment, decision-making, and behavioral economics. Kahneman has provided some key insights into how we approach the impossible task of estimating costs of developing embedded software systems.

This article will not delve into function points, use-case points, software metrics, COCOMO models, SEER-SEM, or any of the other methods for estimating software. These are good, useful, and well documented in the literature. We use function points, use-case points and software metrics in our company. I would heartily recommend that you understand function points, use-case points, and develop software metrics. In particular, develop software metrics that relate to your experience of function points or use-case points. In other words, count the function points or use-case points during the estimation phase of three to four projects and see what you learn about yourself, your estimating process, and your projects. Go back to completed projects and determine the number of function points or use-case points and factor that into a metric.

What I want to do this month is to look at estimating from 5,000′. I want to see how Kahneman's research can help us become better at estimating software. The stated purpose of his book was, in his words, to "learn to recognize situations in which mistakes are likely and try harder to avoid significant mistakes when stakes are high." If we can glean that from his book, we will become better at estimating the costs of embedded software systems.

Kahneman proposes almost 50 heuristics in his book. A heuristic is a method or process that enables us to learn something on our own. Only a few of them are applicable to us in estimating software. But if we can master them, they will enable us to become better at estimating embedded software.

PRIMING

Sometimes one of our customers will tell us that a project needs to be completed in three months. Or that it needs to be completed for under $15,000. These numbers can have a very bad effect on our ability to accurately estimate a software project. I am amazed how often my estimate closely parallels the customer's estimate. Can it be that the customer really knows how long it is going to take or how much it is going to cost? Or is something else going on?

A heuristic discussed by Kahneman in his book is called priming. He and other researchers have demonstrated that our behavior can be primed by what goes immediately before us. For example, he cites one study, where two groups of young people (aged 18–22) were asked to form some sentences from a set of five words. One group had a set of five words associated with the elderly. After the exercise, the young people were asked to walk through a corridor. Those who worked with words about the elderly walked slower than the other group! Experiments like this have been repeated many times with a wide variety of different priming mechanisms. The evidence seems to indicate that we are deeply influenced by priming.

How are we to use this knowledge about ourselves to become better at estimating? First, as much as possible, we need to avoid obtaining from our customer or bosses expected costs and schedule before we make our estimates. Estimating is difficult and I really want to know what the customer or my boss expects me to estimate. But resist the urge. Priming is a powerful and proven effect and we must avoid it as much as possible. Watch out when your boss tells you that he needs this in two weeks and then asks you to estimate it.

If however, the cat is out of the bag, we need to make an extra effort to not let that number influence us. This is by far the harder of the two options. Once primed, even when I take herculean strides to not be influenced, the priming has its effect. But at least I am aware of the effect. Develop your estimate with your usual method of function points or use-case points or whatever, and if it comes in the same ball park as the "primed" number, be wary of your numbers and extra-vigilant—run your numbers again.

ANCHORING EFFECT

I have noticed that a lot of my estimates seem to be remarkably similar to previous estimates. Could it be that my projects are so similar that it always takes 40 hours to write the software specification for all projects? Or that user interface designs always take 160 man hours. Or is something else at work?

One of the heuristics Kahneman has identified is what he calls the anchoring effect. The anchoring effect "occurs when people consider a particular value for an unknown quantity before estimating that quantity." With serious academic rigor, Kahneman demonstrates how we are influenced by previous numbers. For example, he tells us that if we were asked if we thought that Gandhi was 114 years old when he died, we would immediately say "No." If we were then asked how old we thought he was when he died, our number would be higher than if we were first asked if we thought Gandhi was 35 years old when he died. That first number acts as an anchor to pull our estimate in its direction. Kahneman's claims that this phenomenon is "one of the most reliable and robust results of experimental psychology: the estimates stay close to the number that people considered [previous]—hence the image of an anchor." This means that we will be affected by it when we do our estimating.

Anchoring is closely related to priming. I would make the distinction that priming involves numbers related to the estimate (the customer's estimate for the same project). Anchoring happens when I take numbers from an unrelated project into account before I make my estimate.

How can we take this knowledge and become better at estimating? Here is where software metrics come into play. Look back over the last several estimates of unrelated projects. Were the estimates similar to each other? How did the actuals compare to the estimates? If you see a correlation with the estimates but not in the actuals, anchoring is a possible cause. Develop a range of estimates based on actuals and use these when making a new estimate. For example, imagine that the user interface took 120 hours on project A, 130 hours on project B, and 250 hours on project C. Attempt to identify the similarities and the differences and place a weighted value to each. How many menu items or screens were involved? Was it a graphical interface?

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.

Was a working driver provided? Was it a touch screen or keypad or both?

To avoid the anchoring effect on new estimates, we need to ruthlessly dissect previous projects into their subcomponents using software metrics. In other words, keep track of how long it took to write the specification, design the user interface, design the manufacturing test fixture, etc. Then when we approach a new project we need to make a quantitative comparison between the smaller elements. For example, the user interface is about twice as complex as project B and half as complex as project C. This quantitative approach can help us avoid the anchoring effect.

OPTIMISTIC BIAS

Kahneman describes many biases that affect our ability to estimate. He posits that the optimistic bias may be the most significant. In my earlier articles in this series, I discussed optimism as it relates to estimating but it bears repeating. I would recommend reading chapters 23 and 24 of Kahneman's book in an attempt to hammer home how pervasive this heuristic is and learn to make adjustments.

How do we counter this optimistic bias? I would say that a thorough knowledge of our optimistic bias is a good start. Software metrics can help if you develop actual numbers and then compare them to your estimates during a post-mortem. But human nature such as it is, unless we learn to develop a sort of "humility before the data" we can ignore the stubborn facts the metrics show us. A simple mantra that could be said after you have completed your estimate and before you submit it, is to repeat these words: *All evidence shows that I am repeatedly over optimistic in what I think I can do. How should this estimate change based on that?*

SMALL SAMPLE SIZE

We all know that using a small sample of data sets us up for errors in estimating or drawing conclusions. But Kahneman takes us to a new level of awareness of the danger of using small samples. For example he cites a study of the incidence of kidney cancer in 3,141 counties in the United States. The counties with the lowest incidence per capita are "mostly rural, sparsely populated, and located … in the Midwest, the South and the West." Upon hearing this, most of us immediately start jumping to conclusions. But he goes on to also state that counties with the highest incidence per capita are also mostly rural, sparsely populated, and located in the Midwest, the South and the West. I leave it to you to figure out why sample size is the reason for this apparent contradiction. Email me if you want some help.

In using our software metrics to estimate embedded software systems, we have to recognize that we have an extremely small sample size that we are drawing upon. I have been in this business since 1973. I have estimated almost 1,000 such projects. Yet even with all that experience, that is an extremely small sample to accurately predict how the next project is going to go.

So what can be done in light of this last heuristic? I recommend that you doggedly pursue from other companies the results of actual projects. Most companies are not willing to part with this information. But I have found that it doesn't hurt to ask. In non-competing situations, we can learn a lot as we expand our sample. There are a lot of numbers floating around the web. Take the time to create your own data base of "the other guy's" actual development time.

On one project, we had expended an immense amount over our original estimates. After the project we found that another company designed a very similar product and took 2× to 3× as much calendar time and cost. Had we had that number in the beginning, we may have been more accurate in our original estimates.

Although your environment is unique to you and your company, industry standard metrics like hours/line of code and hours/function point or hours/use-case point can help expand your sample. These metrics are prone to many errors and are widely variable. Nonetheless they are another data point for your estimate. They help us limited engineers do the impossible: expand our sample without actually doing the work.

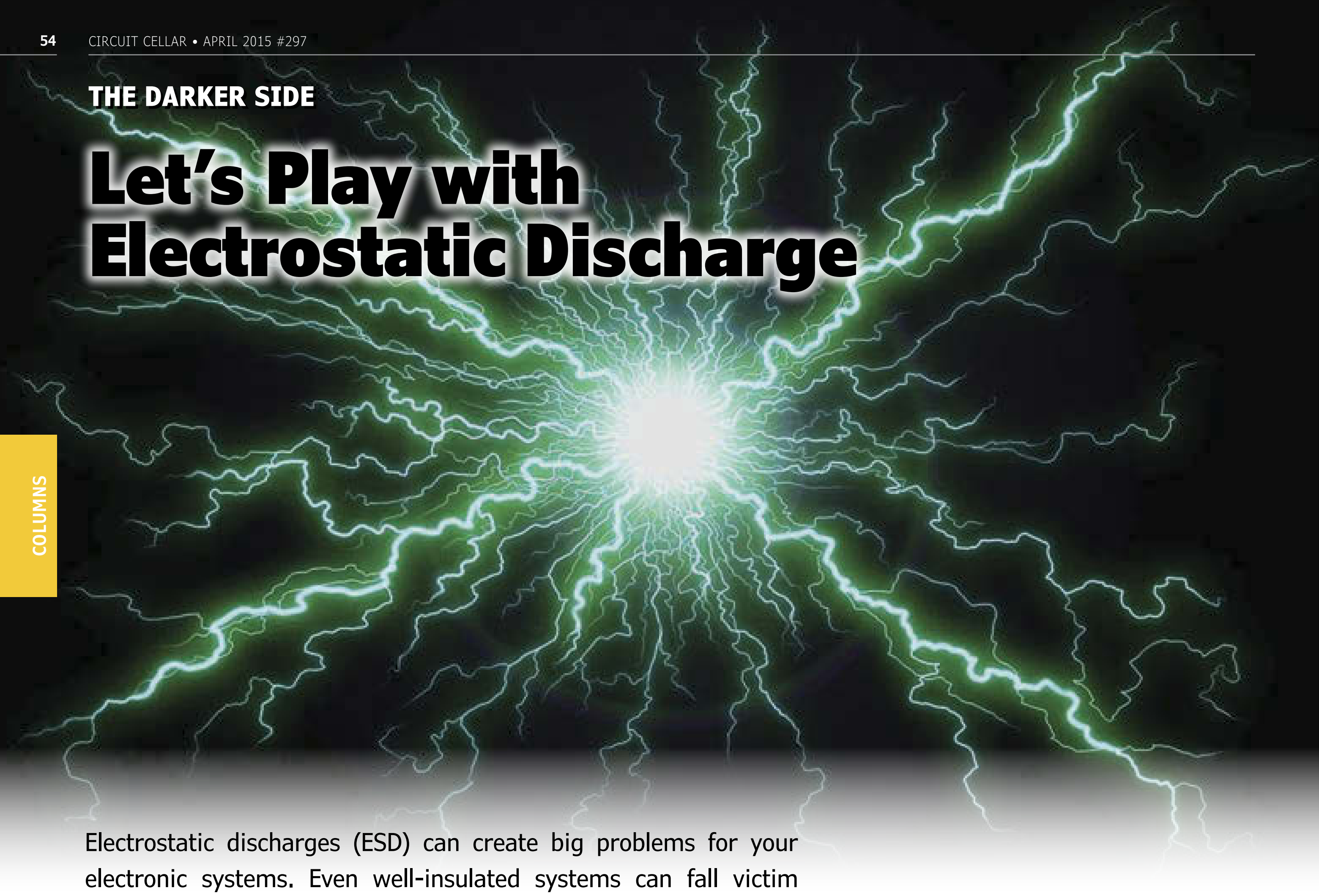
HOW ACCURATE?

Accurately estimating embedded software systems is impossible. Don't let anyone tell you otherwise. Hopefully, with some input from this series, you will become a little better at it than you were before. And that is no small accomplishment. It is with some reluctance that I close this article series since I know that I have taken a very thin slice of a very big topic. Email me if you would like me to elaborate more fully on any of these topics in the coming months.

circuitcellar.com/ccmaterials

RESOURCES

D. Kahneman, *Thinking, Fast and Slow*, Farrar, Straus, and Giroux, New York, NY, 2012.



THE DARKER SIDE

Let's Play with Electrostatic Discharge

Electrostatic discharges (ESD) can create big problems for your electronic systems. Even well-insulated systems can fall victim to ESD. Robert details several "ESD events" and provides tips for protecting your projects.

By Robert Lacoste (France)

Welcome back to the Darker Side column. Every child is amazed by sparks. I leave near Paris, France, and we have old science museum called Palais de la Découverte. You might have something similar close to you. I remember that I was more than astonished the first time I visited it and saw an electrostatic experiment. A young girl was installed in a Faraday cage, and exposed to 1,000,000,000-V electrostatic discharges! The sparks were several feet long, and her hair was, well, strangely straight.

Electrostatic discharge, or ESD for short, is not only impressive. It's also the cause of plenty of failures in electronic circuits. Simply speaking, a 5- or 3.3-V circuit isn't suppose to survive to 1,000 V. But think about it twice: such a voltage is very common between your body and the ground. Have you ever walked on a carpet? Just touch any electronic part, or any connector not sufficiently protected, and you will have a dead piece of silicon. Moreover, even if your product is well insulated, nearby ESD can easily create strong induced currents, which can either destroy something or at least generate nasty reset pulses to an on-board microcontroller.

Even some experienced electronic designers are ignoring ESD. That's risky. ESD events exist, so you'd better be prepared. In addition, regulatory groups often require ESD testing before selling products. This is true in Europe with CE marking requirements. (Refer to my December 2011 column on the subject in *Circuit Cellar* 257.) And, last but not least, dealing with ESD can be really fun!

THE ORIGIN OF ESD

As Wikipedia explains, the main contributor to ESD is triboelectric charging. Some nonconductive materials (e.g., skin, glass, mica, hair, and paper) love to give away electrons, while other materials (e.g., acrylic, PVC, polyester, and Teflon) prefer to receive them. John Carl Wilcke published the first ranked list of such materials in 1757. Take a pair of these materials and put them in contact. There will be a slight transfer of electrons between them, according to their preferences. Next, quickly move them away from each other. As they are insulators, the electrons are not moving that quickly, so one of the materials stays positively charged and the other remains negatively charged. That's triboelectricity. The accumulated charge Q (in coulombs) is simply the number of transferred electrons multiplied by the charge of an individual electron (i.e., 1.6×10^{-19} C). Next, move this charge between the electrodes of a capacitor C (in farads). A voltage V (in volts) will appear between these electrodes. And the relationship between these three values is simply $V = Q/C$. So, the voltage will be higher is the charge is higher or if the capacitor is lower valued.

That's exactly how the first friction-based electrostatic generators worked. Take a glass disk, rotate it over a woolen cloth, add a collecting conductor and a storage capacitor, and you will have recreated the early 18th-century generators. Rubbing the two surfaces creates successive contact/release between the materials as they have a rough surface. These generators were then improved up to the Van De Graaff and Tesla generators just before 1939.

The bad news is that exactly the same phenomenon exists between your shoes and the carpet, or between a plane and the surrounding air (which is also an insulator). And this explains the sparks that appear at the end of your finger when you want to open your car after exiting your home, especially if the weather is dry. According to Henry W. Ott in *Electromagnetic Compatibility Engineering* (which is one of my favorite books), walking across a carpet can generate voltages up to 35,000 V if the ambient humidity is lower than 20%. Come close to any grounded conductor and you will have a nice spark. Do you know what such a high-voltage spark looks like? I can't easily generate 35,000 V, but I do have a 25,000-V Shaffner NSG432 ESD generator in my lab, which is close. I switched it on at full voltage and spent some time to get a good picture for you (see **Photo 1**). Impressive, isn't it?

THE HUMAN BODY MODEL

ESD events are difficult to reproduce as the voltages and capacitor values are varying. In order to facilitate the job of the designers and testers, standards were fortunately defined. The idea is to specify a reasonable model for the prime source of ESD concerns: humans. As usual, several models exist, but one of the most common is the one specified in the European standard EN61000-4-2. In this one, the human body is assumed to be equivalent to a 150-pF capacitor with one grounded terminal and the other connected to the test finger through a 330-Ω resistor. The capacitor is initially charged to a voltage of 4- to 15-kV depending on the test and product specifications.

Where do these values of 150 pF and 330 Ω come from? The model assumes that the capacitor formed between each of our feet and the ground is about 50 pF (see **Figure 1**). There is also the so-called free-space capacitance of the body. (That is the capacitance when the second electrode is an infinity. There are more details in Ott's book.) For a human, this free space capacitance is about 50 pF too. Add some surrounding walls and you will understand that the 150 pF of the standard is not unreasonable. Lastly, the 330-Ω resistor simulates the resistance of the skin (which is, of course, far from stable).

BASIC TEST SETUP

I know that you prefer experiments over long talks, so I built one for you (see **Photo 2**).

