TIPS: Debugging & Developing USB Devices
LOCATION: United States
PAGE: 30

INSIGHT: Effectively Predict Product Reliability
LOCATION: Canada
PAGE: 60

PROJECT: Tachometer Design Explained
LOCATION: United States
PAGE: 64

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

NOVEMBER 2012
ISSUE 268

ANALOG TECHNIQUES

Embedded Security: True Random Number Generation

Time Broadcasting: A GPS-Based Time Server

Build an Automated Electronics Component Dispenser

Q&A: The Work of an Aerospace Engineer & Successful DIYer



PLUS

Photo-Pal Complete

The Final Phases of the Camera Controller Project

// Software-Defined Buttons
// Parameter Entry
// Timing, Resolution, & Accuracy
// And More

$9.00US $10.00CAN
11>
0 74470 75349 0
www.circuitcellar.com

Now with 32MB Flash and 64MB RAM!



NANO54415

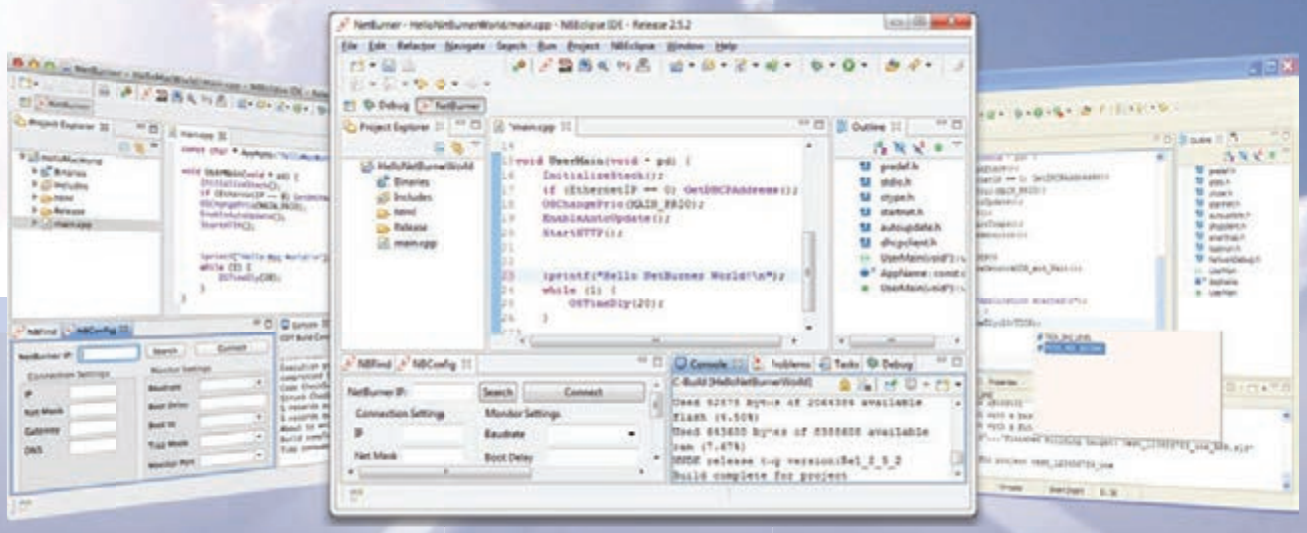
$69.00 Qty. 100



MOD54415

$89.00 Qty. 100

MOD54415 Core Module	NANO54415 Core Module
32-bit 250 MHz processor	32-bit 250 MHz processor
64MB DDR2 RAM	64MB DDR2 RAM
32MB flash	**8MB flash**
10/100 Mbps Ethernet	10/100 Mbps Ethernet
44 general purpose I/O	**30 general purpose I/O**
Eight UARTs	Eight UARTs
Five I2C	**Four I2C**
Two CAN	Two CAN
3 SPI	3 SPI
1-Wire®	1-Wire®
5 pulse width modulators (PWM)	**8 pulse width modulators (PWM)**
SSI	SSI
MicroSD flash card	MicroSD flash card ready
8 analog to digital converters (ADC)	6 analog to digital converters (ADC)
Two digital to analog converters (DAC)	Two digital to analog converters (DAC)



Quickly create and deploy applications from your Mac or Windows PC

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

Development Kit for MOD54415
Part No. NNDK-MOD54415-KIT
$99.00 for a limited time

Development Kit for NANO54415
Part No. NNDK-NANO54415-KIT
$99.00

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

WIRELESS POWER: CHARGING INNOVATION



Wireless power integration made easy with TI's Qi Compliant Wireless Power bqTESLA evaluation modules and solution portfolio from TI & Würth enable design engineers to easily accelerate the integration of wireless power technology in consumer electronics, such as smart phones, digital cameras, MP3 players, along with infrastructure applications such as furniture and cars. We'll help you cross the design finish line in record time with a wide range of evaluation modules both on the transmitter and receiver side to help reduce the design cycle of wireless power solutions. Whether implementing wireless charging within an existing design, or adding it to a new one, we've got the tools, support and expertise to help you – cut the cord! Make your own kit by selecting a TI transmitter and receiver module with corresponding Charging-Coils provided by Würth Elektronik.

Farnell element14

element14.com/wireless-power-solution

TASK MANAGER

A History of Improvement

At the end of September 2012, an enthusiastic crew of electrical engineers and journalists (and significant others) traveled to Portsmouth, NH, from locations as far apart as San Luis Obispo, CA, and Paris, France, to celebrate *Circuit Cellar*'s 25th anniversary. Attendees included Don Akkermans (Director, Elektor International Media), Steve Ciarcia (Founder, Circuit Cellar), the current magazine staff, and several well-known engineers, editors, and columnists. The event marked the beginning of the next chapter in the history of this long-revered publication. As you'd expect, contributors and staffers both reminisced about the past and shared ideas about its future. And in many instances, the conversations turned to the content in this issue, which was at that time entering the final phase of production. Why? We purposely designed this issue (and next month's) to feature a diversity of content that would represent the breadth of coverage we've come to deliver during the past quarter century. A quick look at this issue's topics gives you an idea of how far embedded technology has come. The topics also point to the fact that some of the most popular '80s-era engineering concerns are as relevant as ever. Let's review.

In the earliest issues of *Circuit Cellar*, home control was one of the hottest topics. Today, inventive DIY home control projects are highly coveted by professional engineers and newbies alike. On page 16, Scott Weber presents an interesting GPS-based time server for lighting control applications. An MCU extracts time from GPS data and transmits it to networked devices.

Thiadmer Riemersma's DIY automated component dispenser is a contemporary solution to a problem that has frustrated engineers for decades (p. 26). The MCU-based design simplifies component management and will be a welcome addition to any workbench.

USB technology started becoming relevant in the mid-to-late 1990s, and since then has become the go-to connection option for designers and end users alike. Turn to page 30 for Jan Axelson's tips about debugging USB firmware.

Electrical engineers have been trying to "control time" in various ways since the earliest innovators began studying and experimenting with electric charge. Contemporary timing control systems are implemented in a amazing ways. For instance, Richard Lord built a digital camera controller that enables him to photograph the movement of high-speed objects (p. 36).

Security and product reliability are topics that have been on the minds of engineers for decades. Whether you're working on aerospace electronics or a compact embedded system for your workbench (p. 52), you'll want to ensure your data is protected and that you've gone through the necessary steps to predict your project's likely reliability (p. 60).

The issue's last two articles detail how to use contemporary electronics to improve older mechanical systems. On page 64 George Martin presents a tachometer design you can implement immediately in a machine shop. And lastly, on page 70, Jeff Bachiochi wraps up his series "Mechanical Gyroscope Replacement." The goal is to transmit reliable data to motor controllers.

cj@circuitcellar.com



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

EDITORIAL CALENDAR

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

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

Communications: Projects that deal with computer networking, human-to-human interaction, human-to-computer interaction, and electronic information sharing (e.g., speech recognition, data transmission, Ethernet, USB, I²C, and SPI)

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

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

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

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

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

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

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

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

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

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

UPCOMING IN CIRCUIT CELLAR

FEATURES

EBike Meter: Computer and Data Logger Design, by Dan Karmann

Electrically Actuated Sound Effects, by Joe Pfeiffer

Task-Specific Interconnected Device Development, by Scott Weber

Winners: Renesas RL78 Green Energy Challenge 2012

COLUMNS

Energy Extraction, by Jeff Bachiochi

Arduino Survival Guide, by Ed Nisley

Locked In: Synchronous Detection Explained, by Robert Lacoste

Product Reliability (Part 2): Failure Rate, by George Novacek

Concurrency in Embedded Systems (Part 4), by Bob Japenga

ROUTE FASTER !



WITH PROTEUS PCB DESIGN

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

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

PROTEUS DESIGN SUITE

- Hardware Accelerated Performance.
- Unique Thru-View™ Board Transparency.
- Over 35k Schematic & PCB library parts.
- Integrated Shape Based Auto-router.
- Flexible Design Rule Management.
- Polygonal and Split Power Plane Support.

Features:

- Board Autoplacement & Gateswap Optimiser.
- Direct CADCAM, ODB++, IDF & PDF Output.
- Integrated 3D Viewer with 3DS and DXF export.
- Mixed Mode SPICE Simulation Engine.
- Co-Simulation of PIC, AVR, 8051 and ARM7.
- Direct Technical Support at no additional cost.

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

labcenter Electronics

www.labcenter.com

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

INSIDE ISSUE 268

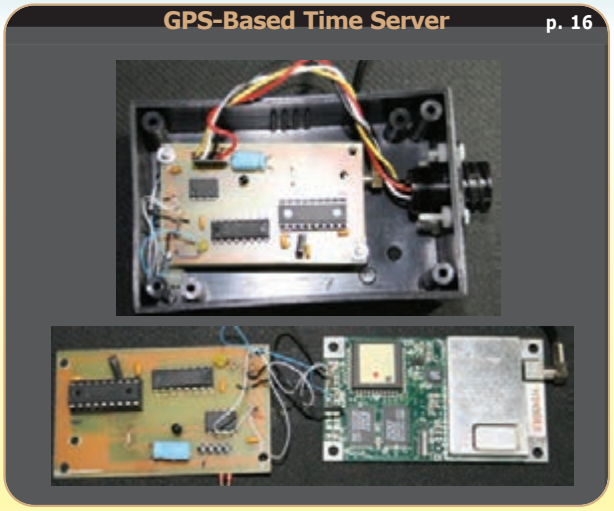
November 2012 • Analog Techniques

16 **Time Broadcasting**
A GPS-Based Time Server for the RS-485 Network
Scott Weber

26 **DIY Automated Component Dispenser**
Thiadmer Riemersma

30 **Debugging USB Firmware**
Tips and Tricks for Developing USB Devices
Jan Axelson

36 **Digital Camera Controller (Part 2)**
Code, User Interface, and Timing
Richard Lord









52 **EMBEDDED SECURITY**
True Random Number Generation
Patrick Schaumont

60 **THE CONSUMMATE ENGINEER**
Product Reliability (Part 1)
Reliability Prediction
George Novacek

64 **LESSONS FROM THE TRENCHES**
Tachometer Design
George Martin

70 **FROM THE BENCH**
Mechanical Gyroscope Replacement (Part 2)
Gravity and Acceleration
Jeff Bachiochi

TASK MANAGER 2
A History of Improvement
C. J. Abate

NEW PRODUCT NEWS 10

MEMBER PROFILE 14

TEST YOUR EQ 15

QUESTIONS & ANSWERS 44
Hands-On Innovation
An Interview with David Penrose
Nan Price

CROSSWORD 76

PRIORITY INTERRUPT 80
An Internet Education
Steve Ciarcia

AS9120A
Certified Distributor

Scan Here

mouser.com
Semiconductors and electronic components for design engineers.

Authorized Distributor

We deliver ASAP, PDQ and JIT. So, you're never SOL.

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

mouser.com

The Newest Products for Your Newest Designs®

MOUSER ELECTRONICS
a tti company

Mouser and Mouser Electronics are registered trademarks of Mouser Electronics, Inc. The Newest Products for Your Newest Designs is a registered trademark of Mouser Electronics, Inc.

The team

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

The network

audioXPRESS

elektor

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

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

VOICE COIL

Our international teams

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Issue 268 November 2012 ISSN 1528-0608

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

Cover photography by Chris Rakoczy—www.rakoczyphoto.com

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

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

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

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

MEMBERSHIP COUNTER

We now have  members in  countries.

Not a member yet?

Sign up at www.circuitcellar.com

SUPPORTING COMPANIES

2013 International CES 59
All Electronics Corp. 79
AP Circuits 62
ARM. 39
Beta Layout, Ltd. 61
BusBoard Prototype Systems. 78
Butterfly Network, Inc. 13
Cadsoft Computer GmbH 1
Circuit Cellar 25th Anniversary USB 69
Cleverscope. 41
Comfile Technology 55
Custom Computer Services 77
DesignSpark 49
Elektor 42, 43
Elektor 63
EMAC, Inc. 61
ExpressPCB. 33
Front Panel Express 24
FTDI Chip. C3
Grid Connect, Inc. 24
Holtek Semiconductor, Inc. 35
Humandata, Ltd. 57
Imagineering, Inc. 17, 19, 21, 23
Ironwood Electronics 79
Jameco Electronics 51
Jeffrey Kerr, LLC. 79
JK microsystems, Inc. 13, 79
Labcenter Electronics 3
LinkSprite 78
MaxBotix, Inc. C4
MCC, Micro Computer Control 79
Microengineering Labs, Inc. 77
Mosaic Industries, Inc. 79
Mouser Electronics, Inc. 5
NetBurner C2
Newark element14 47
PCBMAIN Technology Co., Ltd. 77
Pico Technology, Ltd. 29
Pololu Corp. 25
Rigol Technologies 15
Saelig Co., Inc. 20
SiliconRay 41
Sealevel Systems. 40
Technologic Systems 8, 9
Tern, Inc. 78
Triangle Research International, Inc. 78

Not a supporting company yet?

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

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

Copyright Notice
Entire contents copyright © 2012 by Circuit Cellar, Inc. All rights reserved. Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

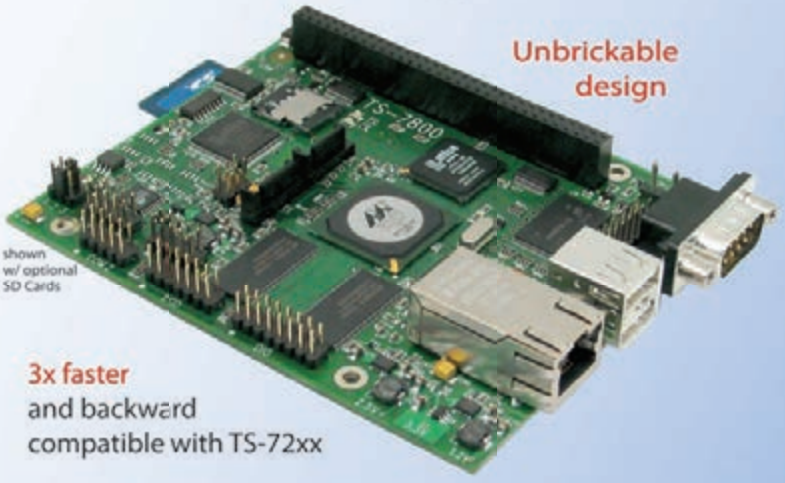
Disclaimer
Circuit Cellar® makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled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

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

© Circuit Cellar 2012 Printed in the United States

Embedded Systems

High-End Performance with Embedded Ruggedness



TS-7800
500MHz ARM9

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

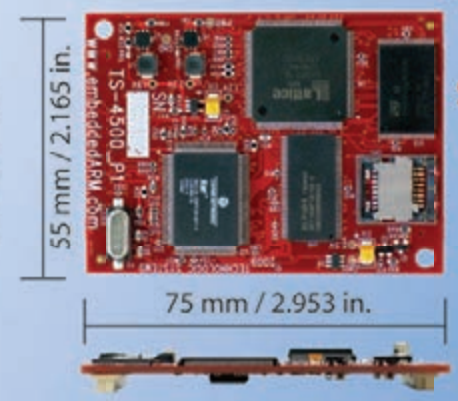
$229 qty 100
$269 qty 1

TS-SOCKET Macrocontrollers
Jump Start Your Embedded System Design

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

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

series starts at
$92 qty 100
$139 qty 1



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

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

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

New Products

Touch Panel Computers
800MHz with Video Acceleration

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



Fully enclosed TPC available Q3

NEW!



series starts at
$415
qty 100

$479
qty 1



picture of TS-8820-BOX

NEW!

series starts at
$199
qty 100

$229
qty 1

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

Industrial Controllers
Powerful, Rugged, Affordable

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

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

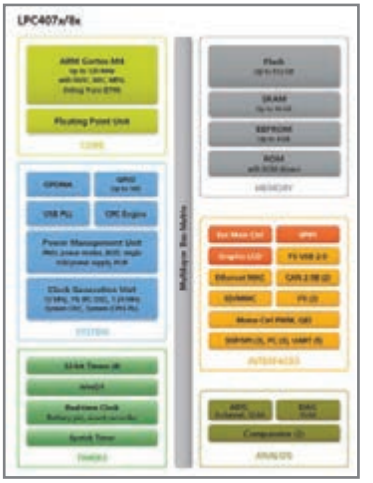
www.embeddedARM.com

EXPANDED DIGITAL SIGNAL CONTROLLER FAMILY & DEVELOPMENT BOARD

The **LPC408*x*** and **LPC407*x*** microcontrollers feature advanced signal-processing capabilities and many connectivity options, including USB 2.0, Ethernet, and CAN 2.0 B. Based on an ARM Cortex-M4 processor, the LPC4000 series of microcontrollers provide drop-in compatibility with NXP's LPC178*x* and LPC177*x* series and many other LPC2000 microcontrollers. The cost-effective, low-power LPC408*x* and LPC407*x* are ideal digital-signal control solutions for displays, scanners, industrial networking, alarm systems, medical diagnostics, and motor-control applications.



The LPC408*x* and LPC407*x* operate at speeds up to 120 MHz, and provide up to 512 KB of flash memory, 96 KB of SRAM, 4 KB of EEPROM, and two analog comparators. The devices also provide a range of connectivity peripherals, including up to five UARTs, three SPI/SSP, and three I²C interfaces. The LPC408*x* and LPC407*x* feature a multilayer AHB bus that enables high-bandwidth peripherals (e.g., Ethernet and full-speed USB) to simultaneously run without affecting performance. Additional serial peripherals include two CAN controllers, SD/MMC, and an I²S interface.



Similar to NXP's dual-core LPC4300 family, the LPC408*x* and LPC407*x* microcontrollers feature an optional 32-bit floating-point unit and a graphical LCD controller providing 768 × 1,024 pixel display resolution. The LPC microcontrollers also feature NXP's SPI flash interface (SPIFI), which enables you to add a significant amount of low-cost memory.

Featuring the NXP LPC4078 microcontroller, the **Code Red RDB4078** development board includes a 240 × 320 in-plane switching (IPS) touchscreen with a wide viewing angle, Ethernet, RS-232, a USB device and host, a microSD, and advanced audio functionality including high-quality Wolfson stereo codecs and stereophonic digital microphones. The RDB4078's I/O functionality is simultaneously accessible without the need for reconfiguration and it includes on-board debug for instant debugging, which utilizes the LPC4000 support available in LPCXpresso and Red Suite. A suite of example applications—including an embedded web server, audio examples, and an SD-card filing system—is also available.

Contact NXP for pricing.

NXP Semiconductors
www.nxp.com

IDE SUPPORTS 200+ DEVICES & FEATURES DUAL-CORE DEBUGGING

TrueSTUDIO v3.2 is the latest version of Atollic's integrated development environment (IDE) for embedded microcontroller designs. The updated version introduces native RTOS-aware debugging for RTOSes including: FreeRTOS, OpenRTOS, ThreadX, and embOS. Atollic TrueSTUDIO v3.2 also provides an integrated interface to the Micrium µC/Probe tool that provides RTOS-aware debugging for the µC/OS-III RTOS. RTOS-aware debugging provides insight to the RTOS status (e.g., message queues and tasks) to help you comprehend how your application is being handled within the RTOS.



TrueSTUDIO v3.2 supports more than 200 new target devices including: the new STMicroelectronics's STM32F3 family, Freescale Semiconductor's Kinetis L series, ARM's Cortex-A class application processors, and many new Fujitsu microcontrollers. TrueSTUDIO v3.2 also supports more than 1,000 ARM-based microcontrollers and more than 70 evaluation boards, including Freescale's Freedom Development platform and STMicroelectronics's STM32 microcontroller series. An integrated GUI client for the Git version control system was also added. Other features of the IDE and debugging tool include: a state-of-the-art editor, an optimizing C/C++ compiler, and a multiprocessor-aware debugger with real-time tracing.

Atollic offers a family of code analysis and validation tools to complement TrueSTUDIO v3.2. TrueINSPECTOR provides static source code analysis with MISRA-C compliance checking and code metrics including code complexity measurements. TrueANALYZER utilizes MC/DC-level coverage using dynamic execution flow analysis to facilitate in-target test quality measurement. TrueVERIFIER uses auto-generated and auto-executed unit test suites on the target board to provide an embedded test automation function.