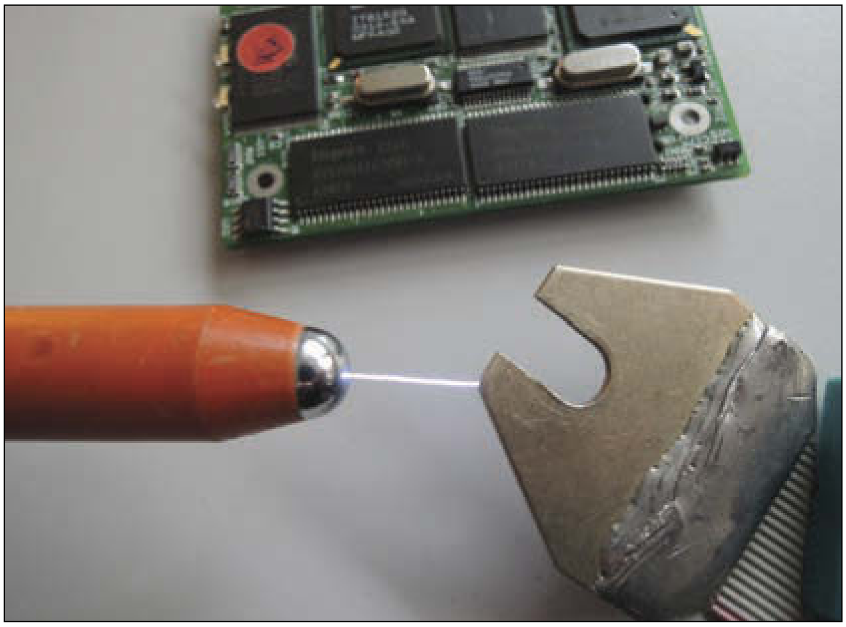


PHOTO 1
A 25,000-V spark is up to 1-cm long. Nice, isn't it ?

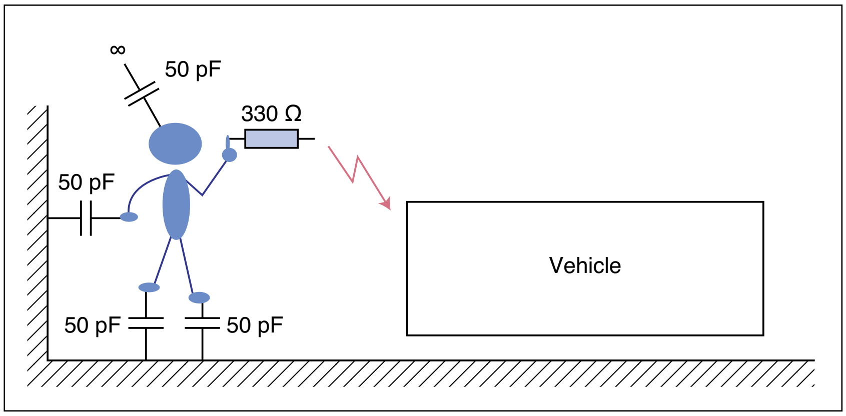


FIGURE 1
The human body model assumes a total capacitance of 100 to 300 pF to the ground and a contact resistance of some hundred ohms. Note that 150 pF and 330 Ω are standardized values.

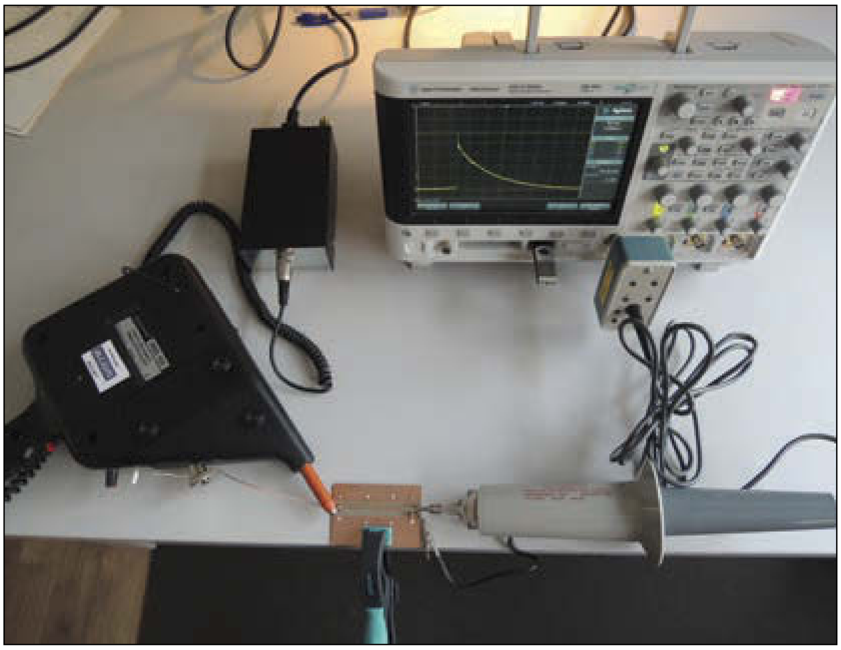


PHOTO 2
My test setup includes an NSG432 generator (left), a small test PCB, a DSO-X 3024 digital scope (top), and an impressive 20-kV 1:1000 Tektronix P6105 probe (bottom right).

I etched a simple 2" long track on a PCB. (The other side is a full ground plane.) I connected my NSG432 ESD generator to one end of the track and set it to 8 kV. I soldered a 47-kΩ load resistor on the other end to simulate a load, and I measured the voltage across this resistor with my Keysight Technology DSO-X 3024A digital scope. (Keysight Technology is the new name of Agilent instruments, which was the new name of Hewlett Packard instruments. It's difficult to follow.)

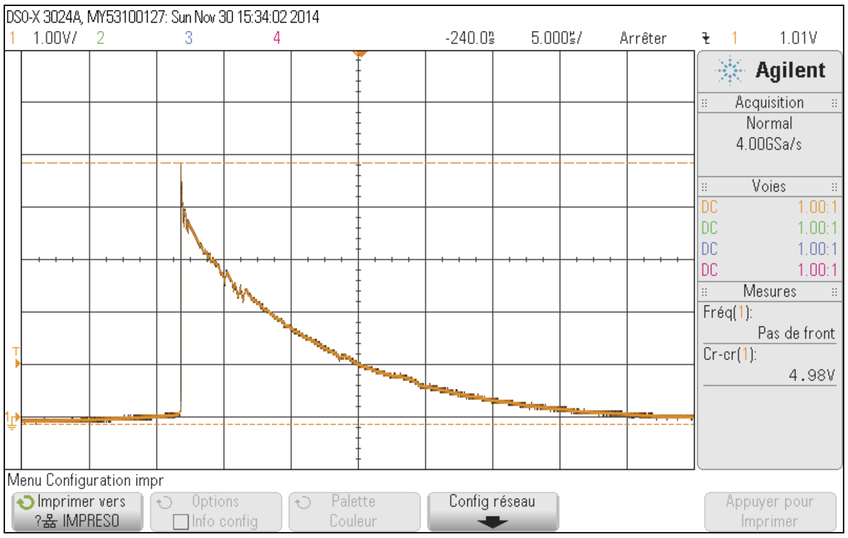


PHOTO 3
The ESD surge, measured without any protection devices, is up to 5 kV. The horizontal scale is here 5 µs per division.

Of course, connecting directly a 8-kV generator to the delicate input of a $4,000 scope would not be a clever idea, except may be if you are looking for an excuse to buy a new scope! So, I used an old but impressive Tektronix P6015 high-voltage 1:1000 attenuation probe, which is also visible in Photo 2. I got this monster on eBay a while ago and was happy to find a use for it. This probe is even supplied with a bottle of fluorocarbon gas to fill it when you want to use it close to its 20-kV limit. Probably not environmental friendly, but impressive.

Then it was time to test. I configured the scope for one-shot triggering, kept my hands off, and fired the ESD gun. The good news is that the scope survived, and I got the plot you see in **Photo 3**. The maximum measured voltage was 5 V, which means 5 kV due to the 1:1000 attenuation of the probe. This is not too far from the 8-kV setting of the ESD generator. The difference is due to the impedance of the PCB track, which tamper the very fast positive edge of the electrostatic discharge. The horizontal scale is 5 µs per division, which means that the ESD-generated voltage stayed above 1 kV for about 13 µs. So, an ESD event is quite fast. Do you remember that an energy is a power times a duration? This implies that the energy of an ESD spike is reasonably low, even if it is largely enough to destroy virtually any unprotected electronic component. Remember these key characteristics of ESD: voltage is very high, duration is very short, and energy is quite low. This is very different from other kind of surges. In particular, ESD must not be confused with lightning. When a line receives (directly or not) a lightning blast, the voltage is usually lower than ESD, but its duration is far longer, so energy is far higher. Not the same subject.

ESD PROTECTION DEVICES

OK, let's come back to ESD. How can you protect a product from ESD events? Using ESD protection devices, of course. They come in different flavors. For low-power applications, the most common is the so-called transient voltage suppressor diode, or TVS for short. ("Transil" is, as far as I know, an equivalent denomination.) A TVS diode is a kind of specially designed Zener diode. As a Zener, a TVS has a specified clamping voltage and conduct when a higher voltage is applied on its terminals. In order to protect from spikes, they must be very fast, and they are. Their response times are measured in tens of picoseconds, as long as they are well used. More on that later. Both unipolar and bipolar variants exist, and you must select the appropriate one for your circuit. The unipolar

TVS should be used on a signal with is always above ground level, like a TTL input. The bipolar TVS includes two Zener diodes and has a symmetrical behavior. It can protect a line which accept a positive or negative voltage input. For example, **Photo 4** shows you a TVS design kit from Würth Electronik, but you can find also similar devices from companies like Vishay, Littlefuse, Texas Instruments, and tens of others.

Here are a few notes of caution. Firstly, you must select a TVS with a voltage rating appropriate to your circuit. The margin should be high enough to avoid any functional degradation. Usually, the threshold voltage with be twice higher than the nominal line voltage. The goal of a TVS is to avoid spikes of some kilovolts, not to protect a 5-V input from a 10-V DC voltage. That should be managed by other components if required. Wolfgang Kemper's 2010 Texas Instruments application report, "Reading and Understanding an ESD Protection," is an excellent reference, which describes all the parameters of a TVS diode.

Note that specific TVS exists for fast I/O lines like USB or Ethernet. You'd better use them if you have these kind of interfaces. Most importantly, you must take care to reduce as much as possible the parasitic inductance of the TVS pins connections. On any product, the TVS diode must always be installed as close as possible to the signal input. The PCB track from input to TVS diode must be minimal. As critical is the ground connection. It should be as short as possible and directly connected to the ground input, which means directly connected to the ground pin of the input connector. A point of emphasis: Any extra millimeters on the TVS connections to either signal input or ground return will drastically reduce the efficiency of the protection. And a final note of caution: TVSes are quite resistant; but if they fail, they usually fail as short circuits.

Other ESD protection devices are available, even if they are not as typical as TVSes. The well-know metal-oxide varistor (MOV) is an example. A varistor is a voltage-dependent resistor; its resistance become drastically lower above a threshold voltage. As compared to a TVS, a varistor can absorb far higher energy levels, but it is also far slower (some microseconds). Unfortunately, ESD events are very fast and the voltage can be up to some kilovolts in nanoseconds. Therefore, varistors are not very efficient for ESD protection. They are, however, perfect for lightning protection. Gas discharge tubes are also protection devices. They are, like varistors, mainly used for lightning protection, but they provide a very low parasitic capacitance, which makes them a great choice for antenna protection in particular.

PHOTO 4
ESD protection devices come in different small packages. This is a design kit from Wüth Elektronik.

LET'S TRY !

OK. It's time to switch on the soldering iron one more time. I took my test board and added a small TVS diode from the Würth Elektronic WE-VE series in a surface-mounted 0603 package. I soldered it directly on the PCB track, and connected its ground pin to the ground plane on the bottom side of the PCB through a via (see **Photo 5**). I zapped the input with my ESD generator once more and looked at the oscilloscope plot (see **Photo 6**).

The effect of the TVS diode is more than impressive. Firstly the spike maximum voltage

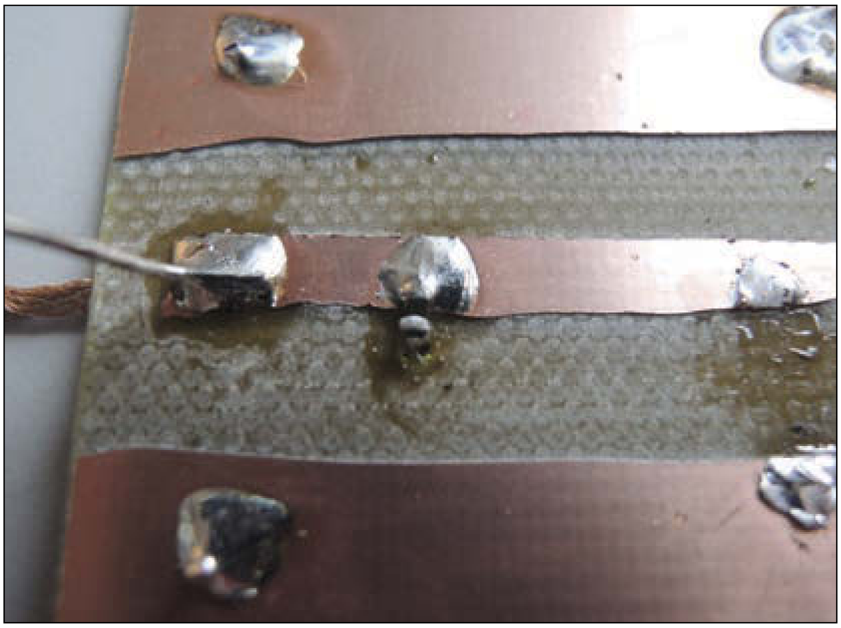


PHOTO 5
A TVS diode must be positioned as closely to the input as possible and with very short return paths. Here the tiny 0603 SMT TVS diode is connected directly to the bottom ground plane layer thanks to a via.

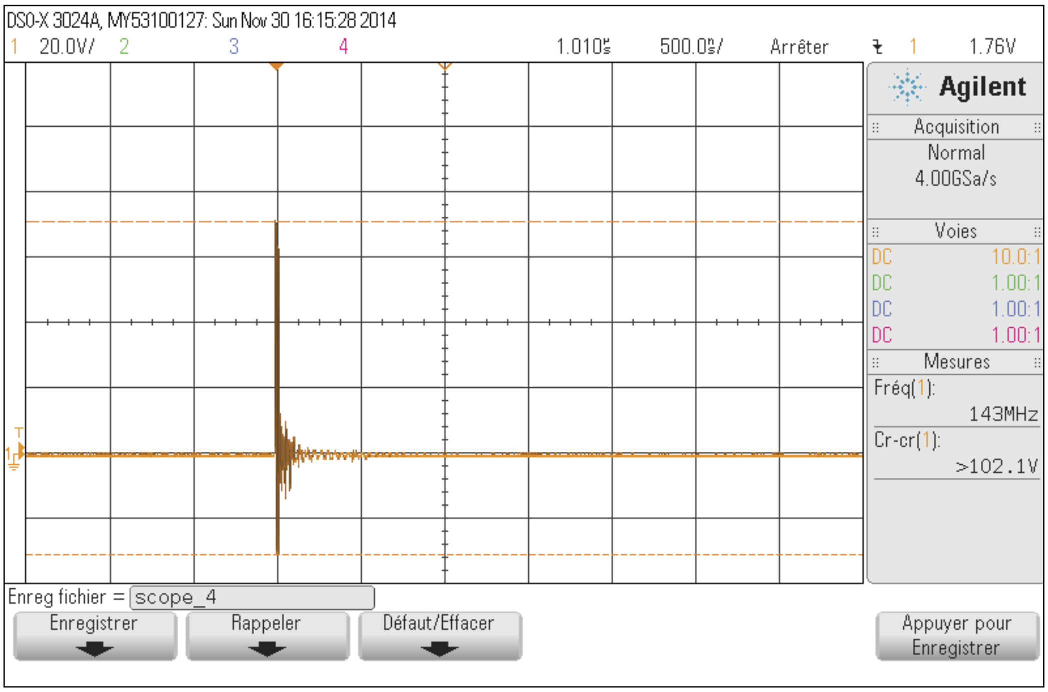


PHOTO 6
With a TVS the waveform is completely different. The maximum voltage is reduced to 100 V and the time scale is far shorter. Here the horizontal scale is 500 ns per division.