Contact Atollic for pricing.

Atollic
www.atollic.com

SMARTCARD ICs & DISCOVERY KIT WITH MEMS RESISTORS

The **ST23ZR** family of secure microcontrollers includes dual-interface devices supporting ISO 14443-A/B contactless protocols and the ISO 7816-3 standard for contact-type cards. The devices can create one software application that can be used with either standard. The microcontrollers automatically distinguish between the protocols and respond with the appropriate one.



The ST23ZR family is well suited for transport, banking, and e-ID applications. Because they operate smoothly with older, single-protocol installed readers, the microcontrollers can also be used for applications such as mass transit, electronic payment, or electronic ID cards. The ST23ZC family also provides wireless communication for contactless-only applications.

The microcontrollers utilize tamper-proof hardware to execute robust security algorithms (e.g., Triple DES and AES 256). The ST23ZR/ZC devices also meet government and credit card security standards including Common Criteria EAL5+ and EMVCo to provide protection from fraud and identity theft.

The ST23ZR/ST23ZC families' additional features include: an enhanced 8-/16-bit ST23 processor core, low power consumption, 96 Kb of user ROM, up to 8 Kb of user EEPROM, support for ISO 14443-A and ISO 14443-B contactless protocols, support for all 106-to-848-kbps contactless data rates, and a dual interface for contact (ISO 7816) for contactless applications.



STMicroelectronics also introduced the **STM32 F3 Discovery Kit**, which features built-in microelectromechanical systems (MEMS) sensors (gyroscope and e-compass) with nine degrees of freedom (DOF). The microcontroller family's advanced signal processing and arithmetic capabilities can enable sensor-fusion applications, such as attitude heading reference systems (AHRS) and can help designers take advantage of 3-D motion-sensing systems in many applications (e.g., mobile gaming, augmented reality, optical image stabilization, portable navigation, robotics, and industrial automated systems).

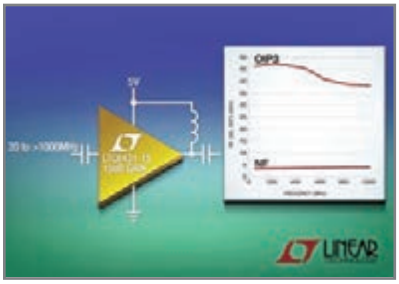
The STM32 F3 Discovery Kit includes a ready-to-use prototype board containing an STM32F303 microcontroller and associated chips, indicator LEDs, push-button controls, I/O pin headers, and a USB connection for the host PC. The board's MEMS devices include STMicro's L3GD20 three-axis digital gyroscope and LSM303DLHC ultra-compact, high-performance e-compass. The kit is compatible with STM32 software-development environments from vendors including Altium, Atollic, IAR, and Keil.

Pricing for the STM32F3 discovery kit starts at **$10.90** for 1,000-unit quantities. Contact STMicroelectronics for pricing for the ST23ZR microcontroller family.

STMicroelectronics
www.st.com

GAIN BLOCK WITH HIGH DYNAMIC RANGE

The **LTC6431-15** is a 15.5-dB gain block that achieves 20-MHz-to-1-GHz high dynamic range in a 50-Ω environment. The gain block is manufactured on an advanced SiGe process and is available in two performance grades. At 240 MHz, the A-grade OIP3 is typically 47 dBm and is fully tested and guaranteed for a minimum of 44 dBm. The noise figure is 3.33 dB, which corresponds to an input-referred amplifier noise of 1 nV/√Hz. Power consumption is less than 450 mW. This combination of low noise and low distortion makes the LTC6431-15 well suited for high-performance intermediate-frequency communications and CATV applications.



The easy to use LTC6431-15 is unconditionally stable. Its input and output are internally matched to 50 Ω. Its only external requirements are DC blocking capacitors and an RF bias choke. The gain block provides greater than 2-V linear output swing operating on a single 5-V supply. P1dB is typically 20.6 dBm and linearity is maintained to 17-dBm output power.

The LTC6431-15 is available in a 4-mm × 4-mm QFN-24 package with an exposed pad for advanced thermal performance and low inductance. It is specified for operation from –40°C to 85°C case temperature. Pricing starts at **$4.89** each for the A-grade version and **$1.89** each for the B-grade version, both in 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

CRYSTAL-FREE USB MICROCONTROLLERS

Microchip Technology added three enhanced midrange 8-bit families to its full-speed USB 2.0 device PIC microcontroller portfolio. The 15 scalable microcontrollers range from 14 to 100 pins with up to 128 KB of flash memory. The three families include internal clock sources with 0.25% clock accuracy for USB communication, which eliminates the need for an external crystal. The devices are eXtreme Low Power-compliant, with 35-µA/MHz power consumption in Active mode and a 20-nA power consumption in Sleep mode.



The 14- and 20-pin **PIC16F145*x*** microcontrollers are low cost and feature a small form factor USB. This three-member family is available in 4-mm × 4-mm packages and includes a variety of integrated peripherals. The microcontrollers enable embedded applications that require USB connectivity and capacitive-touch sensing (e.g., pulse oxymeters, PC accessories, and security dongles).

The 28- and 40-/44-pin **PIC18F2*x*/4*x*K50** devices offer a cost-effective, pin-compatible migration option for customers utilizing legacy PIC18 USB microcontrollers. This three-member family features 1.8-to-5-V operation. It integrates a "Charge Time Measurement Unit" for higher-performance capacitive-touch sensing and measurement in applications (e.g., audio docks and data loggers).

The full-featured **PIC18F97J94** family offers integrated LCD control, RTCC with VBAT, and USB on a single 8-bit PIC microcontroller. The nine-member family, which is available in 64, 80, and 100 pins, offers a 8 × 60 LCD controller (for a total of 480 segments). The controller eliminates the need for an external controller in applications with large segmented displays and integrates a real-time clock/calendar with battery back-up for end products (e.g., home-automation/security panels, handheld scanners, and single-phase energy meters).

To help speed development times, the downloadable and open-source USB framework within the free Microchip Library of Applications (MLA) includes USB drivers for many common USB classes, including HID, CDC, Mass Storage, Win-USB, and Audio-MIDI. These drivers can be used with all 15 of the new PIC microcontrollers.

In addition to providing free USB software drivers and stacks, Microchip hardware development tools are also available. The low-pin-count USB Development Kit costs **$39.99** and is for use with the PIC16F145*x* family. The PICDEM FS-USB Board costs **$59.99** and is for use with the PIC18F*xx*K50 family. The PIC16F1454 and PIC16F1455 are offered in 14-pin SOIC, TSSOP, PDIP, and 4-mm × 4-mm 16-pin QFN packages. The PIC16F1459 is available in 20 pins with the same package types and sizes. Pricing starts at **$0.50** each.

Microchip Technology, Inc.
www.microchip.com

ACHIEVE SAFETY-CRITICAL DESIGN DEVELOPMENT & CERTIFICATION

SafeTI design packages are designed to accelerate safety-critical product development and help manage systematic and random failures. They are well suited for a range of products in markets including industrial, automotive, transportation, energy, and medical. SafeTI-61508 design packages include 15 Hercules RM4x ARM Cortex-R4 safety microcontrollers and Texas Instruments's (TI's) complementary TPS65381-Q1 multi-rail power supply for industrial, medical, and energy motor-control applications.

SafeTI design packages include five key components for functional safety: functional safety-enabled semiconductor components developed as safety-standard-compliant items; safety documents (e.g., a safety manual detailing product safety architecture and recommended usage, a safety analysis report including details of safety analysis, and a safety report summarizing compliance to targeted standards), tools, and software to decrease development and certification time; complementary embedded processing and analog components, which work together to help designers meet safety standards; a quality manufacturing process, which helps ensure that SafeTI components meet the component-level requirements concerning ISO9 001 and/or ISO/TS 16949 (including AEC-Q100 for automotive); and a safety-development process that follows ISO 26262, IEC 61508, and IEC 60730 requirements that is assessed by auditors as prescribed by safety standards.

SafeTI's development tools, software, and support for safety-critical development include the SafeTI ARM Compiler Qualification Package, which helps you document, analyze, validate, and qualify the usage of the TI ARM compiler to help meet the requirements of the ISO 26262 and IEC 61508 standards. In addition, the SafeTI HALCoGen graphical user interface (GUI) configures peripherals, interrupts, and clocks and generates peripheral and driver code.

Designers can obtain TI's Microcontroller Abstraction Layer (MCAL) 4.0 and Safe Automotive Open System Architecture (AutoSAR) from TTTech/Vector. ISO 26262 AutoSAR support is available from Vector and Elektrobit. SafeTI also features certifiable real-time operating system (RTOS) support for IEC 61508. RTOS support is available from Wittenstein High Integrity Systems's SAFERTOS, Micrium's µC/OS, Express Logic's ThreadX, and SCIOPTA RTOS.

Contact TI for pricing.

Texas Instruments, Inc.
www.ti.com

NPN

TYPE 6 MODULE FEATURES INTEL CORE PROCESSOR

The **Express-HR** is a high-performance COM.0 R2.0 Type 6 module. It features an Intel Core i7/i5 processor and supports the latest digital graphics interfaces for future designs. The module offers high-level processing and graphics performance making it well suited for medical, gaming, and military applications.

The Express-HR features the Intel Core i7/i5 processor supporting Intel Hyper-Threading Technology (four cores, eight threads) and up to 16 GB of DDR3 dual-channel memory at 1,066/1,333 MHz on dual-stacked SODIMM sockets. Intel Flexible Display Interface and Direct Media Interface provide high-speed connectivity to the Mobile Intel QM67 Express chipset. Intel HD Graphics is integrated on the CPU and a PCI Express x16 bus is available for discrete graphics expansion or general-purpose PCIe.

Benefits of the Type 6 module include three digital display interface (DDI) ports supporting HDMI, DVI, and DisplayPort outputs, in addition to legacy VGA and dual-channel 18-/24-bit low-voltage differential signaling (LVDS) displays. The Express-HR is designed for customers with high-performance processing and graphics requirements who want to outsource their systems' core logic and focus on their core competency to reduce development time.



The Express-HR features gigabit Ethernet, up to eight USB 2.0 ports, two SATA 6-Gbps ports, two SATA 3-Gbps ports (RAID 0/1/5/10), and support for SMBus and I^2C. The module is equipped with SPI AMI EFI BIOS supporting a remote console, a CMOS backup, a hardware monitor, and a watchdog timer.

Contact ADLINK for pricing.

ADLINK Technology, Inc.
www.adlink.com

NPN

STATEMENT REQUIRED BY THE ACT OF AUGUST 12, 1970, TITLE 39, UNITED STATES CODE SHOWING THE OWNERSHIP, MANAGEMENT AND CIRCULATION OF CIRCUIT CELLAR, THE MAGAZINE FOR COMPUTER APPLICATIONS. Published monthly at 4 Park Street, Vernon, CT 06066. Annual subscription price is $50.00. Publisher: Hugo Van haecke. Editorial Director: Steven Ciarcia. Editor-In-Chief C.J. Abate. The owner is Circuit Cellar, Inc., Vernon, CT 06066. The names and addresses of stockholders holding one percent or more of the total amount of stock are: Elektor International Media, LLC, 4 Park Street, Vernon, CT 06066.
EXTENT AND NATURE OF CIRCULATION: Average number of copies of each issue published during the preceding twelve months; (A) total number of copies printed, 15,363; (B.1) paid/requested mail subscriptions, 4,355; (B.3) sales through dealers and carriers, street vendors, and counter sales, 4,329; (B.4) paid/requested copies distributed by other mail classes, 10; (C) total paid/requested circulation, 8,694; (D.1) samples, complimentary, and other nonrequested copies,1,676; (D.4) Nonrequested copies distributed outside the mail, 2,214; (E) total nonrequested distribution (sum of D.1 & D.4), 3,890; (F) total distribution (sum of C & E), 12,584; (G) copies not distributed (office use, leftover, unaccounted, spoiled after printing, returns from news agents), 2779; (H) total (sum of F & G), 15,363. Percent Paid Requested: 69% Actual number of copies of a single issue published nearest to filing date: (A) total number of copies printed, 14,500; (B.1) paid/requested mail subscriptions, 4,934; (B.3) sales through dealers and carriers, street vendors, and counter sales, 3,967; (B.4) paid/requested copies distributed by other mail classes, 10; (C) total paid/requested circulation, 8,911; (D.1) samples, complimentary, and other nonrequested copies, 1,162; (D.4) Nonrequested copies distributed outside the mail, 2406; (E) total nonrequested distribution (sum of D.1 & D.4), 3,568; (F) total distribution (sum of C & E), 12,479; (G) copies not distributed (office use, leftover, unaccounted, spoiled after printing, returns from news agents), 2,021; (H) total (sum of F & G), 14,500. Percent Paid Requested 71%. I certify that the statements made by me above are correct and complete. Hugo Van haecke, Publisher.

BUTTERFLY Network, Inc.

GOD-LIKE ENGINEERS & PROGRAMMERS:

BUILD A 23RD CENTURY MEDICAL DEVICE

Butterfly Network is a fully-funded startup backed by entrepreneurs who have already shaped our future. Be at the intersection of Computer Science, Electrical Engineering and Medicine as we develop advanced diagnosis and treatment technologies that will transform medicine.



WANT TO CHANGE THE WORLD? JOIN US.

- FPGA and Embedded Design Engineer
- 3D Visualization Computer Scientist
- Device Driver and Software Engineer
- Signal and Image Processing Specialist
- Electrical Engineer or Applied Physicist

info@butterflynetinc.com | www.butterflynetinc.com

What's your flavor Linux or DOS?

OmniFlash



Linux Kernel 2.4 Installed
16MB FLASH / 32MB RAM
200 Mhz Arm9 CPU
16 Digital I/O Lines
10/100 Base –T Ethernet
2 USB and 2 Serial Ports
Hardware Clock Calendar
Watchdog and Audio In/Out
5V DC Power
$169.00
$129.00 Qty 100

Flashlite 186

Preloaded with DOS & Flash File System
33MHz 186 Compatible Processor
512K Flash, 512K DRAM
32 Pin DIP Socket
44 Digital I/O Lines
2 Serial Ports
Console / Debug Port
Watchdog & (2) 16 bit Timers
7-34V DC or 5V Power
$83.00
$66.00 Qty 100



Made in the USA - Distributed Worldwide

JK microsystems, Inc.
530-297-6073 www.jkmicro.com sales@jkmicro.com

MEMBER PROFILE: Marty McLeod

Member Name: Marty McLeod

Location: Alpharetta, GA

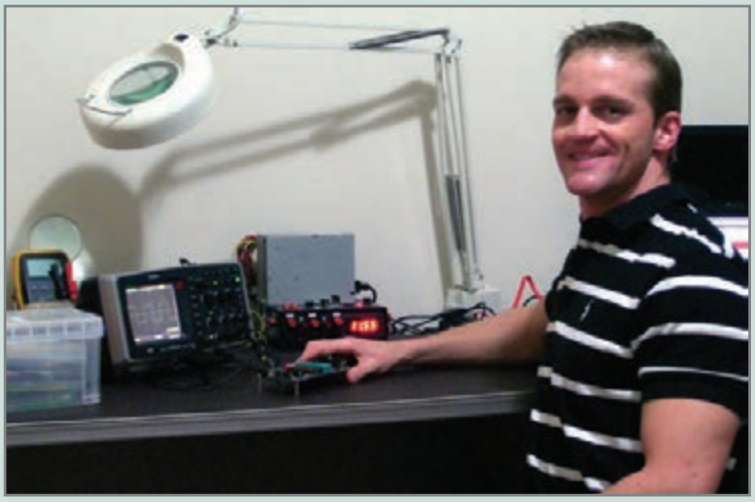
Education: BS in Electrical Engineering (Auburn University) and Associate's Degree in Electronics Technology (West Georgia Technical College)

Occupation: Embedded Firmware Engineer

Member Status: Marty has been a subscriber since late 2009.

Technical Interests: Marty's interests include general embedded technology, real-time operating systems (RTOSes), and audio (solid-state and tube-based).

Current Projects: Marty says he is working on general PIC-based designs with RTOS applications and touchscreen interfaces.



Thoughts on the Future of Embedded Technology: "I think we may eventually see multicore devices introduced into the less expensive world of microcontrollers," Marty said. "I expect to see changes in development tools toward the use of more open-source tools. I anticipate 32-bit devices becoming more popular than some 8- or 16-bit devices currently in use," he added.



CIRCUIT CELLAR

Test Your EQ

Problem 1—A transformer's windings, when measured individually (all other windings disconnected), have a certain amount of inductance. If you have a 1:1 transformer (both windings have the same inductance) and connect the windings in series, what value of inductance do you get?

Problem 2—If you connect the windings in parallel, what value of inductance do you get?

Problem 3—Suppose you have a 32-bit word in your microprocessor, and you want to count how many contiguous strings of ones appear in it. For example, the word "01110001000111101100011100011111" contains six such strings. Can you come up with an algorithm that uses simple shifts, bitwise logical and arithmetic operators, but—here's the twist—does not require iterating over each bit in the word?

Problem 4—For the purpose of timing analysis, the operating conditions of an FPGA are sometimes known as "PVT," which stands for "process, voltage, and temperature." Voltage and temperature are pretty much self-explanatory, but what does process mean in this context?

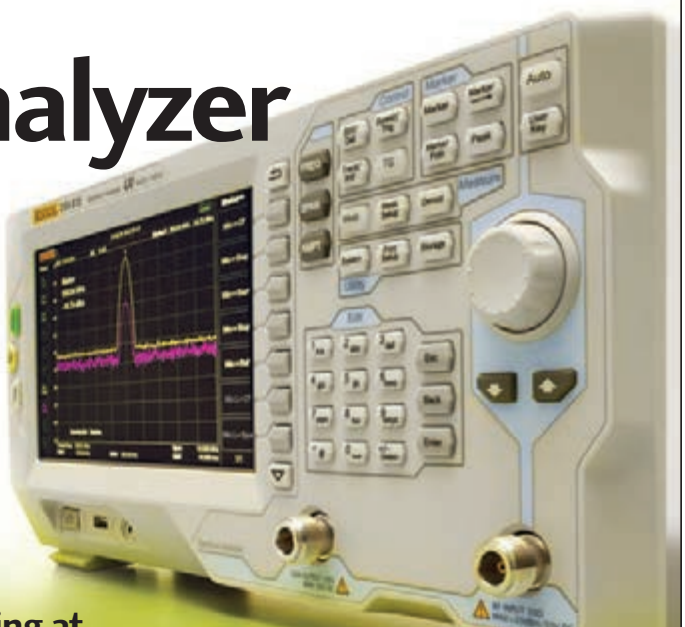
Contributed by David Tweed

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

NEW! DSA815
Spectrum Analyzer

Our new analyzer with all-digital IF technology redefines the product category!

- 9 kHz to 1.5 GHz Frequency Range
- Typical -135 dBm Displayed Average Noise Level (DANL)
- -80 dBc/Hz @10 kHz offset Phase Noise
- Total Amplitude Uncertainty <1.5 dB
- 100 Hz Minimum Resolution Bandwidth (RBW)



Get a Spectrum Analyzer at oscilloscope prices!

Starting at $1,295

RIGOL Beyond Measure

Before your next compliance test, check out the DSA815... save one trip to the compliance lab and it pays for itself!

RIGOLNA.com

by Scott Weber (USA)

Time Broadcasting

A GPS-Based Time Server for the RS-485 Network

If you're interested in automating a home or building, you'll need to construct and install light controllers. Once you do, you'll need an automated way to keep accurate time. Instead of manually setting the time on each device, you can use a microcontroller to extract the time from GPS data and forward the time to other devices. This keeps all the devices synchronized and helps them recover from power outages. This article explains how the GPS data is read and how it is used to correctly automate a set of light timers.

In my last article, "MCU-Based Light Control: Longer Serial Communication on Differential Wires" (*Circuit Cellar* 265, 2012), I described how I designed and installed a RS-485 network in my home with the intention of connecting a couple of devices to control my exterior lights. Driven by my need to improve an inferior timer, I decided this was an opportunity to create some task-specific devices I would interconnect in my home. Once the controller that turns the lights on and off was in place, I needed to keep everything on the same time, even after a power outage. Constantly using my laptop to connect to the system was not a good solution. So I built an inexpensive time broadcasting device that periodically broadcasts the current time on the network (see Photo 1).

The big challenge was determining where time originates. There are a number of solutions, including a real-time clock (RTC) on a chip, Internet time, and radio time. RTC on a chip would still need to be synchronized to something occasionally, even if it was backed up by a battery. Internet time would require a TCP/IP-enabled component that could connect through my home's high-speed Internet connection. Radio-based time could be obtained from a WWVB receiver, which can be spotty at times. In the end, the most cost-effective solution was to use an old GPS board from the bottom of my junk box. This meant minimal code and minimal parts, which I liked.

CAN YOU SPARE THE TIME?