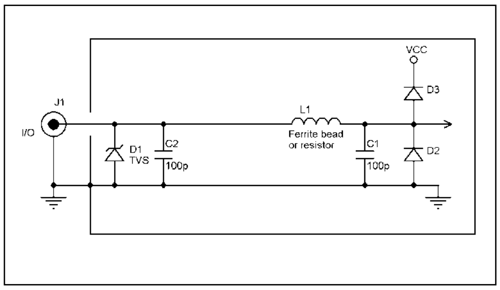


FIGURE 2
An ESD protection requires a TVS as close as possible to the input. A filtering network (R/C or L/C) close to the load reduces high-frequency spikes. Lastly, a small capacitor close to the input reduces transmitted perturbations but this has nothing to do with ESD.

was reduced from 5,000 V down to 100 V. That is 50× lower. Even more important, the duration of the ESD event, as seen by the receiving circuit, is now far shorter. Just compare with the first plot (see Photo 3), where the duration was around 15 µs. It is now down to 100 ns or so, which is 150× lower. Some calculations? Once again, an energy is the product of a power and duration. And do you remember that power is proportional to the square of a voltage? This means that the energy of the spike was roughly reduced by a factor of (50 × 50) × 150, which is 375,000×. A significant energy reduction with a small 0603 TVS diode, isn't it ?

ESD-SAFE DESIGNS

So, TVS diodes are the key components to add to your design if you suspect that an input line could be ESD prone. But in some cases, the remaining energy still will be too high for sensitive electronic devices. How can you ensure it will be harmless? First, refer back to Photo 6. You can see that the remaining energy is now a kind of fast oscillation, meaning that the TVS diode converted the ESD pulse into a kind of high frequency burst. How can you reduce it? With a low-pass filter, of course. Just add, after the TVS diode, a serial resistor and a parallel capacitor to ground and you will have a low pass RC filter, which will drastically reduce the remaining spike. Or, even better, replace the resistor by a ferrite bead to implement an LC filter and you'll have the close-to-perfect ESD protection you see in **Figure 2**.

circuitcellar.com/ccmaterials

RESOURCES

W. Kemper, "Reading and Understanding an ESD Protection," Texas Instruments, 2010, www.ti.com/lit/an/slla305/slla305.pdf.

A. Khan, "Electrostatic Discharge (ESD) Tutorial," Cypress Semiconductor Corp., 2012, www.cypress.com/?docID=35736.

H. W. Ott, *Electromagnetic Compatibility Engineering*, Wiley, 2009.

SOURCES

DSO-X 3024A Digital oscilloscope
Keysight Technologies | www.keysight.com

NSG432 Static discharge simulator
Schaffner | www.schaffner.com

P6015 High voltage probe
Tektronix | www.tektronix.com

82350120563 WE-VE ESD Suppressor
Würth Elektronik | http://katalog.we-online.de/en/pbs/WE-VE

You might think that this schematic is far too complex for the job. It could be the case for your design, but I want to present you with the safest approach. All the details are important. Firstly, as discussed, the TVS diode (D1) must be the first component on the signal path as it must be as close as possible to the input. Its ground connection should be as short as possible and directly connected to the signal ground input. If the enclosure is a metallic box, then grounding it at this point is probably the best choice. Forget the capacitor C2 for the moment. Then, the low-pass filter is made with the ferrite choke L1 and the capacitor C1. On the schematic, I intentionally put this L1/C1 low-pass filter quite far from the input components. Why? Simply because it's extra protection. Every inch of track between the TVS diode and the circuit to be protected will drastically reduce the ESD transient, as it will act as a good inductor. If you have some spare room on your circuit board, you should try to do the same.

The L1/C1 filter will reduce the remaining spike energy down to reasonable levels. The difficulty with such a circuit is that it is impossible to predict the amplitude of the remaining pulse with real-life ESD events. May be it will be very low, or may be it will still be some tens of volts, depending on the characteristics of the electrostatic discharge. But, wait. Now you are sure that the energy will be low, thanks to D1, and that the surge will be slow, thanks to L1 and C1. So now any classic over-voltage protection method is applicable. You can add a standard Zener diode to limit the spike voltage to any threshold you want, or you can simply add two diodes D2 and D3 to limit the voltage excursion from ground to VCC as shown.

Now lets go back to C2. This capacitor is as close as possible to the input. It has nothing to do with ESD but to electromagnetic compatibility (EMC). Its role is to make a low pass filter with L1, but this time in the reverse direction. It is filtering out noise coming from the internal circuit and will avoid it to leak externally. You may need it or not. It's your game.

WRAPPING UP

Electrostatic discharge causes problems for engineers. In this article, I only briefly covered the topic. Once again I strongly recommend you to read Ott's book and the other articles listed in Resources section of this article.

Even if they are different phenomenons, ESD and EMC have something in common. The difficulties associated with them are much greater if the ground is not well managed. Grounding is fundamental. A TVS diode won't help if the PCB doesn't provide a good, full ground plane. Nor will it help if the PCB ground is not closely linked to the input and enclosure ground. Do you want an example of what could happen if you include a PCB in a metallic box with external wires but without a proper ground connection? **Photo 7** shows what's referred to as a secondary spike. A first spike is created by an ESD event on the metallic enclosure. The voltage of the enclosure climbs to some thousand volts, and another spike fires between any internal part of this enclosure and the electronic circuit! OK, for this picture I intentionally used a worst-case situation, but this happens. Just add a solid ground connection between the PCB and the enclosure and you will avoid it.

Here we are. ESD is fun topic to study. But but it sure can be a nasty reality to deal with from time to time.

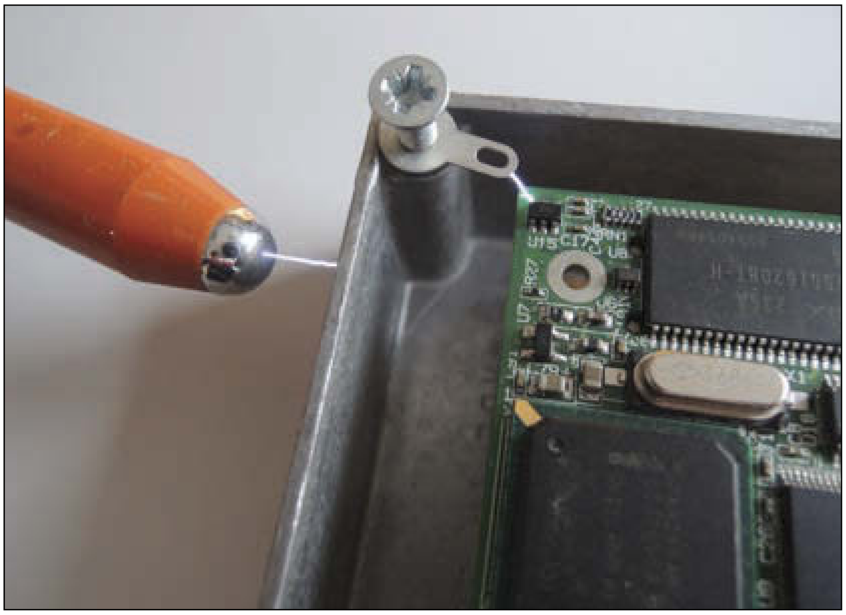


PHOTO 7
An illustration of indirect ESD spikes. Here you see that a first spike is created between the gun and the enclosure. A second spike appears between the enclosure and the electronic board.

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. Don't forget to put "darker side" in the subject line to bypass spam filters.

FROM THE BENCH

Ladder Logic (Part 1)

An Introduction to PLCs

A true programmable logic controller (PLC) is more than just a microcontroller. This article is an excellent introduction to PLC technology and its various uses. After covering the basics, Jeff explains how PLCs operate.

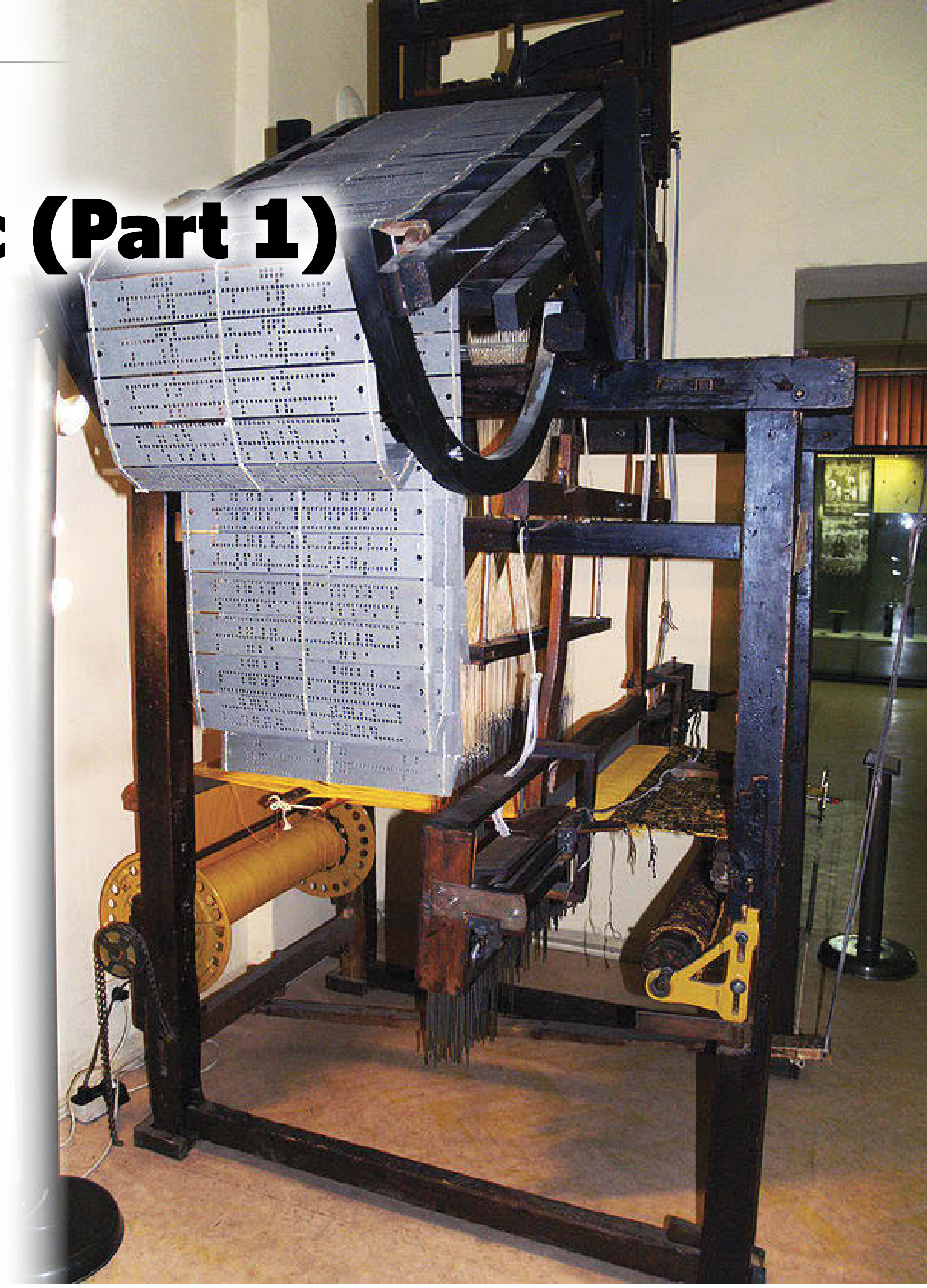


PHOTO 1
Even the semiautomated hand driven treadle looms required extensive setup and proficient operators until the invention of using punched cards to select heddles by Austrian Joseph Jacquard at the end of 19th century. ("Hand-Driven Jacquard Loom," Edal Anton Lefterov, http://en.wikipedia.org/wiki/Jacquard_loom)

By Jeff Bachiochi (US)

If we ignore the fig leaf, "skins" were most likely the first fashion statement. These predated the weaving of fibers that may reach back about 100,000 years. Before evidence of the sewing needle, which goes back tens of thousands of years, clothing was more or less wrapped or draped. However, it wasn't until just a few thousand years BC that spun fibers, made into yarn, were woven to make fabrics. This process involved a frame or loom that was used to hold many parallel fibers in place so perpendicular fibers could be interwoven between them, creating a "sheet" of fabric. While fabric could be dyed to give it a bit of color, a new process was developed using dyed fibers of different colors and adjusting the weave to select which colors were visible. This allowed for the weaving of textiles with very complex patterns. It's a very labor-intensive operation.

The production of fabric didn't advance from a cottage industry until about the 15th century. Prior to the industrial revolution passing the shuttle between the weaves was a hand operation. Once mechanized, the weaver's sole task was to adjust the heddles, the mechanism that pulls a particular fiber up. Raising a number of fibers above the rest leaves a gap in which the shuttle can be passed. Multiple heddles could be raised by a single treadle or foot pedal. Each treadle moved a different combination of heddles each forming a unique row pattern. The over all pattern is produced by following a particular sequence of treadle changes, one for each pass of the shuttle (row).

In 1801 Joseph Jacquard demonstrated a loom that automated the complete process of manufacturing textiles. Virtual treadles were created by punching a hole in a giant card indicating the heddles that should be raised. Each card corresponds to one row of the design. By chaining prepunched cards in a linear chain, a sequence of patterns could be "programmed" into the continuous loop of chained cards (see **Photo 1**). Punched cards? Programming?

COUNT ON IT

Merchants and traders had a continual need for keeping track of their daily transactions. Many used a counting frame or abacus as a calculating tool. It's easy to imagine how manual manipulation could be automated by substituting gears for beads (see **Photo 2**) and then continue by morphing into a desktop device used by companies worldwide (see **Photo 3**).

People counting was an ongoing event that reaches back thousands of years. Census data had to be estimated because it took so long to collect the data that it was outdated before it could be compiled. Once communication improved and the question of quantity could be answered in a timely fashion, it became evident that other data would be informative. Thus began the collection of massive amounts of information. Yet even with efficient collection, it often took years just to document it, not to mention analyze it.

Just over 100 years ago, Herman Hollerith developed a system for storing census data on punched cards for the 1890 US census. His "tabulator" was a glorified adding machine, which could rapidly tabulate the census statistics by reading the holes in the punched cards. (His company later merged to become IBM.) As can be inferred by **Photo 4**, the tabulator used the holes in the nonconducting card material to enable electromagnets to advance mechanical counters. Here the inputs (card holes) were directly wired to the outputs (counters) creating a specific operation. By replacing the interconnections with a control (patch) panel, the operation (intent) of the device became programmable. Thus, we have the basis of a generic device where all of its resources can be reconfigured.

Prior to vacuum tubes (and transistors), all operations were controlled via relays, or electromagnets that operate one of more switches. Most relays have at least one normally open (NO) contact and one normally closed (NC) contact in addition to the COM contact (switch wiper). The contacts reflect the state while remaining isolated from the electromagnetic coil. This allowed the contacts to be connected in combinatorial ways via the patch panels.

LADDER LOGIC

Relays grouped into logic circuits are often called line diagrams, because the inputs and

PHOTO 2
Older adding machines like this used a mechanism of gears similar to a car odometer. (Source: "Old Adding Machine," David R. Ingham, http://en.wikipedia.org/wiki/Adding_machine)

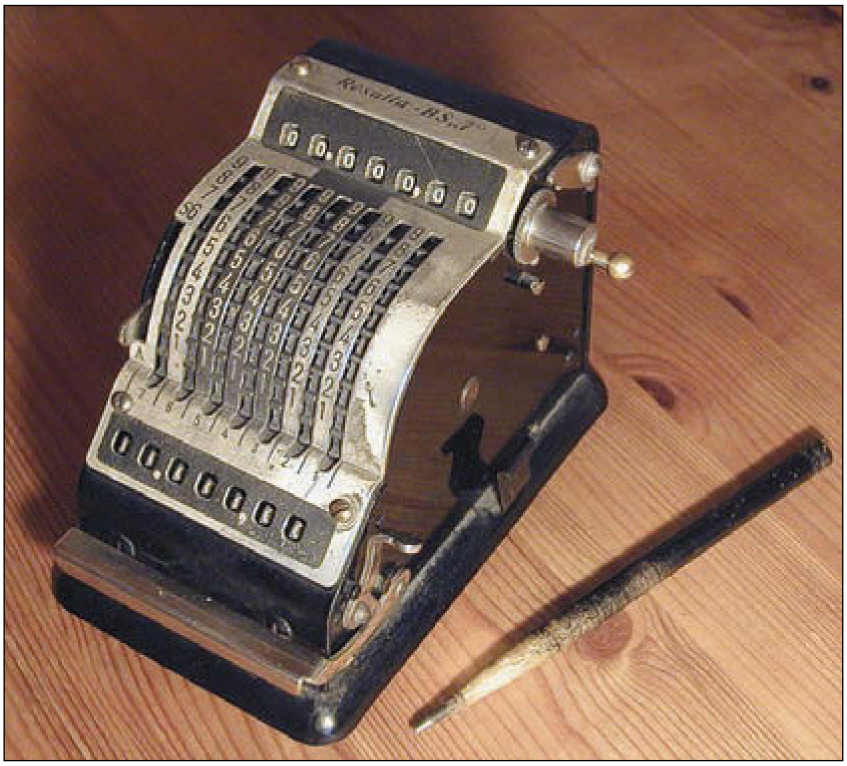


PHOTO 3
Results adding machine produced in Germany circa 1950s. (Source: "Mechanical Calculating Machine," Roger McLassus, http://en.wikipedia.org/wiki/Adding_machine)

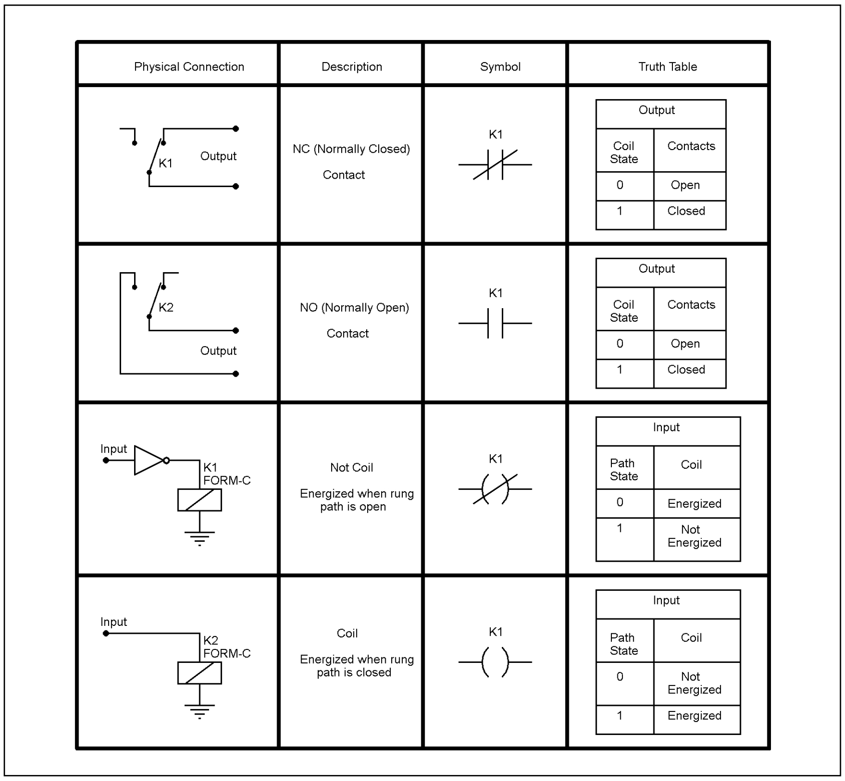


FIGURE 1
This shows the physical (contact and coil) connections for a relay and the four possible combinations of input and output states.

PHOTO 4
Herman Hollerith's tabulating machine used punched cards to hold data and tabulated the total number of holes seen in each section (category) of a card. Note the bed-of-nails approach to data input. (Source: "Hollerith Machine CHM," Adam Schuster, Flickr: Proto IBM)

outputs are essentially drawn in a series of lines. **Figure 1** the physical connections, symbol, and logic table for two types of relays used in control logic, those whose contacts are NC while not powered and those whose contacts are NO while not powered.

A relay logic circuit is an electrical network consisting of lines or rungs (contact interconnections) that must have continuity to enable an output device. A typical application may consist of a number of rungs (circuits), with each rung defining rules for its output. The output state is based on the combination of state or contact conditions of its (input) devices, such as input switches and control relays. Input states can be connected in series, parallel, or series-parallel to obtain the logic required for the output. The relay logic circuit forms an electrical schematic diagram for the control of output devices. Relay logic diagrams represent the physical interconnection of devices.

When unpowered a relay is defined as having an input state=0. The relay is energized the input state=1. An NC relay's output state=1, when the relay is unpowered. Energizing the relay opens the contacts and the output state=0. The NO relay's output state=0, while the relay is unpowered. However, when energized, the contacts close and the output state=1. **Figure 2** shows how a relay's coils and contacts are physically wired on a "rung" to form logic functions. A rung consists of input states and/or relay contact states to form the rules for a particular output's state.

PLC

Earlier I mentioned "control panels," which were patch bays used to reprogram relay connections. While the ability for an operator to redefine the rules of the game has advantages, in the industrial world, this could be a weak link in the reliability of a production process. Affixing connections with a level of permanence (hardwired) makes more sense. A design engineer would determine the rules required to achieve a set of specific tasks structured to fulfill the objective. He would then draw a line diagram consisting of rungs (sets of rules to complete an individual task) that would result in the objective being reached. An electrician would then wire the system of relays based on this schematic. It's easy to see that any error by the design engineer or electrician could require major debugging energy and delay production.

It was the auto industry that pushed for a better way. In the late 1960s, the modular digital controller (modicon) was developed to eliminate the need for all hardwired rule-based interconnections. This used a microcontroller to monitor existing inputs and drive existing outputs, based on the already established rule-based objectives. What this eliminated was all of the relays used to define a rule the rules. Output relays required to operate external equipment were still required as were all inputs (sensors, switches, etc.). However, the how input states affect the output states could now be handled within the microcontrollers. Would design engineers now be required to become microcontroller programmers?

The grand design of using a microcontroller for the job of programmable controller (PC) had already considered this potential retraining issue. Note: To eliminate confusion between the new personal computers (PC) and the programmable controller, the name was expanded to programmable logic controller (PLC). To eliminate the need for a design engineer to learn programming, an application would be written that would take the already established line diagram and automatically translate the ladder logic into the programming code necessary to replicate the rule in software.

DIFFERENT STROKES

The relay-infested hardwired control system of old is for the most part, a static system. That is, an output state is based on established input states. While a change may be considered dynamic, the state itself (once established) will affect the output directly without any other outside assistance. With the new PLC, things are a bit different. Outputs can only change when an executing program instructs it to change based on its rule-based analysis of all inputs. The output state is based on established input states, at a particular point in time. The difference here being the time required for execution of the program code.

Yes, the physical relays do in fact have some turn on/off time associated with them, but in most cases the outputs of each rung will settle into a fixed state on their own schedule (independently). The PLC must use a loop or scan, which consists of equating of each "rung" sequentially. To keep potential race conditions down, the PLC uses a scan made up of three steps: read all inputs, evaluate all rungs, and change any necessary outputs.

Up to this point, the functions discussed have been basic logic functions: AND, OR, XOR, and the like. There are other useful functions like counters and timers that up to now were extremely cumbersome to implement outside of the PLC. Like many new technologies being implemented, each manufacturer had

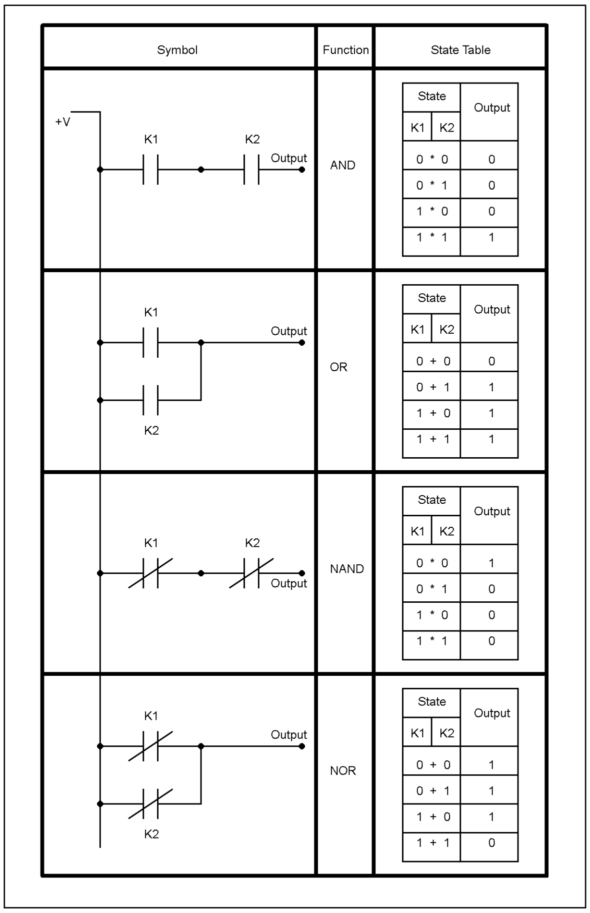


FIGURE 2
These diagrams show how a relays can be used to perform logic functions. Series connections produce AND functions while parallel connections look like OR functions.

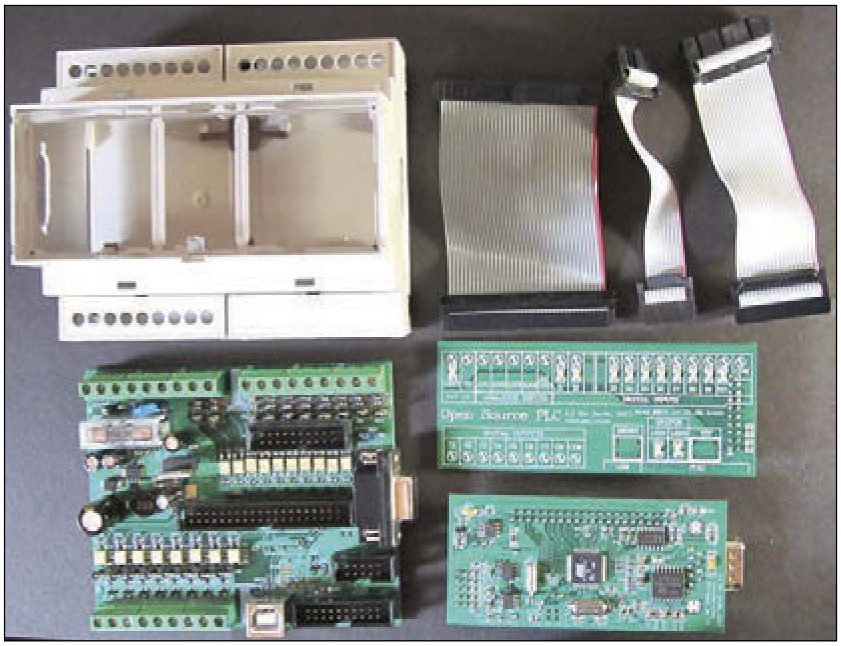


PHOTO 5
The Small Brick Open Source PLC consists of three circuit boards, a CPU board, and I/O. The enclosure is a standard DIN mount case typical to many PLCs. (Source: Starting Electronics, http:/startingelectronics.com/projects/small-open-source-PLC/PLC-components)

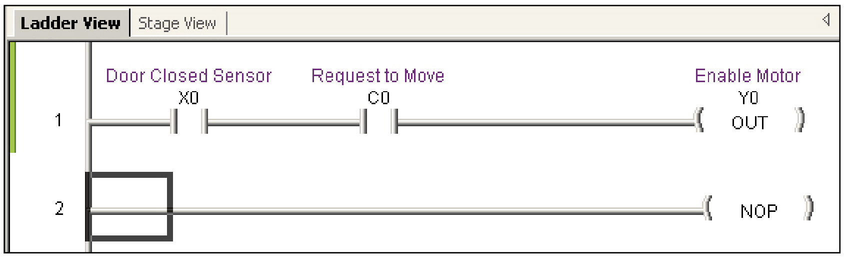


PHOTO 6
This ladder's rung has one NO (open) input and one NO virtual relay contact in series with an output relay's NO (unpowered) coil. The output can only be true (energized) if input 1 and input 2 are both true (closed). (Source: Starting Electronics)

their own way implementing each function. This generally means that one company's solution isn't necessarily compatible with other equipment. You probably have a microcontroller manufacturer that you favor. If you program it in assembly language, chances are that can't directly be used on another manufacturer's micro. However, by using a higher-level language, like C, that program can be used by any microcontroller that supports C. Likewise, programming with ladder logic for one PLC allows your application to be implemented by other PLC manufacturers. PLCOpen (www.plcopen.org) was established to organize standards for PLCs including the definition of different programming languages that can be used to program PLCs: Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD), and Structured Text (ST).

In the PLC industry, each manufacturer provides its own tools to allow a user to program an application for their particular PLC. These tools will allow you to use some/all of the various languages to write your application and compile it for a specific PLC. In addition, you will be able to save and document your application. Export and import commands make your application somewhat transportable.

MORE THAN A MICRO

A true PLC is more than just a microcontroller. **Photo 5** shows the general makeup of a PLC. It shows a three PCB system made up of a microcontroller, status, and I/O boards. I chose this photo because it clearly shows the use optoisolators and other protection on both the lower inputs and upper outputs of the I/O board. Like most PLCs this open source system is designed for a DIN rail-mounted enclosure.

Harsh industrial environments call for isolation and protection on both inputs and outputs. This keeps everyone playing nice. Your inputs and outputs can be a combination of low and high voltage AC and/or DC equipment, so many manufacturers offer I/O, usually in blocks of eight AC or DC inputs and AC, DC or relay outputs. As you might expect wiring terminals for each I/O bit require a great deal of real estate. So PLCs often require large cabinets to handle all the connections and necessary power supplies (see **Photo 6**). On larger equipment it often preferable to switch the power locally and control that relay via a lower power control from a PLC output as opposed to running high current lines inside the control cabinet.

IDE

Limited or restricted software is available for free from most manufacturers. Since the final compiled code is different for almost every make and model it is only applicable to their equipment. To write unrestricted code for, or as a third party maintain it, you may be forced to purchase a copy of the software to be able to complete the task.

Like most tools, they will not write the application for you, but merely document your logic and allow the microcontroller to use your rules to perform the actions you designate. So, even before you open the IDE you should have a pretty good idea of all the actions and rules necessary for the task at hand. That being said, let's take a quick look at the typical IDE.

One of the first steps required in a new project is to assign each of your input devices to a hardware input bit and hardware output bits to the external device it will control. Good programming practice would include assigning optional nicknames and descriptions to each. Every bit of I/O is assigned a virtual memory address. In fact the memory map is well defined so you will know where every input, output, status, and internal function registers are located. Each of these has a predefined group of addresses such as counter values or communication buffers. With the byte playing such a central role in early computing, it's no wonder octal was used as it easily defines 1 of 8. It is customary to use octal to describe groups of inputs and outputs. There are no bits 8 and 9. After the bits 0 to 7 in group 0, come bits 0 to 7 in group 1. Thus, the octal designations 00–07 and 10–17.

The PLC application is a collection of rules (conditions) that allow actions to take place. Each action and its associated rules can be visualized as a separate "rung" on the application's "ladder" of actions. The output action will be true if the rung evaluation is true otherwise it will be false. For example, take an action of 'opening' an elevator door. The elevator door has an NO sensor, X0 (input bit 0) that must be physically closed by the door. A call from any other floor (someone pushing the up or down button) will energize the C0 (Control relay bit 0) and close its NO contacts. This set of conditions can be entered by drag and dropping a NO contact to 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@imaginethatnow.com or at www.imaginethatnow.com.

Experience extraordinary training for embedded systems professionals at ESC Conference Series 2015!

The Embedded Systems Conference (ESC) is the industry's largest, most comprehensive technical conference for embedded systems professionals in the U.S. ESC is excited to announce an expanded 2015 USA Conference Series, taking place in **Boston (May 6-7), Silicon Valley (July 20-22),** and **Minneapolis (November 4-5).**

esc

EMBEDDED SYSTEMS CONFERENCE '15

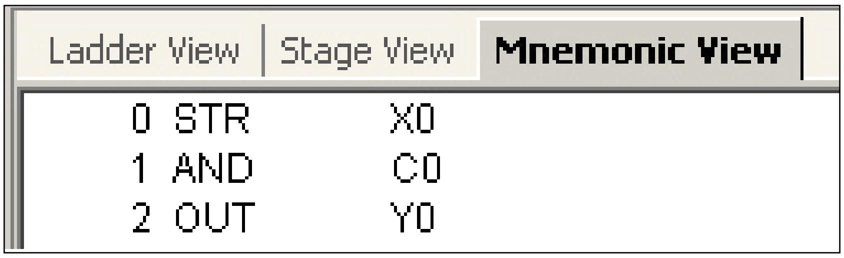
BOSTON

ESC's first stop in 2015 is Boston, MA, known for its good mix of hi-tech, biotech/life sciences, medical device, and military defense companies. It's also home to the #1 engineering university in the world - MIT.