GPS modules provide information in "Time/Fix" messages, which are so named because the message contains

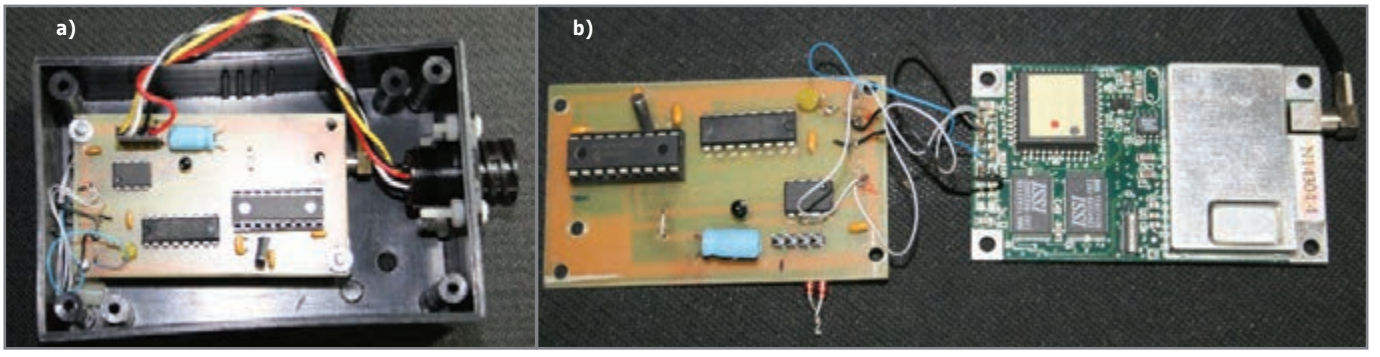
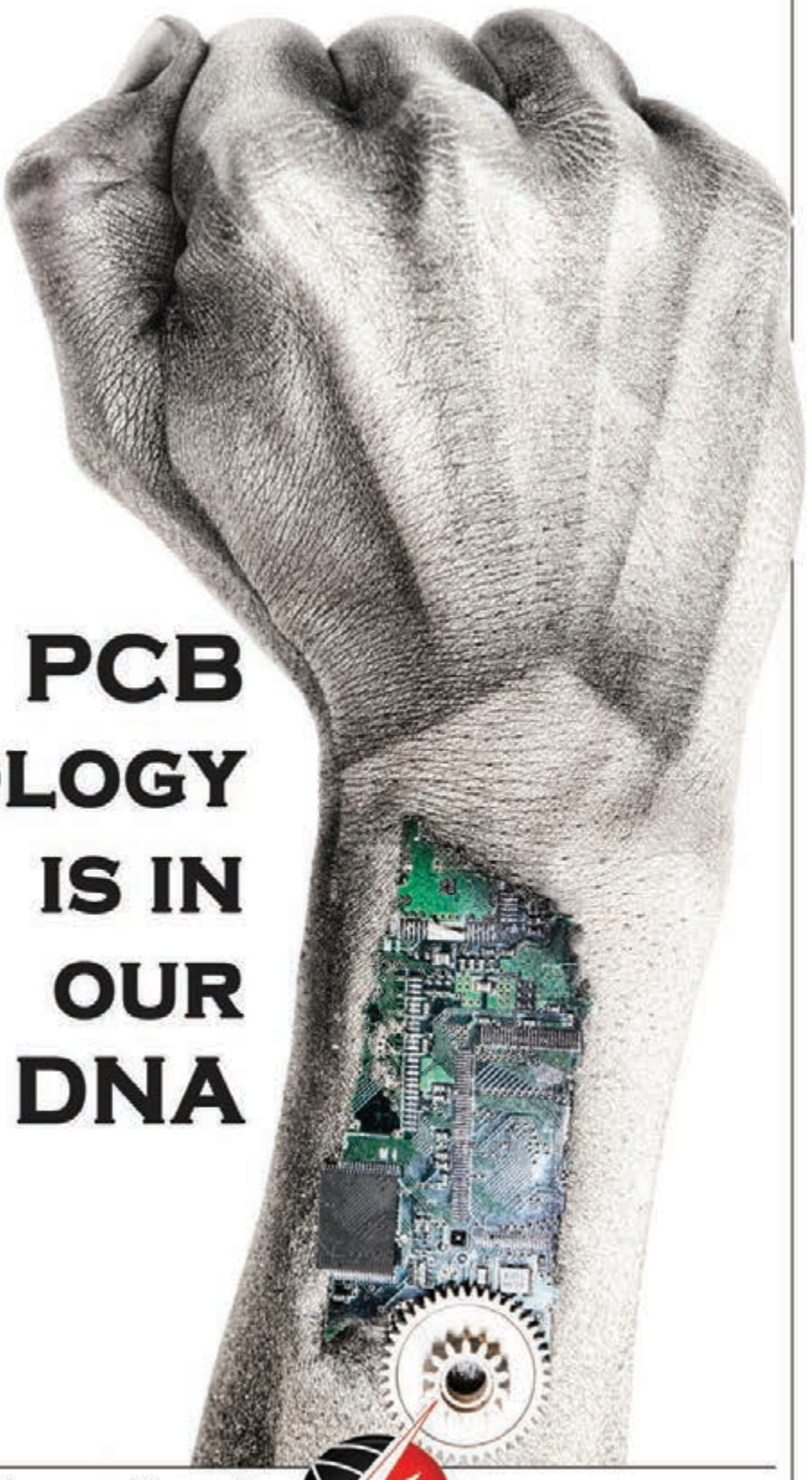


Photo 1a—The assembled time-broadcasting device has the microcontroller board over the GPS board. A four-pin connector enables me to attach it to the network. **b—**The circuit board on the left is attached to the GPS module with wire wrap jumpers. Because they were left off my original circuit, the two protection Zener diodes protruding from the bottom of the circuit board are visible.



PCB TECHNOLOGY IS IN OUR DNA

For 26 years Imagineering has been providing multi layer, rigid and flex PCB's with some of the most stringent quality standards in the industry.

With HDI capabilities like 3mil laser drilling plus Blind & Buried Stacked & Staggered Sequential Vias, you can clearly see High Tech PCB's are in our DNA.

www.PCBnet.com

847-806-0003 sales@PCBnet.com

ITAR, ISO 9001:2008, UL Approved

PCB CONTRACT MANUFACTURING SINCE 1986

```
$GPGGA,,,,,,0,0,,,,,,,*56
…
$GPGSA,A,3,02,29,30,15,05,10,12,26,24,21,,,2.05,0.94,1.82*0B
$GPGSV,3,1,10,10,69,048,36,24,48,301,46,29,48,320,43,30,41,254,41*76
$GPGSV,3,2,10,02,37,060,45,05,34,222,39,12,33,206,33,26,23,144,37*75
$GPGSV,3,3,10,15,19,161,31,21,11,274,40,,,,,,,,*76
$GPRMC,213736,A,3240.1630,N,09706.5185,W,0.000,0.0,131208,5.0,E*6A
$PRWIZCH,02,7,29,7,30,7,15,7,05,7,10,7,00,0,12,7,00,0,26,7,24,7,21,7*45
$GPGSA,A,3,02,29,30,15,05,10,12,26,24,21,,,2.05,0.94,1.82
$GPGGA,171545.0,3240.141,N,09706.505,W,1,3,002.2,,M,-024,M,,*5C
```

Figure 1—The GPS receiver creates several messages. The top line shows when the unit has not yet found enough satellites; the bottom line shows when it has. The latitude and longitude values show the exact location of my workbench inside my house. Each line's last value is a checksum, which I do not use in this software.

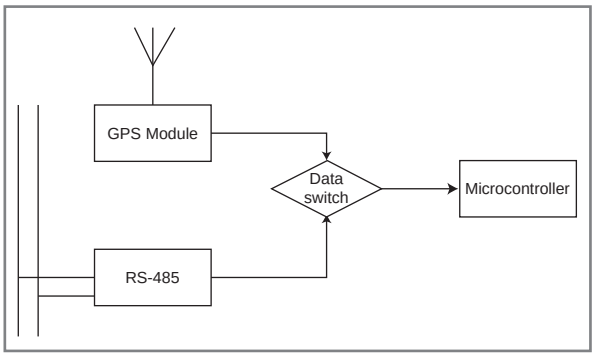


Figure 2—The time-broadcasting device switches between receiving time from the GPS and broadcasting time on the RS-485. Because the unit has an internal clock driven by Timer 1, it does not need to receive the GPS time very often.

both the time and the receiver's fixed position at that instant. All GPS messages are sent using the National Marine Electronics Association (NMEA) protocol. The full standard is available for purchase from the NMEA website. However, the protocol's common messages are available almost anywhere online. The basic data rate is 4,800 bps, no parity, 8 bits, 1 stop bit (4,800 8 N 1). As for the contents, I consulted Wikipedia and obtained the following information. Each message begins with a dollar sign ($) and ends with a carriage return. Following the dollar sign is a five-letter code that usually begins with GP, the remaining three letters identify the message type. After the five-letter code is a string of ASCII letters or numbers that are comma separated.[1] Figure 1 is an example of a datastream from my GPS unit. The messages contain the number of satellites detected, each satellite's signal strength, the time, the position, and many other pieces of information.

The only message I am interested in is the Time/Fix message, which has the $GPGGA prefix. But, I'm only interested in its time portion, because I'm fairly confident my house's position won't move anytime soon. (We were in a tornado once, but that's a different story.) The message's time portion is the first six digits after the prefix. In the last line of Figure 1, the time is 17:15:45, or 5:15:45 P.M. For my accuracy, I am not interested in fractional seconds either, so this seems perfect.

BUILDING THE HARDWARE

Figure 2 shows the time-broadcasting device's basic block diagram. A single microcontroller uses a data selector switch to choose whether the UART receives input from a GPS receiver or from the RS-485 network. Because I use a 32,768-Hz quartz crystal on the microcontroller's Timer 1, the unit can keep track of its own time and only switch over to the GPS receiver once an hour or so. It was a simpler decision to use an old data selector line to switch over the serial input instead of using a microcontroller with two UARTs or bit banging the 4,800-bps signal in software. I dug up an old 75157 data selector chip—another score for using leftover parts.

Figure 3 shows the layout, which was created using the ExpressPCB software suite. Because I only needed a single unit, the actual board was made by hand.

The unit plugs into the RS-485 cable through J1 and is wired in the same way as the devices in my previous article with power on pins 1 and 2, and the A-B differential signal on pins 3 and 4. The differential driver is a DS-1487 integrated circuit (IC) designed to convert TTL signals to RS-485 signals The input to the differential driver U2 is protected by Zener diodes, where it is converted to a TTL level and fed into data input selector U3, which is used to toggle the input between the RS-485 line or the GPS receiver. From the data selector, the serial signal makes its way to the microcontroller, U1. The microcontroller is a simple Microchip Technology PIC16F688, which uses only a single UART and a 16-bit timer on TMR1. The serial signal is fed to the UART and a 32,786-Hz crystal is attached to Timer 1 to maintain the clock.

When the unit wishes to obtain the

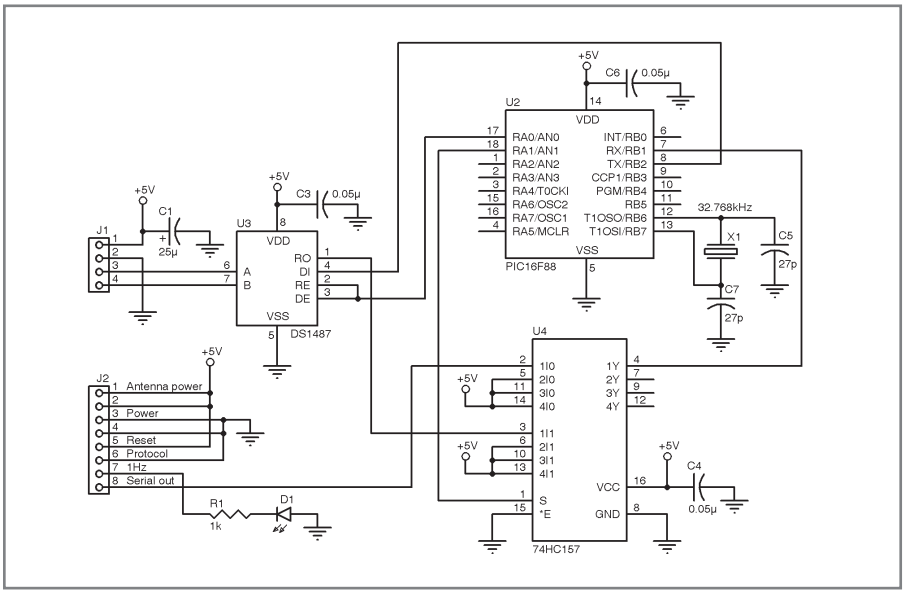


Figure 3—The time-broadcasting device's layout was created using the ExpressPCB software suite, but I made the actual board by hand since I only needed a single unit.



WE BREAK DOWN BARRIERS

We know time is money, so we developed an instant online quote engine to reduce time to market exponentially.

Join the Revolution Happening NOW @ **www.PCBnet.com**

847-806-0003 sales@PCBnet.com

ITAR, ISO 9001:2008, UL Approved

PCB CONTRACT MANUFACTURING SINCE 1986



Listing 1—This is the interrupt routine that attempts to locate the Time/Fix message in the GPS datastream. Once the message is located, it will extract the seconds, minutes, and hours from the message.

```
/*****************************************************
*       ProcessTime() - Procedure to digest the time
*       message from the GPS rx.  We collect the first
*       13 characters after the $ and see if it starts
*       with GPGGA, which is the time/fix message
*
*****************************************************/
void ProcessTime() {
        /* If we haven't started a message, see if we can start */
        if (bMsgStarted == 0) {
                /* if it's '$'  then it's a start */
                if (ch == '$') bMsgStarted = 1;
                /* Set the pointers and bail. (we don't keep the $) */
                rxcount = 0;
                return;
        }
        /*  add it to the RX buffer */
        cRXbuffer[rxcount] = ch;
        rxcount++;
        /* if this is a newline, then end any msg we are collecting */
        if (ch == 0x0d) {
                rxcount = 0;
                bMsgStarted = 0;
                return;
        }

        if (rxcount < 13) {      /* if we didn't reach the msg len, bail */
                return;
        }
        /* This is the max we want from a message, so
                reset the indicators */
        rxcount = 0;
        bMsgStarted = 0;
        /* Is it the TIME/FIX message */
        if (memcmp(cRXbuffer,timemsg,6) != 0) {
                return;
        }
        /* if there is no time values in the message, skip */
        if (cRXbuffer[6] == ',') {
                return;
        }

        /* This means it's the time msg */
        cRXbuffer[12] = 0;                          /* get the seconds */
        sec = atoi(&cRXbuffer[10]);
        cRXbuffer[10] = 0;                          /* get the minutes */
        min = atoi(&cRXbuffer[8]);
        cRXbuffer[8] = 0;                           /* get the hour */
        hour = atoi(&cRXbuffer[6]);
        requestedfeed = RS485_FEED;
        hour -= gmtOffset;
        if (hour & 0x80) hour += 24;
        bTimeFromGPS = 1;
        bTimeFromCmd = 0;
        return;
}
```

data feed from the GPS receiver, it toggles the data input selector to select the alternate input. Because I used the GPS's default data protocol, there is no reason to instruct the GPS to enter any other configuration. As a result, there is no TX line going to the GPS. Instead, the TX line can remain constantly connected to the differential driver. The connections to the GPS module are made through J2. They provide just enough connections to get the module to power up. They also provide a single 1-Hz LED display to indicate that it's working.

For the GPS receiver, I had an old

FALLOUT RATE

REDUCE YOUR FALLOUT RATE



INCREASE YOUR YIELD

With full DFM check and BOM review, we can guarantee that your fallout will be negligable

When you have a one stop solution for all your PCB needs, you can rest assured that your project will be taken from concept to realization without any issues
We do free DFM check, BOM and Assembly review on all orders placed with us.

www.PCBnet.com

847-806-0003 sales@PCBnet.com

ITAR, ISO 9001:2008, UL Approved

PCB CONTRACT MANUFACTURING SINCE 1986

Command	Payload	Purpose
G	none	Request an immediate switch to obtain GPS time
D	GMT offset	Set the GMT offset, which I use to correct for Daylight Saving Time

Table 1—The time-broadcasting device understands a few extra messages, in addition to the common messages understood by all devices on the RS-485 line.

Navman Jupiter TU30-D400 series receiver I purchased from eBay many years ago. It was one of those purchases I made because I wanted a unit to learn how it worked. It ended up in my parts box once I was done with it. I searched for the datasheet and found the WA5RRN HAM radio website (the Resources section includes a link to the PDF). Thank you to WA5RRN, whoever you are.

GETTING THE TIME

The time server was built around a PIC16F688 microcontroller. I used the CCS PCW compiler to write the program in C. This was my first project working with the CCS compiler, which has a large collection of built-in functions to work with I^2C, I/O pins, UARTs, timers, and so forth.

The program begins by setting up the watchdog timer, determining the reason for the reset, and recording it to the EEPROM so the unit's stability can be examined. Then the UART is configured for communication on the RS-485 line, which runs at 19,200 bps. Lastly, the I/O pins are configured to enable control over the differential driver and data selector. Timer 1 is set to use the external crystal. Just before entering the infinite loop, it enables the interrupts. Events (e.g., characters in the UART or the Timer 1 expiring) will generate an interrupt service request (ISR).

During the infinite loop, the program tests to see if the minute value has changed. This can happen due to the internal clock, or as a result of a new (corrected) time message from the GPS receiver. The program is set up to broadcast the time to the special "universal" address every 20 min. at 0, 20, and 40 min. past the hour. However, it will only do this if it has received a time message from the GPS, or if someone else has manually set the internal clock through a message on the RS-485 network. This prevents the time server from broadcasting a message when it hasn't yet received accurate time after a reset.

At specific intervals, the microcontroller instructs the data selector to change from the RS-485 line to the GPS line so it can obtain an accurate time fix. The frequency with which this occurs depends on whether or not the unit has already received a valid time from the GPS unit. Initially, it will switch to the GPS and listen for a Time/Fix message every 10 min. The unit will listen for 10 s or until it receives a Time/Fix message with a valid time, whichever comes first. Once it obtains a GPS time, it will slow down to 30 min. past every hour. If it fails to receive a Time/Fix message again, it will reenter the more frequent pattern of attempting a Time/Fix message every 10 min.

Changing the UART source from the RS-485 network to the GPS doesn't just mean changing the data selector. Because the protocols are different, the UART must be turned off, reset, and reactivated. The CCS compiler's built-in functions simplify that task, so all that needs to be done is to ensure it is not busy processing a message from the RS-485 line. A weakness is that it will only attempt once, and if the line is in use, the program waits until the next time event. It does not keep trying until the line is clear. However, in this design, communication traffic is so low that a collision is rare. I could easily correct this, but I haven't taken the time to disassemble the unit to reprogram the microcontroller. Once the line has been switched to the GPS data feed, an expiration time is set to 10 s from the current time. If there is no good Time/Fix message before the time expires, it forces a switch back to the RS-485 line anyway. This places a limit on the amount of time the unit remains off the RS-485 network looking for the time value from the GPS. If a clean Time/Fix message is found before the expiration, the data selector is immediately switched back.

INTERRUPTS

When a UART interrupt occurs, the program first determines if the input data selector is feeding from the RS-485 line or from the GPS receiver. Listing 1 shows the code that processes the GPS receiver's message. If the source is from the GPS receiver, it begins watching the UART bytes one by one until is sees the `$`, which indicates the beginning of a message. Then it collects the bytes into the receive buffer until it reaches 13 characters, which is enough to extract the desired information. The time message is a string that contains the `$GPGGA` message header followed by the time, latitude, longitude, number of satellites, and so forth. The first 13 bytes are more than enough to obtain the time information, as shown in Figure 1. The simplest way to determine if there is a time value in the message is to check the sixth position for the comma, which is where the hour value should be. If the comma is there, the unit doesn't yet have a satellite fix. If the comma is not there, the seconds, minutes, and hours can be extracted from the message, and the microcontroller clock can be synced to that time. Once a correct time message has been found, the unit will immediately switch back to the RS-485 network.

Alternatively, if the message is from the RS-485 line, the ISR simply watches for the start of header (SOH) and end

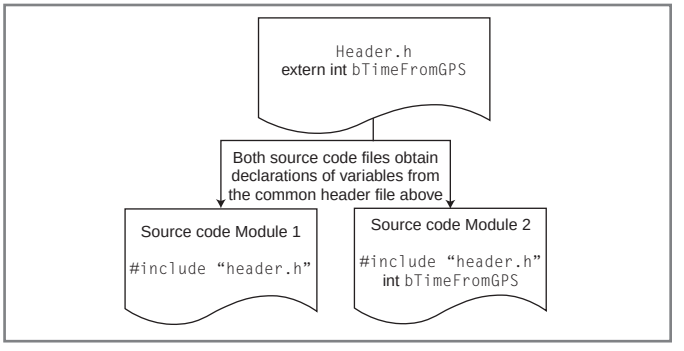


Figure 4—The actual memory byte for `bTimeFromGPS` is assigned in source code Module 2. The `#include` header file for both source modules tells source code Module 1 there is a variable somewhere named `bTimeFromGPS` so it doesn't create its own memory for it. All this gets collected by the "Link" step when the program is built.