Come join us at the Boston Convention Center and check out some of our sponsors for the event:

**Rohde & Schwarz | Green Hills Software, Inc | AdaCore| Altium, Inc.
AMP Display | Azul Systems, Inc. | Convergence Promotions | IAR Systems AB
Ironwood Electronics | MathWorks | Rigol Technologies | SmartBear Software
Symmetry Electronics | Technologic Systems | Teledyne LeCroy
Toradex | Total Phase | Vision Components GmbH | WolfSSL | Lauterbach, Inc.
Phoenix Contract Software | Security Innovation | Code
e-con Systems India Pvt. Ltd | Express Logic | XJTAG**

Register today at **www.embeddedconf.com/boston** and get 15% off a Conference Pass when using a promo code CircuitCellar15Bos, offer expires May 1, 2015.


Ladder View | Stage View | Mnemonic View

```
0 STR    X0
1 AND    C0
2 OUT    Y0
```

PHOTO 7
The IDE evaluates the ladder diagram into PLC instructions. These instructions describe the evaluation process but not necessarily the actual program code a particular PLC will use to implement the application. PLC code will be compiled specifically for that PLC from the instruction list.

beginning of a new rung and indicating the condition comes from the X0 input. Next, add a second NO contact in series (conditions in series produce an AND, conditions in parallel produce an OR) and indicate this condition is C0, a virtual state. We want these two conditions to produce the action of moving the elevator car. This action is indicated by replacing the NOP action at the end of the rung with a NO coil icon (output) labeled as Y0. **Photo 6** shows what this looks like. If the contacts of X0 are false, the door is open, the elevator can never move. If no one has called for the elevator, the virtual contacts of C0 remain open and the elevator car can't move. Only when door is closed (contact closes) and a request has been made (contacts close) can the output coil Y0 be true (energized). It is assumed that the beginning of the rung is power. If this power can find a complete path through the rung, then the output will be energized! Each rung produces a list of instructions by the IDE. Refer to **Photo 7** for the three instructions that represent this rung. A complete application becomes a single list of each rung's instructions.

PLC instructions are grouped into three types of tasks, two of which may be obvious by the previous example: Contact and Coil operations. Each has a number of associated instructions. Most contact instructions have to do with retrieving input or contact information and evaluating it, while coil instructions evaluate and adjust data or coil states. The third type is the function. These include timers, counters, and other higher level tasks. Since you are most likely familiar with the abilities of microcontrollers, you can imagine other functions that weren't available via relay logic, like the monitoring of analog signals and variable control available through analog outputs. This growing list of additional functions has greatly increased the usefulness of the PLC and thus prolonged its life.

THE IMPORT-EXPORT BUSINESS

Most useful for documentation purposes the EXPORT command provides a number of ways to put your application to paper. The two most used are the ladder diagramming and instruction list. If the instructions adhere to the standard, they should be able to be imported to any manufacturer's IDE that provides an IMPORT function. If you need to replace your PLC with another model (or even manufacturer), IMPORT allows you to bring your proven code to the new hardware with minimal hassle.

The plan is to explore this export file to obtain key information that describes the application. Next replicate the application using some independent circuitry that we can experiment with on the bench. While the project may be based on a specific microcontroller, the discussion should be general enough that it can be implemented for use with almost any microcontroller, using the language of your choice.

While this introduction was more than I originally intended to present, it barely uncovers the power of the PLC. An interesting aspect that I have left entirely untouched is the PLCs ability to communicate with other devices through a comm channel. This allows one system to work in concert with other systems perhaps as part of an assembly line. Many PLC systems also include a user interface that allows dynamic control of the process. This permits skilled labor to interact with the process. An interface might also give visual feedback on the status of the system's process.

Process control is the foundation of industry. It is necessary to enable the continuous production of goods. While automation does reduce the manufacturing labor force required, it requires an increase in labor in other areas of expertise. In addition to operating personnel who closely monitor and control the complex processes from a central location, program engineers are required to design and make application changes, electrical engineers are needed to maintain interconnections, mechanical engineers are needed to maintain and replace equipment, and system architects must coordinate the integration of all systems in the process facility. I for one am willing to spend more for any product that says Made in the USA. It's a mighty symbol, and should be the impetus for companies to invest in on-shore facilities.

circuitcellar.com/ccmaterials

SOURCE
Small Brick Open-Source PLC
Starting Electronics | www.startingelectronics.com

FROM THE ARCHIVES

Quad Bench Power Supply

The need for a bevy of equipment for building and testing presents a problem: how to deliver an adequate power supply while keeping workbench clutter to a minimum. Brian decided to tackle this classic engineering conundrum with a small, low-capacity quad bench power supply.

By Brian Millier (US)

I hate to admit it, but my electronics bench is not a pretty sight, at least in the midst of a project anyway. Of course, I'm always in the middle of some project that, more often than not, contains two or three different projects in various stages of completion. To make matters worse, most of my projects involve microchips, which have to be programmed. Because I use ISP flash memory MCUs exclusively, it makes sense to locate a computer on my construction bench to facilitate programming and testing. To save space, I initially used my laptop's parallel port for MCU programming. It was only a matter of time before I popped the laptop's printer port by connecting it to a prototype circuit with errors on it.

Fixing my laptop's printer port would have involved replacing its main board, which is an expensive proposition. Therefore, I switched over to a desktop computer (with a $20 ISA printer port board) for programming and testing purposes. The desktop, however, took up much more room on my bench.

You can't do without lots of testing equipment, all of which takes up more bench space. Amongst my test equipment, I have several bench power supplies, which are unfortunately large because I built them with surplus power supply assemblies taken from older, unused equipment. This seemed like a good candidate for miniaturization.

At about the same time, I read a fine article by Robert Lacoste describing a high-power tracking lab power supply ("A Tracking Lab Power Supply," Circuit Cellar 139). Although I liked many of Robert's clever design ideas, most of my recent projects seemed to need only modest amounts of power. Therefore, I decided to design my own low-capacity bench supply that would be compact enough to fit in a small case. In this article, I'll describe that power supply.

MY WISH LIST

Even though I mentioned that my recent project's power demands were fairly modest, I frequently needed three or more discrete voltage levels. This meant lugging out a couple of different bench supplies and wiring all of them to the circuit I was building. If the circuit required all of the power supplies to cycle on and off simultaneously, the above arrangement was extremely inconvenient. In any event, it took up too much space on my bench.

I decided that I wanted to have four discrete voltage sources available. One power supply would be ground referenced. Two additional power supplies would be floating power supplies. Each of these would have the provision to switch either the positive or negative terminal to the negative (ground) terminal of the ground-referenced supply, allowing for positive or negative output voltage. Alternately, these supplies could be left floating with respect to ground by leaving the aforementioned switch in the center position.

This arrangement allows for one positive and two positive, negative or floating voltage outputs. To round off the complement, I added Condor's commercial 5-V, 3-A linear power supply module, which I had on hand in my junk box. **Table 1** shows the capabilities of the four power supplies.

I wanted to provide the metering of voltage and current for the three variable power supplies. The simultaneous voltage and current

Supply number	Voltage range	Currnet capacity	Notes
1	2.5–8 V	500 mA	Ground-referenced
2	2.5–15 V	300 mA	Bipolar or floating
3	2.5–15 V	300 mA	Bipolar or floating
2	5 V	3 A	Fixed logic supply, commercial mode

TABLE 1
As you can see, there are four power supplies. I've included all of the information you need to understand their capabilities.

measurement of three completely independent power supplies seemed to indicate the need for six digital panel meters. Indeed, this is the path that Robert Lacoste used in his tracking lab supply.

I had used many of these DPM modules before, so I was aware of the fact that the modules require their negative measurement terminal to float with respect to the DPM's own power supply. I solved this problem in the past by providing the DPM module with its own independent power source. Robert solved it by designing a differential drive circuit for the DPM. Either solution, when multiplied by six, is not trivial. Add to this the fact that high-quality DPMs cost about $40 in Canada, and you'll see why I started to consider a different solution.

I decided to incorporate an MCU into the design to replace the six DPMs as well as six 10-turn potentiometers, which are also becoming expensive. In place of $240 worth of DPMs, I used three inexpensive dual 12-bit ADCs, an MCU, and an inexpensive LCD panel. The $100 worth of 10-turn potentiometers was replaced with three dual digital potentiometers and two inexpensive rotary encoders.

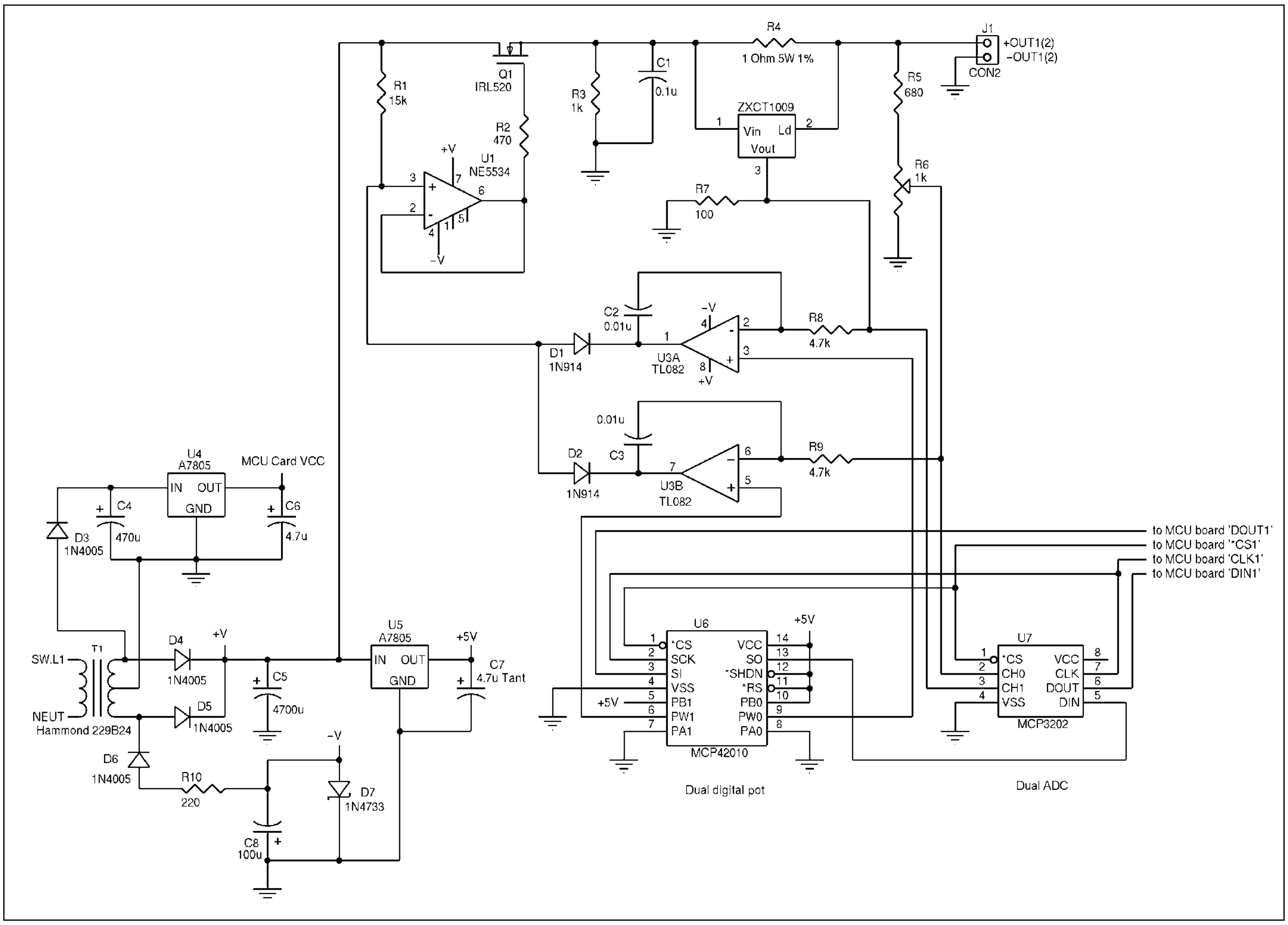
Using a microcontroller-based circuit basically allows you to control the bench supply with a computer for free. I have to admit that I decided to add the commercial 5-V supply module at the last minute; therefore, I didn't allow for the voltage or current monitoring of this particular supply.

THE ANALOG CORE

Although there certainly is a digital component to this project, the basic power supply core is a standard analog series-pass regulator design. I borrowed a bit of this design from Robert's lab supply circuit.

Basically, all three power supplies share the same design. The ground-referenced power supply provides less voltage and more current than the floating supplies. Thus, it uses a different transformer than the two floating supplies. The ground-referenced supply's digital circuitry (for control of the digital

FIGURE 1
The ground-referenced power supply includes an independent 5-V supply to run the microcontroller module.



potentiometer and ADC) can be connected directly to the MCU port lines. The two floating supplies, in addition to the different power transformer, also need isolation circuitry to connect to the MCU.

Figure 1 is the schematic for the ground-referenced supply. As you can see, a 24VCT PCB-mounted transformer provides all four necessary voltage sources. A full wave rectifier comprised of D4, D5, and C5 provides the 16 V that's regulated down to the actual power supply output. Diodes D6, R10, C8, and Zener diode D7 provide the negative power supply needed by the op-amps.

A UA7805 regulator is used to drop the 16-V supply down to the 5 V needed for the digital potentiometer and ADC. Finally, an independent 5-V power supply for the MCU is provided by D3, C4, and U4, another UA7805 three-terminal regulator. Because I eventually added a 5-V, 3-A commercial power supply to the unit, I think it would have made more sense to run the MCU from that supply instead.

The series-pass element is an IRL520 power MOSFET that's driven by U1, which is configured as a unity-gain buffer. I had the IRL520 devices on hand, but I suspect that NPN Darlington transistors could have been used in their place with the advantage of a lower base drive voltage requirement.

Voltage regulation is performed by comparing a portion of the power supply output voltage with the B-section output of the digital potentiometer U6. A TL082, U3-B acts as a comparator for this purpose. The full-scale output of the digital potentiometer is 5 V, and the power supply output voltage is scaled down to this level by R5 and the potentiometer R10. Without any initialization from the MCU, the digital potentiometer presets itself to half scale, or 2.5 V at power-up. When testing this power supply, prior to connecting it to the MCU, potentiometer R10 is adjusted to provide an output voltage of 6.4 V at power-up. This gives a resolution of 50 mV per step of the digital potentiometer.

Current limiting is provided by comparator U3A and the A section of the digital potentiometer. Current monitor IC U2, which you'll learn more about later, provides a voltage that's proportional to the output current. Basically, comparator U3A compares a voltage proportional to current draw, with the current limit set point value programmed into the digital potentiometer, and throttles back the drive to the pass regulator when necessary.

The two sections of the TL082, acting as comparators, have their outputs connected to buffer U1's input via diodes D1 and D2. In combination with R1, these components provide a NOR function. To be precise, if either comparator's output goes low, the drive to the pass regulator (provided by R1) will be reduced until the over-voltage/current condition ceases.

Apart from the digital potentiometers replacing mechanical ones, this circuit is somewhat similar to that used by Robert in his lab power supply. You'll learn more about how I used the high-side current monitor circuit a little later.

Although Robert didn't mention any instability problems in his article, I experienced them myself as I was building this circuit. I found it necessary to use 0.01 capacitors (C2, C3) for feedback compensation on both comparators in order to eliminate RF oscillation on the power supply output under varying load conditions. I thought I could eliminate buffer U1 in my design because of the low current requirements of the MOSFET pass element; however, the diode NOR circuit seemed to produce RF oscillations on the power supply's output without this buffer in place.

The final part of the circuit is the metering portion. In place of the DPMs, I used a Microchip MCP3202, which is a dual 12-bit ADC. This ADC is inexpensive (it costs less than $3) and doesn't need an external reference. The fact that it uses an SPI interface really simplifies the isolation circuitry needed in the floating supplies.