Our Services and Capabilities

Process		2008	2009	2010	2011	2012
Material		FR 4	FR 4	FR 4	FR 4	FR 4 Metal Core
		PTFE	PTFE	PTFE	PTFE	PTFE
		Polyimid	Polyimide	Polyimide	Polyimide	Polyimide
		PI	PI	PI	PI	PI
Min. Dielectric Thickness		2.5 mil	2 mil	2 mil	2 mil	2 mil
Max. Layer Count		36	40	42	42	42
Max. Working Panel Size		20"x26"	20"x30"	21"x31"	21"x31"	21"x31"
Max. Board Thickness		.250"	.250"	.250"	.280"	.280"
Min. Board Thickness		.017"(6L)	.016"(6L)	.016"(6L)	.016"(6L)	.016"(6L)
Min. Line/Space	I/L	3/3 Mil	3/2.5 Mil	2.5/2.5 Mil	2/2 Mil	2/1.5 Mil
	O/L	3/3 Mil	3/2.5 Mil	2.5/2.5 Mil	2.5/2 Mil	2/2 Mil
Warp		.0015"/in.	.001"/in.	.001"/in.	.001"/in.	.001"/in.
BGA Pitch		0.40mm	0.35mm	0.30mm	0.25mm	0.2mm
Layer to Layer Registration		5 mil	4 mil	3 mil	3 mil	3 mil
Finish Hole Size (min.)	Mechanical Drill	.006"	.004"	.004"	.004"	.004"
	Laser Drill	.005"	.004"	.003"	.003"	.0025"
True Hole Position		+/-.002"	+/-.002"	+/-.002"	+/-.002"	+/-.0015"
Finish Hole Size Tolerance	PTH	+/-.003"	+/-.002"	+/-.002"	+/-.002"	+/-.0015"
	Non-PTH	+/-.001"	+/-.001"	+/-.001"	+/-.001"	+/-.001"
Aspect Ratio (Board Tks /FHS)		16	20	20	20	30
Heavy Copper	Inner Layer	4 OZ	5 OZ	5 OZ	6 OZ	6 OZ
	Outer Layer	6 OZ	7 OZ	7 OZ	8 OZ	12 OZ
Buried/Blind Via			Yes 3+N+3	Yes 4+N+4	Yes 9+N+9	Yes 9+N+9
Plasma Desmear			Yes	Yes	Yes	Yes
Outline Tolerance			+/-.004"	+/-.004"	+/-.004"	+/-.004"
Surface Finish			HASL	HASL	HASL	HASL
			ENIG	ENIG	ENIG	ENIG
			Immersion Silver	Immersion Silver	Immersion Silver	Immersion Silver
			OSP(ENTEK)	OSP(ENTEK)	OSP(ENTEK)	OSP(ENTEK)
			Carbon	Carbon	Carbon	Carbon
			Immersion Tin	Immersion Tin	Immersion Tin	Immersion Tin
			Electrolytic Gold	Electrolytic Gold	Electrolytic Gold	Electrolytic Gold
Impedance control			+/-5%	+/-5%	+/-5%	+/-5%
RoHS Compliant			Yes	Yes	Yes	Yes

PCBA CAPABILITIES

- 24 Hr. Turnaround Service
- Passives down to 01005
- BGA and MBGA
- Leaded or Pb-Free dedicated lines
- Stencil-less Assembly - Jet Printing
- SMT and Thru-Hole capabilities
- All SMT Machine placed
- Full Turn-key

EQUIPMENT LIST

- My500 Jet Printer
  Eliminates use of stencils
- My100 Pick and Place Machine
  Hydra-Head Technology
  Passive value checks before part placement
  Cut tape and loose parts can be machine placed
- Vitronics Reflow Oven
  8-Zone Reflow oven

www.PCBnet.com
847-806-0003 sales@PCBnet.com ITAR, ISO 9001:2008, UL Approved

of file (EOF) marker bytes and turns on a received message flag value to tell the main loop a message arrived. The main loop then processes the messages. In addition to the common messages all devices on the RS-485 line understand, there are a few extra messages this time-broadcasting device comprehends (see Table 1).

The other interrupt is Timer 1, which simply fires every time the 16-bit (65536) counter overflows, incrementing the seconds, as well as the minutes and hours, if needed. The timer interrupt routine adds 32,768 to the timer value by setting the high bit in the timer. This makes the interrupt timer fire every second rather than every 2 s.

CCS COMPILER ISSUE

Three source code modules make up the program (see Figure 4). I discovered a problem with my version of the CCS compiler. I declared `bTimeFromGPS`, a single variable that is the old Hungarian source code notation (b means Boolean and TimeFromGPS means received the time from the GPS). This variable was used in two source modules. One module would set it to signal a message was received, so the other modules would test it to realize the message arrived. In a typical C program, when the same variable is used in multiple source modules, the single variable is declared in one module, and a header file is used to tell others about its existence. Figure 4 represents how the header file is referenced by two source code modules.

The problem I experienced with the CCS compiler is that it didn't realize that the header file's value was only an "external" reference. Instead, it made two variables with the same name. This resulted in one module signaling that a message was received, with the other module never knowing it, because it was testing its own copy. I had to use the symbol (SYM) file option that generates a list of the symbols and their addresses

Listing 2—The CCS compiler created two variables of the same name and placed them at two different memory locations. This made it difficult to debug the code, because one module never received the message that the other module had received a time record from the GPS.

```
026      @INTERRUPT_AREA
027      @INTERRUPT_AREA
028      @INTERRUPT_AREA
029      hour
02A      min
02B      sec
02C      sMin
02D      sHour
02E      gmtOffset
02F-05E  cRXbuffer
Ø3B      bTimeFromGPS
05F.0    bMsgRx
05F.1    bTimeFromCmd
05F.2    bMsgStarted
05F.3    bMsgTx
05F.4    requestedfeed
05F.5    feedsource
060-061  strtok.save
062-065  _Randseed
Ø66      bTimeFromGPS
```






to figure this out. The SYM file had two entries in it, one for each copy of the variable, each at a different memory location. Listing 2 shows the symbol file representing how it created two variables of the same name (`bTimeFromGPS`) and assigned them different memory locations. The solution was to move the "extern" declaration from the header and place it in the source code modules.

AUTOMATING ACCURATE TIME

After building and installing some light controllers, I wanted an automated way for them to keep accurate time without needing to constantly adjust their clocks for drift or from a power outage. Using the RS-485 network, I wired in my house enables me to let every device know what the current time is—at least those devices that need to know.

Leftover GPS receivers have become pervasive in the surplus markets. I hope this gives readers an introduction to decoding the datastream produced by GPS receivers. I used this to create a basic time-broadcasting device that was a nice project enabling me use a bunch of leftover components I had lying around. My design broadcasts a local time value, corrected for offsets from GMT by a value stored in EEPROM. I also used this opportunity to exercise my CCS complier, which exposed some behaviors that are not issues in the regular C/C++ compilers in the PC world.

After working in software development for 20 years, Scott Weber (scotty42@csweber.com) is tired of the direction the PC software world is taking. He is now striving to complete his Electrical Engineering degree at the University of Texas Arlington. He lives in Texas with his beautiful wife and her garden.

REFERENCE

[1] Wikipedia, "NMEA 0183," 2012, http://en.wikipedia.org/wiki/NMEA_0183.

RESOURCES

ExpressPCB, www.expresspcb.com.

The National Marine Electronics Association (NMEA), www.nmea.org.

Navman, "Jupiter GPS Receiver: TU30-D400 Series Data Sheet," Revision A, www.wa5rrn.com/GPS%20Other/Jupiter/NAVMAN_Jupiter_11_Datasheet.pdf.

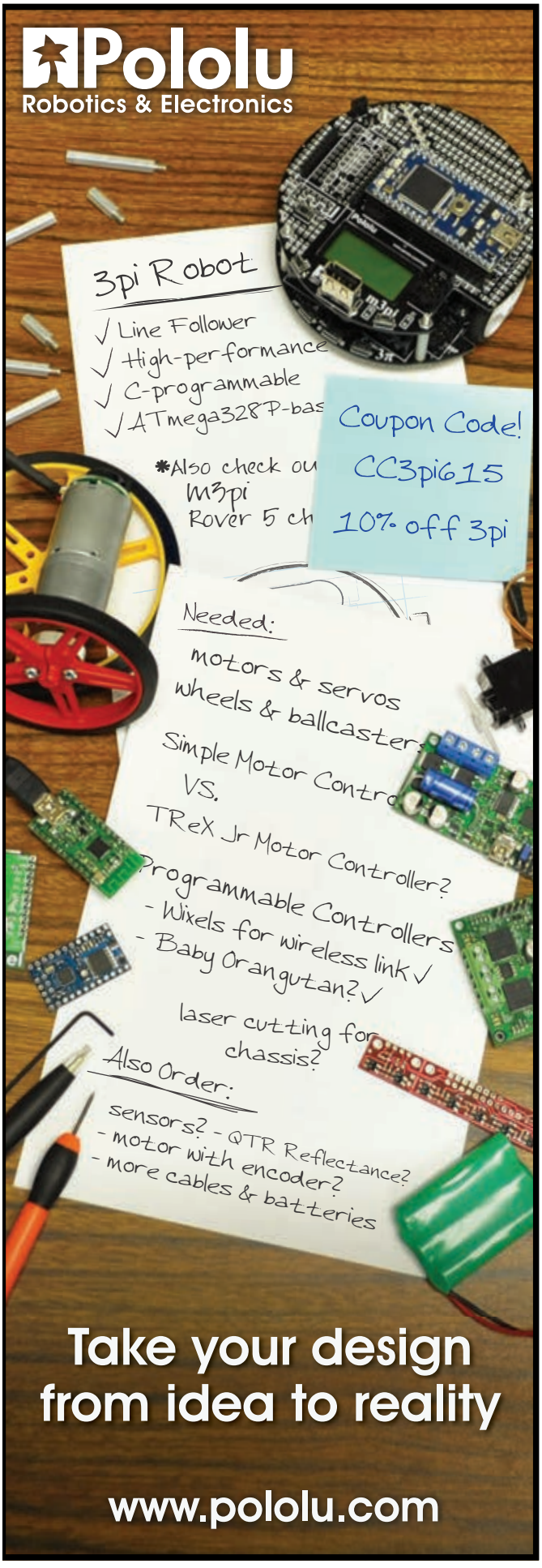
S. Weber, "MCU-Based Light Control: Longer Serial Communication on Differential Wires," *Circuit Cellar* 265, 2012.