Even though the MCP3202 can operate from 2.7 to 5.5 V, I chose to operate it from 5 V, because that regulated voltage was easy to provide with a UA7805. The disadvantage to this power supply voltage was that the ADC's full-scale input is also 5 V. Though the power supply's output voltage is scaled down to this range for the regulation circuitry, the current-monitoring circuitry converts current to a somewhat lower voltage. Despite the fact that the actual scaling differs between the floating and non-floating power supplies, the net result is that current resolution is only about 9 bits. This current resolution was sufficient for my purposes, however.

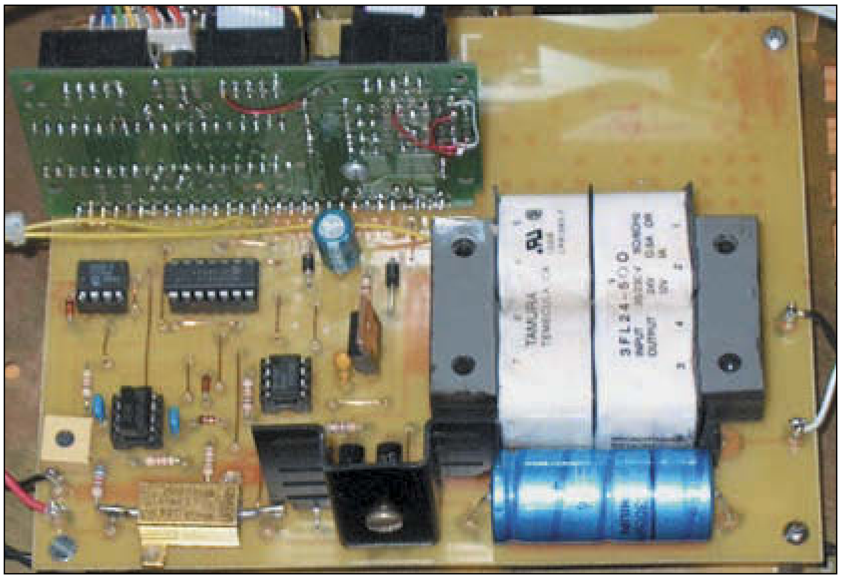


PHOTO 1
The ground-referenced power supply PCB also contains the SIMM100 MCU daughterboard. The IsoLoop isolators, being SMD components, are mounted on the bottom of the PCB and aren't in view.

The MCP42010 dual digital potentiometer has a neat feature: it contains a Serial Out terminal. Using this feature, you can daisy-chain these devices and load many of them simultaneously, using only three control lines—CS, SCK, and SI (with the daisy-chained devices being fed from the previous device's SO line).

Although I needed only one dual potentiometer for each power supply, I used this feature to daisy-chain the MC3202 ADC device to the digital potentiometer, thereby eliminating one—the CS control line—for the nonessential ground-referenced supply. For the floating supplies, it allowed me to provide all of the necessary isolation in one device package, which was beneficial.

To protect against short circuits, I added a Raychem PolySwitch RXE075 resettable fuse, which limits short-circuit current to 750 mA. I did this because the Zetex high-side current monitors need at least 2.5 V to operate properly. A direct short circuit would not provide this, and the current-limiting action would not work. The PolySwitch fuses more than function: they act as fuses and provide enough voltage drop during short-circuit conditions to allow the Zetex current monitors to operate.

Although it isn't obvious from the schematic, I designed this power supply's PCB to include a 30-pin SIMM connector. The MCU module is a daughterboard on this PCB. Also, the two isolation chips that interface the MCU to the two floating power supplies are contained on this PCB. **Photo 1** depicts the PCB and the backside of the MCU module. I'll describe both the MCU module and the isolation circuits later.

SEE IF IT FLOATS

I've explained in detail the ground-referenced power supply. There are only a few differences between it and the two floating power supplies; however, I've provided **Figure 2** to show you these differences.

Where the ground-referenced supply was meant to provide 8 V at about 500 mA, the floating supplies were meant to provide higher voltages for powering analog circuits such as op-amps. I wanted at least a 15-V output, but a current capacity of 300 mA was deemed sufficient for my needs. I substituted a 34-V transformer for T1. It's the same size as the 24-V device used in the ground-referenced supply, which was handy because all three power supplies share a similar PCB layout.

The floating supplies need not include the 5-V regulated MCU power supply that was part of the ground-referenced supply. The value of the output voltage-scaling network is different

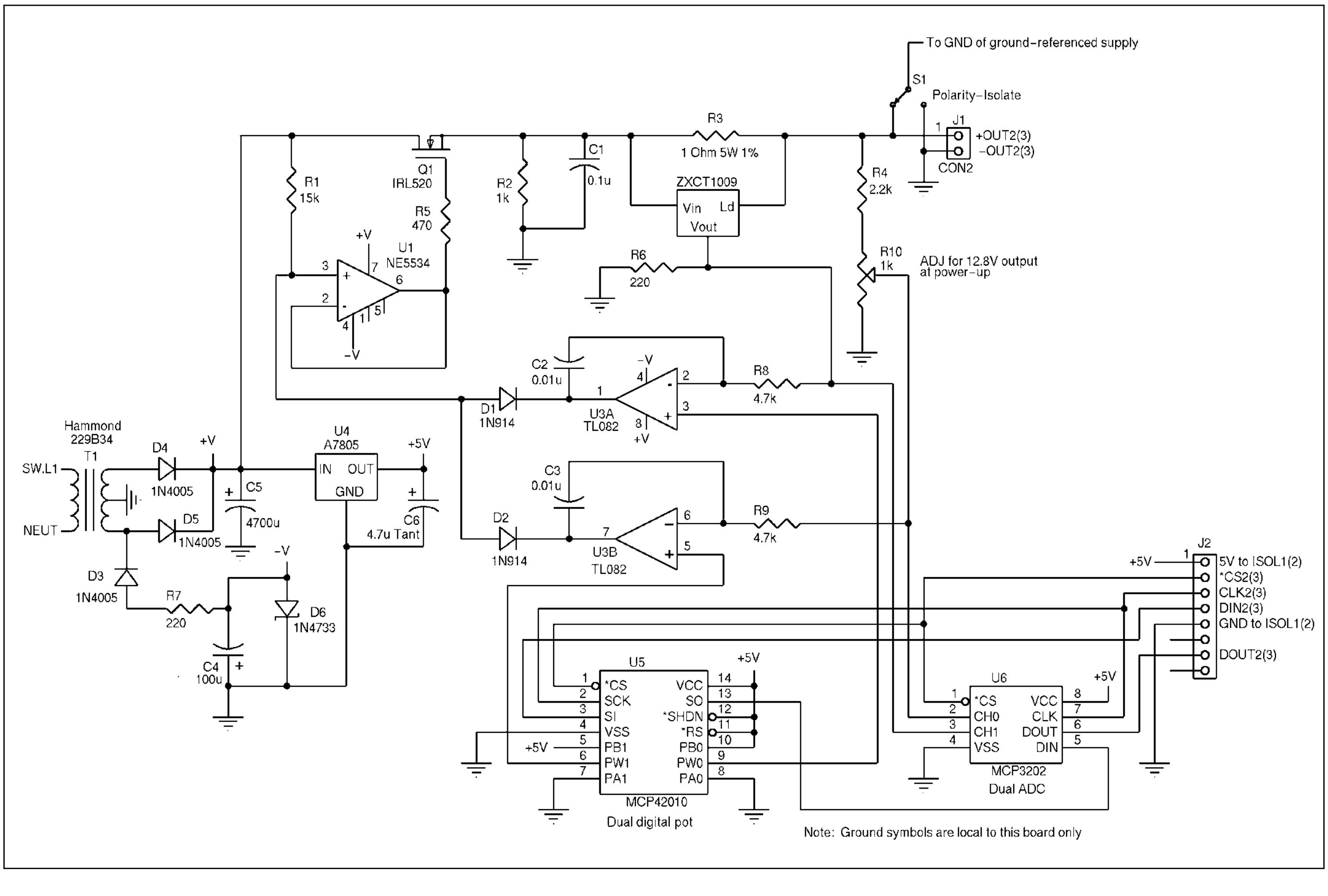


FIGURE 2
The floating supplies are almost identical to Figure 1, but there are different component values. Note that the ground symbols in this figure are local to this board alone (i.e., they are not connected to ground on any other boards shown in the other figures).

from the ground-referenced supply. In this case, potentiometer R10 is set to produce 12.8 V at power-up. This gives a resolution of 100 mV per digital potentiometer step.

The only remaining difference has to do with the value of the current monitor-scaling resistor R6. I increased the value of this resistor from 100 to 220 W to scale the lower current capacity of this supply into a voltage that's compatible with both the 5-V referenced ADC and digital potentiometer.

THE ZETEX ZXCT1009

You can monitor the current drawn from the power supplies in two ways. Both methods involve inserting an accurate low-value resistor in series with the power supply output, and then measuring the voltage drop across that resistor. A measure of the current drawn then will be equal to the voltage drop/resistor value. If that resistor is placed in series with the negative output terminal of the power supply, the resulting voltage drop will be referenced to the power supply's common terminal. This makes it easy to measure with an ADC (or DPM) that is powered by, and referenced to, the power supply's common terminal.

The downside of this method is that whatever voltage is dropped across, this current sense resistor is lost (i.e., the load gets a little less voltage than the power supply thinks it is providing, and you see an inflated reading on the voltage meter).

Alternately, you can place the current-monitoring resistor in series with the positive output terminal of the power supply. Then, the voltage feedback network of the pass regulator can be wired to follow this resistor, eliminating the lost voltage problem that I described earlier.

This method, however, introduces the main problem associated with the measuring of a small current-sense voltage riding on a large common-mode voltage: the power supply voltage itself. You can minimize this problem by using a high-quality instrumentation amplifier and precision-matched resistors, but they are somewhat costly. This second approach is called high-side monitoring.

In his lab supply project, Robert devised a clever circuit to compensate for the lost voltage problem that plagued the first method I described. In my design, I chose to go with the second approach—high-side monitoring.

I came to this decision after discovering a clever IC made by Zetex called a high-side current monitor. The ZXCT1009 is a three-pin device in an SOT23 package that converts the voltage dropped across a high-side current sense resistor into a current. This current is sent through a resistor to the power supply's common terminal, providing an easy-to-measure voltage proportional to the current draw.

The problems of measuring the low sense voltage riding on the high power supply common-mode voltage are addressed inside the ZXCT1009; therefore, you don't have to worry too much about this. Because the device costs roughly $1, it certainly beats designing in an instrumentation amplifier to perform this task.

However, the ZXCT1009 isn't a universal solution to the current-sensing problem. It requires an input voltage of 2.5 V or greater, so you can't easily monitor current if you want to run your power supply at voltages less than this. The maximum input voltage it can withstand is 20 V without additional circuitry. Neither limitation was a deal breaker for me, so I incorporated one of these devices in each power supply. My biggest concern was holding the tiny device steady while I soldered it to the PCB!

You may want to consult the Zetex datasheet for more information, but the only other detail I'll mention is that the device produces 10 mA for every 1 V dropped across the current sense resistor. I had 1-W, 1% 5-W resistors in my junk box, so that's what I used for the current-sense resistors in all three supplies. This didn't waste too much of the power supply's voltage capability.

The lower-current floating supplies used a 220-W resistor to convert ZXCT1009's output current into a voltage. The higher-current, ground-referenced supply has a fitted 100-W resistor, and the MCU's software performs the math that's necessary to convert the ADC's output into the correct current reading on the meter.

AN IDEAL ISOLATOR

After spending years servicing and designing electronics devices, I have to say that I'm as impressed with some of the amazing things that were done with vacuum tube circuits back in the old days, as I am with some of the modern, miniature ICs that are available today.

For this project, though, I pampered myself with state-of-the-art devices rather than depending on clever, but more involved, circuits using conventional devices. I've already described the Zetex current monitor, which is one example of this. I continued with this trend in choosing the isolation technique for the floating power supplies.

The digital control and monitoring signals for the two floating supplies have to be electrically isolated from the ground-referenced MCU circuit. Thanks to the clever design of Microchip's SPI digital potentiometer and SPI ADC, each power supply needed only four control signals: three outputs from the MCU and one input.

ABOUT THE AUTHOR

Brian Millier runs Computer Interface Consultants. He was an instrumentation engineer in the Department of Chemistry at Dalhousie University in Halifax, Canada, for 29 years.

My first inclination was to use optoisolator chips. I had just finished another project using optoisolators to interface the same Microchip SPI ADCs. In that project, meeting the ADC's SPI timing considerations given the rather slow response of the optoisolators was a bit tricky, although possible.

Luckily, Jeff Bachiochi had just written a column about isolation in which he outlined a novel line of isolators made by Nonvolatile Electronics ("You're Not Alone—Dealing with Isolation," Circuit Cellar 142). Rather than using an optical method to achieve galvanic isolation, these isolators use magnetism. Although pulse transformers have been around for ages and can perform isolation using magnetism, they are comparatively bulky, expensive, and don't pass DC levels.

The IsoLoop isolators, on the other hand, use GMR or giant magnetoresistive devices to sense the magnetic field change produced by an excitation coil, which is nearby but electrically isolated. The change in resistance of the magnetic thin film layer is used, along with other on-chip circuitry, to implement the isolation function of the device. The IsoLoop devices actually differentiate the input signal, and send only short magnetic pulses through their excitation coils during input signal transitions. The resulting resistance changes in the magnetic thin film layer—configured in a Wheatstone bridge—are measured, and the resulting output signal is actually the output of an on-chip latch device.

Don't be fooled by the use of the term "giant" in GMR; these devices are tiny. Typically, four isolators will fit into a 16-pin wide SOIC package. The wide package is needed, presumably, to allow the devices to withstand the 2500 VRMS at which they are rated.

With regard to the packaging, I was impressed with NVE's decision to produce several different device configurations. They sell the normal quad devices with all four channels configured in the same direction (IL715); however, they also sell quad devices containing two channels in each direction (IL716). My favorite, the IL717, has three channels in one direction and the remainder going in the other direction. This configuration is perfect for SPI device isolation, which needs a Chip Select, Clock, and Data Out lines coming out from the MCU and a Data In line going back into the MCU.

Given the modest voltage isolation I needed for this supply, I could have used a quad optical isolator and wired up one section "backwards," so to speak, but the PCB layout would have been much less neat. In cases where input and output signals have to be isolated and substantial voltage isolation is required, the only way to achieve this—apart from using separate devices—is to use an appropriately configured device like those in this IsoLoop family.

I've actually saved the best part for last: these IsoLoop devices are fast! The IL700 family exhibits a 100-Mbps data rate. In addition, it has only 2-ns pulse width distortion and 10-ns pulse delay.

Unlike optoisolators, which require LED drive voltage/current and often don't provide logic-level output signals, the IsoLoop devices work directly with 3.3- or 5-V logic devices including MCUs. Although an optical isolator requires a steady drive current whenever its LED is turned on, the IsoLoop devices use only a short pulse of magnetism whenever the input signal changes state (even though a small but steady current is required for the detection and latching circuitry in the chip).

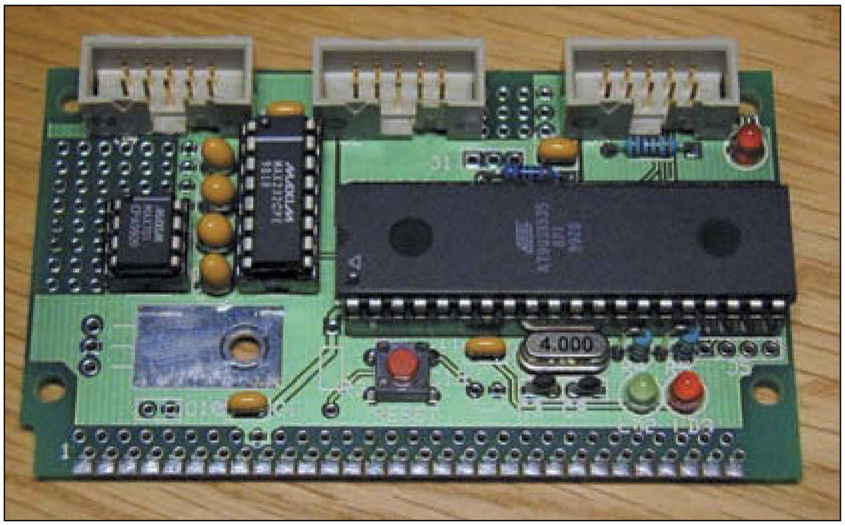
The IL717 that I used requires only a 2.5-mA power supply current on its input side, and 6 mA on its output side. This difference arises from the fact that the device has three channels in one direction and only one in the other.

In my design, I did not have to give any more thought to the SPI timing on the floating channels than I did to the channel that wasn't isolated. Basically, what goes into the IL717 is what comes out the other side!

There are only two cautionary notes that I would add regarding these devices. First, IsoLoop devices transmit their signal across the isolation barrier only on signal transitions. The recovered signal on the other side of the barrier is then electrically latched. Practically, this means that the output of the devices is indeterminate until input transitions occur. For some applications, this means that an initialization routine must be performed to ensure that the device's outputs are in a known state after power-up.

The second cautionary note is just as important. Because the devices rely on sending a short magnetic pulse at each input transition, it is important to place at least a 47-nF ceramic

PHOTO 2
I used a Lawicel SIMM100 module for the microcontroller and associated circuitry.



decoupling capacitor between VDD and ground on both input and output ports of the device. The capacitors should be placed close to the actual device pins.

I tried to share one capacitor between two IsoLoop devices on the common MCU port side of the two devices. This didn't work. There were random output errors on the device farthest away from the sole capacitor that disappeared completely when I followed directions!

MCU AND USER INTERFACE

As with every other project I've worked on in the last two years, I chose the Atmel AVR family for the MCU. In this case, I went with the AT90S8535 for a couple of reasons. I needed 23 I/O lines to handle the three SPI channels, LCD, rotary encoders, and RS-232. This ruled out the use of smaller AVR devices. I could've used the slightly less expensive AT90LS8515, but I wanted to allow for the possibility of adding a temperature-sensing meter/alarm option to the circuit. The '8535 has a 10-bit ADC function that's suitable for this purpose; the '8515 does not.

The '8535 MCU has 8 KB of ISP flash memory, which is just about right for the necessary firmware. It also contains 512 bytes of EEPROM. I used a small amount of the EEPROM to store default values for the three programmable power supplies. That is to say, the power supply will power up with the same settings that existed at the time its Save Configuration push button was last pressed.

To simplify construction, I decided to use a SIMM100 SimmStick module made by Lawicel. The SIMM100 is a 3.5² × 2.0² PCB containing the '8535, power supply regulator, reset function, RS-232 interface, ADC, ISP programming headers, and a 30-pin SimmStick-style bus. I've used this module for prototypes several times in the past, but this is the first time I've actually incorporated one into a finished project. **Photo 2** is the manufacturer's picture of an assembled module. For this project, I populated a bare SIMM100 PCB with only the components that I actually needed.

The MCU port signals needed to operate the three SPI channels and interface the two rotary encoders come out through the 30-pin bus. As you now know, I designed the ground-referenced power supply PCB to include space to mount the SIMM100 module, as well as the IsoLoop isolators. The SIMM100 mounts at right angles to this PCB; it's hard-wired in place using 90° header pins. The floating power supplies share a virtually identical PCB layout apart from being smaller because of the lack of traces and circuitry associated with the SIMM100 bus and IsoLoop isolators.

The SIMM100 module has headers for the ISP programming cable and RS-232 port. I used its ADC header to run the LCD by reassigning six of the ADC port pins to general I/O pins.

When I buy in bulk, it's inevitable that by the time I use the last item in my stock, something better has taken its place. After contacting Lawicel to request a .jpg image of the SIMM100 for this article, I was introduced to the new line of AVR modules that the company is developing.

Rather than a SimmStick-based module, the new modules are 24- and 40-pin DIP modules that are meant to replace Basic Stamps. Instead of using PIC chips/serial EEPROM and a Basic Interpreter, they implement the most powerful members of Atmel's AVR family—the Mega chips.

Mega chips execute compiled code from fast internal flash memory and contain much more RAM and EEPROM than Stamps. Even though flash programming AVR-family chips is easy through SPI, using inexpensive printer port programming cables, these modules go one step further by incorporating RS-232 flash memory programming. This makes field updates a snap. Take a look at the new stAVeR40 module in **Photo 3**. I might have used this module instead of the SIMM100 had it existed when I started the project.

The user interface I settled on consisted of a common 4 × 20 LCD panel along with two rotary encoders. One encoder is used to scroll through the various power supply parameters, and the other adjusts the selected parameter. The cost of LCDs and rotary encoders is reasonable these days. Being able to eliminate the substantial cost of six DPMs and six 10-turn potentiometers was the main reason for choosing an MCU-based design in the first place. **Photo 4** shows the front panel of the unit.

Inexpensive rotary encoders come in two basic flavors: quadrature encoded and 2-bit binary (Gray) coded. I've used the quadrature-encoded style in the past, but the ones I used

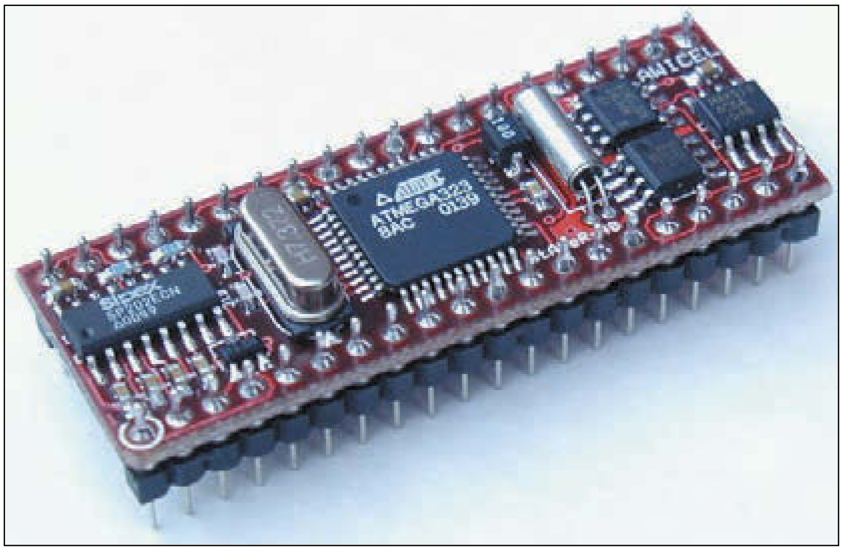


PHOTO 3
Lawicel's new stAVeR40 module is a decent product. I might have used it in place of the SimmStick had it been available when I was designing my project.

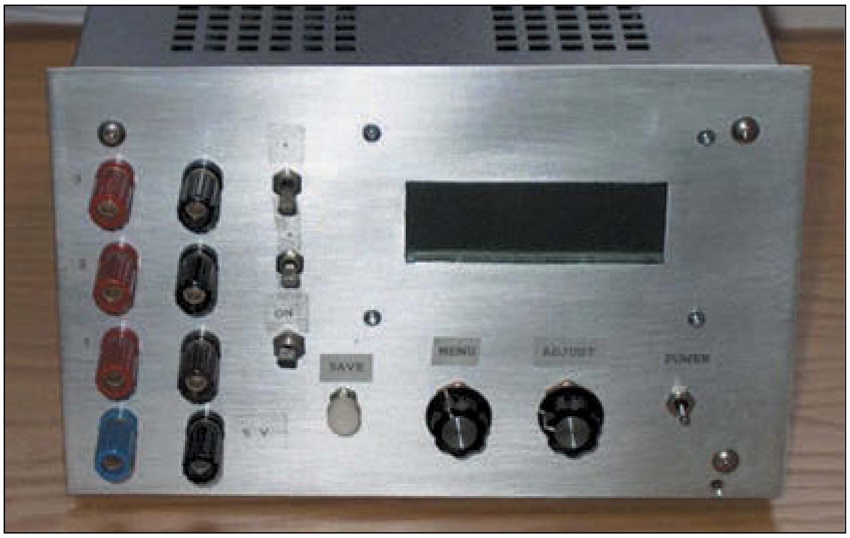


PHOTO 4
To the right of the output Johnson posts are the switches that set the polarity of the floating supplies—as well as the switch that disconnects all power supply outputs—while leaving the unit still powered up.

for this project have a 2-bit output (with Gray coding). With only 2 bits, the encoder can represent only four different values, even though it has 32 detents per rotation. With this in mind, it's necessary for the firmware to constantly poll both encoders and keep track of the carry or borrow conditions that occur as the encoder moves beyond a four-position range. The main control loop in the firmware is executed every few milliseconds, so keeping an accurate track of the rotary encoder's position is accomplished readily.

The RS-232 port came as part of the SIMM100 module. Thinking about the future, I envision adding some firmware code to allow the bench supply to be remotely controlled by a host PC, and to allow for the data logging of the various voltages/currents over time.

I haven't provided you with a complete block diagram, but I did incorporate a few features that don't show up on the individual schematics. Previously, I mentioned adding an additional commercial 5-V, 3-A supply for logic circuits. I also added a 3PST switch, with one section in series with each supply's positive output, to allow all power supplies to be disconnected from the load during power-up. A small DC computer-type fan was mounted on the top of the outer case for cooling purposes, because the pass-transistor heatsinks that I used were not too large.

Lastly, **Figure 3** shows you how the '8535 MCU would typically be connected to the rest of the circuit. It doesn't show the exact wiring of the SIMM100 including the bus connections, because this detail isn't needed when constructing the circuit from scratch (i.e., if you're not using the SIMM100 module). The SIMM100 documentation will give you all of the necessary information regarding the header and bus connections on the module.

FIRMWARE

If you've read any of my more recent articles, then I'm going to sound like a broken record in this section. I used an MCS Electronics BASCOM-AVR compiler for this project (once again). The code did not have to run extremely fast, but floating-point and string operations were needed. Because there was plenty of flash memory available in the '8535, it made sense to program in Basic rather than using Assembly language.

Skipping over the unit's initialization procedure for now, the control loop in the program works basically as follows. Both encoders are checked to see if the user has moved them. If the Menu encoder is changed, nothing is done, apart from moving an arrow cursor amongst the various parameters that can be changed. If the Adjust encoder is moved, the appropriate routine is called to adjust the necessary power supply's voltage or current limit setting. This is accomplished by changing the value of the appropriate section of the digital potentiometer located on the proper supply PCB.

Because each supply's ADC is digitally cascaded with that supply's digital potentiometer, the routine that updates the digital potentiometers also reads the ADC all in one operation. For that reason, in the absence of any changes to the voltage or current-limit settings, each power supply is sent a control message at 0.5-s intervals to set its digital potentiometers and read the dual ADC. Constantly resetting the digital potentiometers at this interval is unnecessary, but periodically reading the ADCs is necessary to give you timely voltage/current readings.

PROJECT FILES

circuitcellar.com/ccmaterials

SOURCES

AT90S8535 Microcontroller
Atmel Corp.
www.atmel.com

Power supply module
Condor D.C. Power Supplies, Inc.
www.condorpower.com

SIMM100, stAVRer modules
Lawicel HB
www.lawicel.com

BASCOM-AVR Compiler/programmer
MCS Electronics (Holland)
www.mcselec.com

MCP42010 Digital potentiometer, MCP3202 ADC
Microchip Technology, Inc.
www.microchip.com

IsoLoop high-speed digital isolators
Nonvolatile Electronics, Inc.
www.isoloop.com

RXE075
Raychem Corp.
www.raychem.com

ZXCT1009 Current monitor
Zetex Semiconductors
www.zetex.com

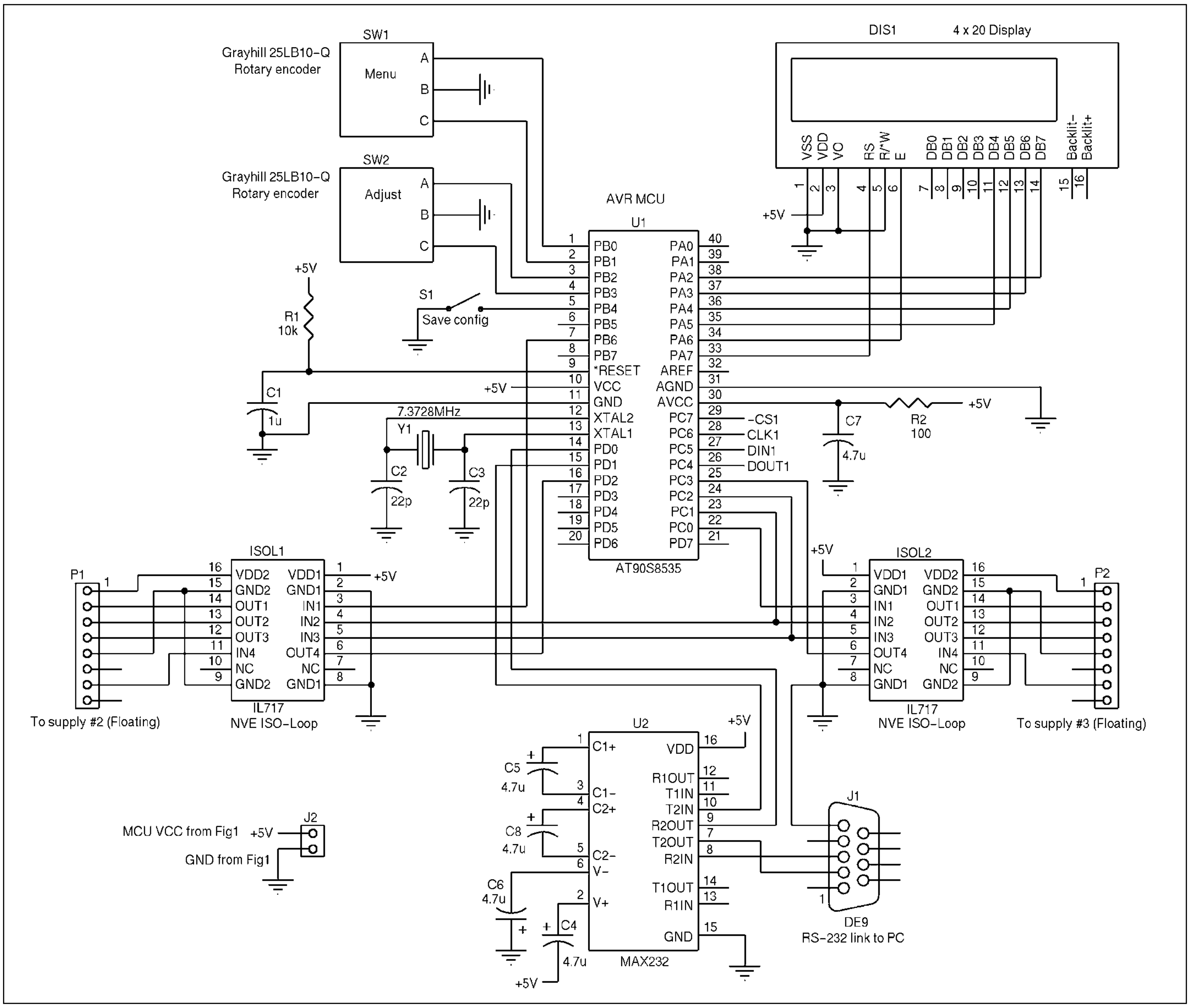


FIGURE 3
Take a look at the MCU, IsoLoop isolators, and the user interface. Some of this circuitry is actually contained on the SIMM100 module.

The only remaining task in the control loop is to check the state of the Save Configuration push button. When it's pressed, a routine is called to save the current values of voltage and current limit, for all three power supplies, to data EEPROM.

At power-up, the data EEPROM is checked for a valid configuration saved from a previous use of the supply. If so, these voltage/current settings are stored in RAM variables, and the three supplies are initialized to these settings. In the absence of valid configuration readings, each power supply is set to half scale, and the current limit settings are preset to maximum.

WRAP UP

I'm looking forward to the convenience of using this multi-output yet compact power supply in my future projects. As with all projects, there were some compromises I made along the way.

I chose Microchip's dual 8-bit digital potentiometers for the voltage/ current settings. Basically, I felt the 50-mV voltage-setting resolution (100 mV for floating supplies) was sufficient for my purposes. The resulting current-limit resolution of 20 mA (8 mA for floating supplies) also seemed reasonable; however, dual 12-bit SPI DACs are available, which would improve this resolution substantially. Maxim makes some nice serial DACs, but they come in such small packages that I can't handle or solder them to a PCB.

The existing version of the firmware uses 6800 of the total 8192 bytes of flash memory. This leaves sufficient room to add remote control via the RS-232 port in future. Because the firmware is written in BASIC, it's reasonably easy to go into the code and add additional features at a later date.

Although it was a bit of an overkill to use the ultra-fast NVE IsoLoop devices for this project, it made that part of the design rather easy. I'd like to thank NVE for quickly sending me a few samples to incorporate in my design.

CC SHOP

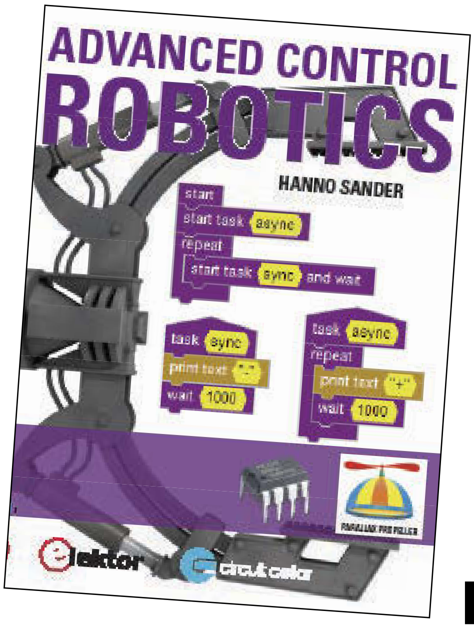
1

1 CC 2014 DIGITAL ARCHIVE SUBSCRIPTION

Just when you thought it couldn't get any easier than a thumb drive...you can now acess a full year of *Circuit Cellar* from any device connected to the Internet! (2014: 12 issues)

You get all the benifits of a printed copy—bookmark pages, make annotations, and write in the margins—combined with the digial advantages of easy storage, zoom, links, and search features.

Item #: CC-DA-2014



2

2 ADVANCED CONTROL ROBOTICS

When it comes to robotics, the future is now! This book simplifies the theory and best practices of advanced robot technologies by providing insight to communication technologies, control robotics, embedded technology, programming, and more. Complete with code samples, schematics, and design tips. Great reading for beginners and experts alike.

Author: Hanno Sander
Item#BK-ELNL-978-0-963013-33-0

3

3 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

Previous Years Also Available

4

4 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

Further information and ordering: www.cc-webshop.com
CONTACT US: Circuit Cellar, Inc. | Phone: 860.289.0800 | E-mail: custservice@circuitcellar.com

What's your EQ? The answers are posted at www.circuitcellar.com/category/test-your-eq/. You can contact the quizmasters at eq@circuitcellar.com.

TEST YOUR EQ

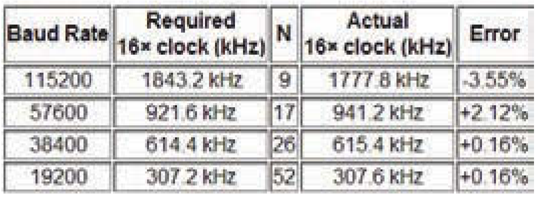
Contributed by David Tweed

Answers to the EQ problems presented in Circuit Cellar 296

ANSWER 1—The frequency generated at the QB output of the counter is 16.000 MHz × 3/13 = 3.6923 MHz. The ratio between this and 3.6864 MHz is 1.0016, so the error expressed as a percentage is +0.16%. This is well within the tolerance required for asynchronous serial communications.

ANSWER 2—The circuit generates rising edges (also falling edges) at intervals of 4 clocks, 4 clocks and 5 clocks, but the ideal spacing would be 4.3333 clocks. Therefore two of the intervals are short by 1/3 clock and one of them is long by 2/3 clock. Therefore, the cycle-to-cycle peak-to-peak jitter is 1/3 + 2/3 = 1 full input clock period, or 62.5 ns. But taking an average over a complete group of 13 clocks, no edge is displaced from its "ideal" location by more than 1/3 clock, or 20.8 ns.

ANSWER 3—The following table shows the divider ratios required for various standard baud rates.



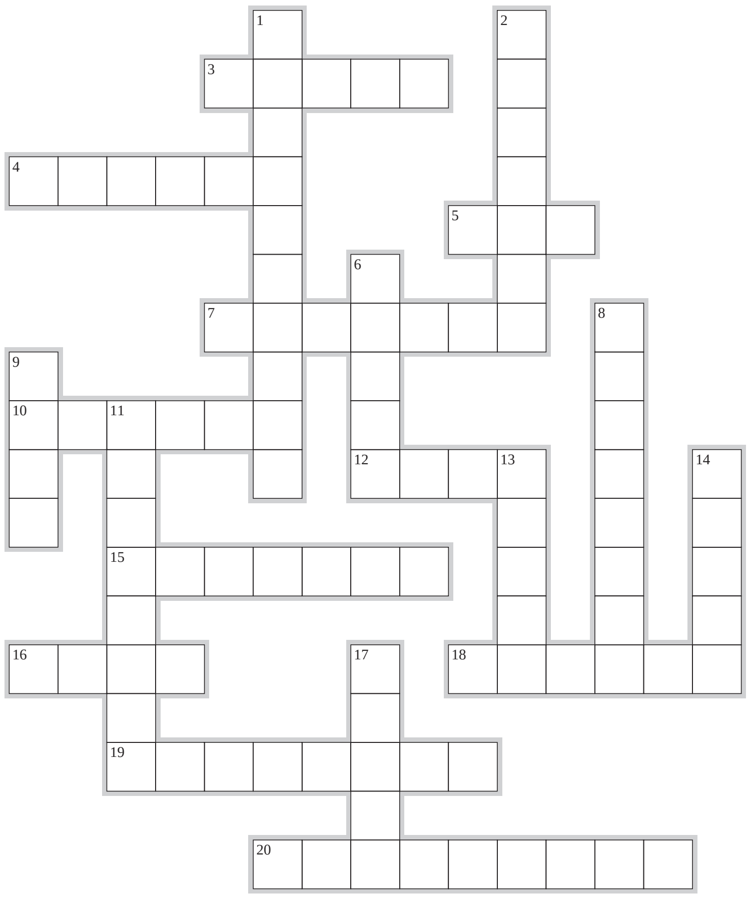
Baud Rate	Required 16× clock (kHz)	N	Actual 16× clock (kHz)	Error
115200	1843.2 kHz	9	1777.8 kHz	-3.55%
57600	921.6 kHz	17	941.2 kHz	+2.12%
38400	614.4 kHz	26	615.4 kHz	+0.16%
19200	307.2 kHz	52	307.6 kHz	+0.16%

As you can see, a modern UART can generate the clocks for baud rates up to 38400 with the exact same error as the 3/13 counter scheme—note that 26 and 52 are multiples of 13. But above that, the frequency error increases. This is why microcontrollers with built-in UARTs often run at "oddball" frequencies such as 11.0592 MHz or 12.288 MHz—these frequencies can be easily divided down to produce precisely correct baud rates.

ANSWER 4—A UART receiver waits for the leading edge of the start bit, and then samples the next 10 bits in the center of each bit "cell." If by the time it gets to the 10th cell, the sampling point at the receiver has moved beyond the edge of the 10th bit (the stop bit) defined by the transmitter, the transmission will fail. This means that the timing error must be no more than ± 1/2 bit over a 9.5-bit span, or a total error between transmitter and receiver of ±5.26%. If the error is split evenly, this means that each baud rate generator must be accurate to within ±2.63%. However, in reality, the receiver cannot determine the location of the leading edge precisely. Since it is using a 16× clock to do the sampling, there could be as much as 1/16 of a bit delay before the receiver actually recognizes the start bit, and all of its sampling points for the subsequent bits will be delayed by that amount. This means that the timing error must be no more than ± 7/16 of a bit by the time we get to the last bit, which means that the maximum total error is ±4.60%, or ±2.30% for each baud rate generator.

CROSSWORD APRIL 2015

The answers will be available at circuitcellar.com/category/crossword/



ACROSS

3. A switch actuated by another electrical signal
4. A permanent deviation from the design center of any device or circuit
5. A component for a PCB use that has pins arranged in two parallel rows
7. 1/10th of a Bel
10. Convert analog information to digital data
12. Suppress a resonance by absorbing stored energy through the application of mechanical or electrical friction
15. To define or set limits on the boundaries of something
16. Two electrically conductive paths carrying the same signal
18. State of having a stored excess or deficiency of electrons, which imparts electrostatic polarity to a capacitor or object
19. 10.1 mal equals 10 what?
20. The tendency for a rapidly increasing or decreasing electrical impulse to exceed momentarily its actual peak level

DOWN

1. 0.3937 inches
2. Sharing the same axis
6. A volume of space through which any radiated energy is distributed
8. A circuit that extracts the modulations from a carrier
9. An electrochemical unit for producing DC electricity
11. Heart-shaped
13. Reroute a signal to a different circuit
14. An "erg" is a unit of work equal to 1^{-7} of these
17. A digital signal comprises a series of these

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.

Rapid PIC® Development Tools
Powerful, Professional & Optimized for PIC®!

Blink an LED on any PIC®
with an easy-to-use C Compiler!

```
#include <PIC16F887.h>
#use delay(crystal=20mhz)
  void main(void) {
      while(TRUE) {
          output_high(PIN_B0);
          delay_ms(500);
          output_low(PIN_B0);
          delay_ms(500);
      }
  }
```

Download Free Evaluation:
www.ccsinfo.com/CC415
sales@ccsinfo.com
262-522-6500 x 35
CCS
PIC® MCU is a registered trademark of Microchip Technology Inc.

75 GHz Sockets
High Speed Microwave Applications
- Contact resistance <30 mOhms
- Bandwidth >75GHz @-1dB
- Current 5.4A @ 20C rise
- Force 50-80 grams per contact
- Operating temperature -55 to +160C
- Insertion/Extraction cycles: 1000
- Contact length (compressed): 0.08mm
- Compliance: 0.15mm

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

MaxBotix®
High Performance Ultrasonic Rangefinders

Call Carlson today for application specific sensors
Phone: 218-454-0766
Email: info@maxbotix.com
www.maxbotix.com

PIC-SERVO
MOTION CONTROL
MOTION CONTROLLERS FOR BRUSH, BRUSHLESS AND STEPPER MOTORS.
- controller chips
- controller boards

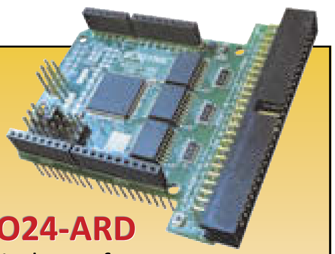
www.picservo.com
JEFFREY KERR, LLC

$20 for 5PCBs
2 layer,4x4inch, FR4(RoHS),0.063",1oz, 2LPI,Green,1SK,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

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

Shop for this book, and others, at www.cc-webshop.com

Connect
with engineers seeking your design solutions.
Magazine, e-newsletters, website, advertorials, and more.
RESERVE ADVERTISING SPACE TODAY!
Strategic Media Marketing, LLC
978.281.7708
circuitcellar@smmarketing.us
www.smmarketing.us



DIO24-ARD
Digital Interface
for **Arduino** and **Compatibles**
- 24 Programmable Input/Output Channels
- Connect I/O Racks, LEDs, Switches, and More
- 85ma Sink Output Capability
- Uses Standard SPI Software Library Functions
- Jumper Selectable and Off-Board SPI Enable
- Long Lead Stack-Through Connectors
- Industrial Operating Temperature Range
- RoHS Compliant

Learn More Details at . . .
SCIDYNE®.com *Call Toll-Free* 1-877-SCIDYNE (1-877-724-3963)

Security Agents for Embedded Intrusion Detection

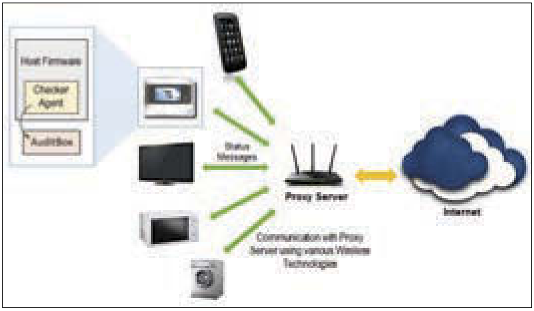
By Syed Kamran Haider, Devu Manikantan Shila, and Marten van Dijk

Syed Kamran Haider is pursuing a PhD in Computer Engineering supervised by Marten van Dijk at the University of Connecticut.

Dr. Devu Manikantan Shila is the Principal Investigator for Cyber Security area within the Embedded Systems and Networks Group at the United Technologies Research Center (UTRC).

Marten van Dijk is an Associate Professor of Electrical and Computing Engineering at the University of Connecticut, with over 10 years research experience in system security both in academia (MIT CSAIL) and industry (Philips Research and RSA Laboratories).

Knowingly or unknowingly, we interact with hundreds of networked-embedded devices in our day-to-day lives such as mobile devices, electronic households, medical equipment, automobiles, media players, and many more. This increased dependence of our lives on the networked-embedded devices, nevertheless, has raised serious security concerns. In the past, security of embedded systems was not a major concern as these systems were a stand-alone network that contained only trusted devices with little or no communication to the external world. One could execute an attack only with a direct physical or local access to the internal embedded network or to the device. Today, however, almost every embedded device is connected to other devices or the external world (e.g., the Cloud) for advanced monitoring and management capabilities. On one hand, enabling networking capabilities paves the way for a smarter world that we currently live in, while on the other hand, the same capability raises severe security concerns in embedded devices. Recent attacks on embedded device product portfolios in the Black Hat and Defcon conferences has identified remote exploit vulnerabilities (e.g., an adversary who exploits the remote connectivity of embedded devices to launch attacks such as privacy leakage, malware insertion, and denial of service) as one of the major attack vectors. A handful of research efforts along the lines of traditional security defenses have been proposed to enhance the security posture of these networked devices. These solutions, however, do not entirely solve the problem and we therefore argue the need for a light weight intrusion-defense capability within the embedded device.



In particular, we observe that the networking capability of embedded devices can indeed be leveraged to provide an in-home secure proxy server that monitors all the network traffic to and from the devices. The proxy server will act as a gateway performing policy based operations on all the traffic to and from the interconnected embedded devices inside the household. In order to do so, the proxy server will implement an agent based computing model where each embedded device is required to run a light weight checker agent that periodically reports the device status back to the server; the server verifies the operation integrity and signals back the device to perform its normal functionality. A similar approach is proposed Ang Cui and Salvatore J. Stolfo's 2011 paper, "Defending Embedded Systems with Software Symbiotes," where a piece of software called Symbiote is injected into the device's firmware that uses a secure checksum-based approach to detect any malicious intrusions into the device. In contrast to Symbiote, we exploit lightweight checker agents at devices that merely forward device status to the server and all the related heavy computations are offloaded to the proxy server, which in turn proves our approach computationally efficient. Alternatively, the proposed model incurs a very small computational overhead in gathering and reporting critical device status messages to the server. Also, the communication overhead can be amortized under most circumstances as the sensor data from the checker agents can be piggybacked to the original data messages being transferred between the device and the server. Our model, as what's described in the aforementioned Cui and Stolfo paper, can be easily integrated with legacy embedded devices as the only modification required to the legacy devices is a "firmware upgrade that includes checker agents."

To complete the picture, we propose an additional layer of security for modern embedded devices by designing an AuditBox, as in the article, "Pillarbox," by K. Bowers, C. Hart, A. Juels, and N. Triandopoulos. It keeps an obfuscated log of malicious events taking place at the device which are reported back to the server at predefined time intervals. This enables our server to act accordingly by either revoking the device from the network or by restoring it to a safe state. AuditBox will enforce integrity by being able to verify whether the logs at the device have been tampered with by an adversary who is in control of the device and covertness by hiding from an attacker with access to the device whether the log reports detection of malicious behavior. To realize these requirements, AuditBox will exploit the concept of forward secure key generation.

Embedded systems security is of crucial importance and the need of the hour. Along with the advancement in embedded systems technology, we need to put an equal emphasis on its security in order for our world to be truly a smarter place.

RESOURCES

K. Bowers, C. Hart, A. Juels, & N. Triandopoulos, "Pillarbox: Combating Next-Generation Malware with Fast Forward-Secure Logging," in Research in Attacks, Intrusions and Defenses, ser. Lecture Notes in Computer Science, A. Stavrou, H. Bos, and G. Portokalidis (Eds.), Springer, 2014, http://dx.doi.org/10.1007/978-3-319-11379-1_3.

A. Cui & S. J. Stolfo, "Defending embedded systems with software symbiotes," in Proceedings of the 14th international conference on Recent Advances in Intrusion Detection (RAID'11), R. Sommer, D. Balzarotti, and G. Maier (Eds.), Springer-Verlag, 2011, http://dx.doi.org/10.1007/978-3-642-23644-0_19.

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

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.

30th ANNIVERSARY 1985 - 2015

Imagineering
Winner
Family Entrepreneurship Award 2014
QUINLAN
SCHOOL of BUSINESS

imagineering inc. www.PCBnet.com

847-806-0003 sales@PCBnet.com
Certified Woman-Owned Small Business