SOURCES

PCW Compiler
CCS, Inc. | www.ccsinfo.com

PIC16F688 Microcontroller
Microchip Technology, Inc. | www.microchip.com

Jupiter TU30-D400 receiver
Navman | www.navman.com



by Thiadmer Riemersma (The Netherlands)

DIY Automated Component Dispenser

Would you like to build an automated component storage and dispensing system for your workbench? The design detailed in this article will be a welcome addition to any serious designer's workspace. It includes small bins for surface-mount components and custom PC software that simplifies the task of hand-assembling PCBs.

The "component carousel" is a rotating dispenser (and storage container) for loose components (see Photo 1). For prototypes and low production runs, PCBs are often populated by hand. During hand assembly, it is helpful to use a dispenser to pick up the next component. The carousel is driven by a controller. It operates in an automated system that shows where to place components and rotates the carousel to the appropriate bin. This project is interesting because it's part mechanical, part electronics, and part software.

THE MECHANICS

The carousel's mechanical construction is simple. It is basically a spindle with an RC-servo driven disk (see Photo 1c). The components' bins are potting boxes that are glued on the disk. Potting boxes are normally only available in black. Mine are reddish-brown on the inside. I sprayed them with copper-based conductive paint so the bins can safely store ESD-sensitive components.

The number of bins a disk carries depends on the bin size and the disk diameter. For my space-constrained work table, I prefer to use relatively small disks with roughly 40 bins. This is a small number, even for moderate projects. To compensate for this limitation, the disks are replaceable. The servo does not directly drive the disk with the bins. Rather, it drives a "base disk." The disks with bins sit on top of this base disk and use a pin-hole construction to ensure it does not slip. The base disk must be about 15 mm smaller than the smallest disk, so a reflective optical sensor can detect a disk's presence without being hindered by the base disk.

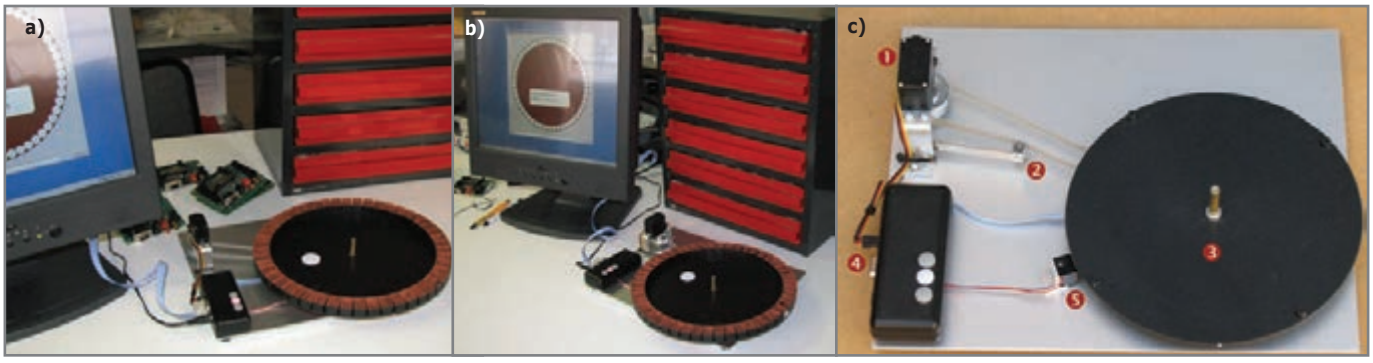


Photo 1a—The assembled carousel is connected to an all-in-one PC. **b—**Here is a view of the PC utility. **c—**These are the carousel's components: the carousel with the disk removed, showing the servo (1), the transmission belt with a tension wheel made from a rather large microswitch (2), the base disk (3), the control unit (4), and the reflective optical sensor (5). The RFID reader, which is not visible, is just below the base disk's edge between the optical sensor and the transmission belt.

SHEET METAL BENDING

The carousel's mechanical construction features a spindle, which is a 5-mm machine screw (see Figure 1). Over the spindle, I slid (and glued) a brass tube with a 5-mm inside diameter and a 6-mm outside diameter. The hole in the disk's toothed wheel is slightly larger than 6 mm. The only parts that require preparatory work and calculations are the Z-shaped brackets holding the servo.

The bracket's width (represented as W in Figure 1) should be approximately 30 mm so each leg is approximately 14 mm, assuming the strip is 2-mm thick. The height must be chosen so the servo's toothed wheel is approximately 4 mm off the base plate, so it is the total height from the servo's shoulder (where it is fastened to the brackets) to the outer edge of the toothed wheel, plus 4 mm. For my design, the height needed to be 34 mm.

The strip from which the bracket is bent must be shorter than the width plus the height. The bends in the material literally

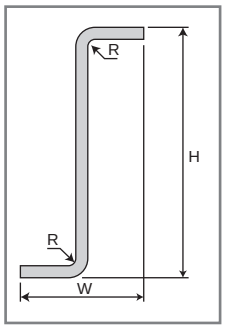


Figure 1—Part of the carousel's mechanical construction includes two brackets, which are bent from an aluminum strip. To calculate the flat length given a width, a height, and a bending radius, see the equation below.

"cut a corner." The amount by which the strip length must be reduced to achieve the desired width and height is called the bend deduction. In my design, because there are two bends in the strip, I have to subtract the bend deduction twice. The bend deduction's value depends on a number of factors, including the angle and the material's mechanical proportions. For a 90° angle, the bend deduction is:

$$2\,(T+R)\;-\;\frac{\pi}{2}\,(R+K\times T)$$

T is the thickness of the strip and K is a material-dependent factor, which is close to 0.5 for aluminum. As a rule of thumb, the bend radius (R) should be at least the same as the material's thickness (i.e., 2 mm for a 2-mm thick strip). With sharper bends, you risk creating hair cracks in the material, which significantly weakens it.

RC SERVO

I chose an RC servo to drive the spindle because it is an absolute positioning actuator in a self-contained package (see Figure 2). They are inexpensive and easy to control. A typical servo is limited to a 180° rotation, so a toothed-belt transmission scales this to a full rotation (many distributors refer to a toothed belt as a "timing belt" due to its common use of synchronizing valve actions in a combustion engine). The servo's pulley has 40 teeth. The base disk's pulley has 18 teeth for a 1:2.2 transmission.

An RC servo is controlled with PWM using a 50-Hz frequency and a 0.5-to-2.5-ms pulse width for minimum and maximum position. It uses a 1.5-ms pulse width for the neutral position.

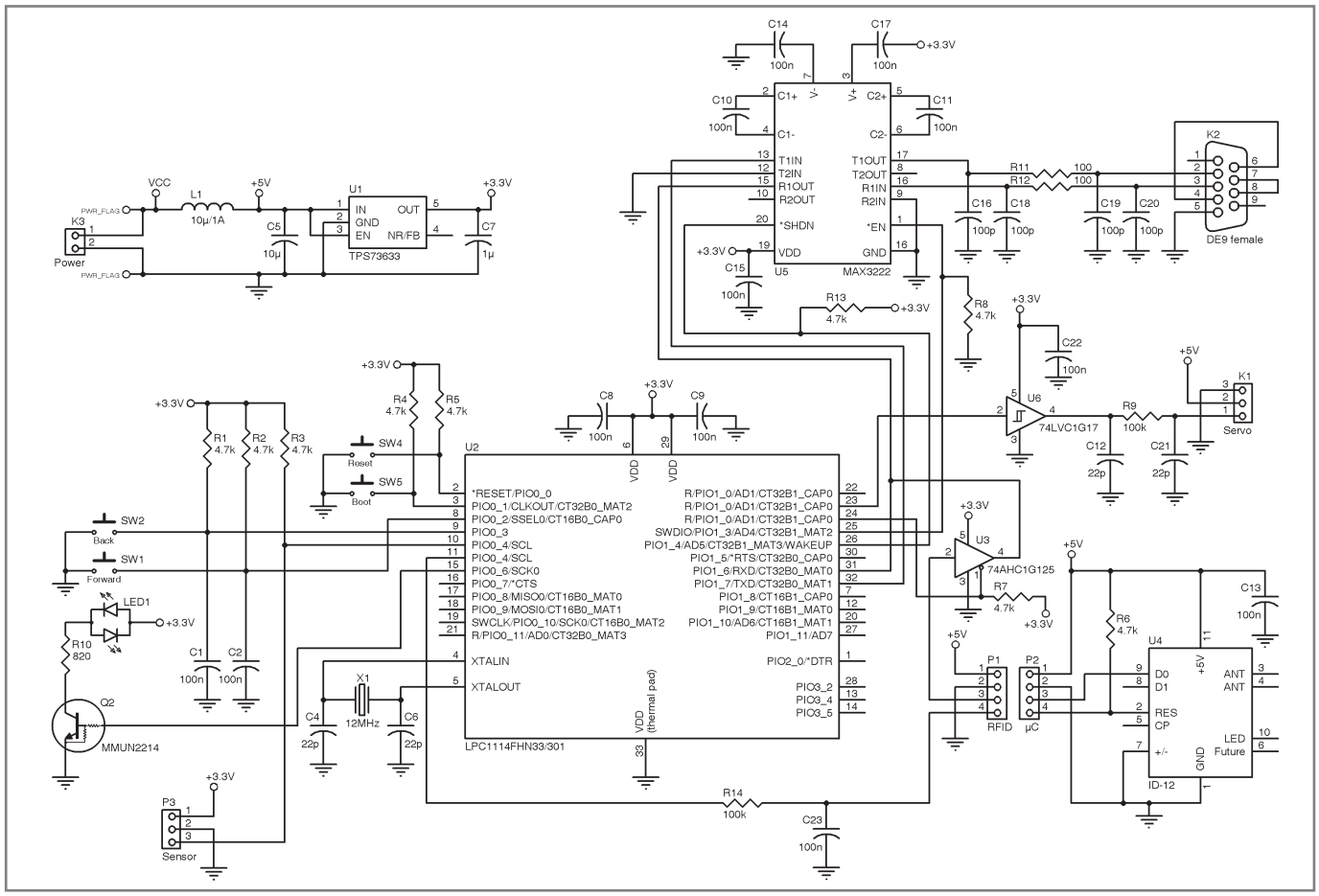


Figure 2—The controller's schematic, which drives the servo, reads the RFID tag and communicates with PC software.

There is variation between servos in this area as well. I used a Hitec HS-5485HB servo, which accepts pulses in a 0.75-to-2.25-ms range. There are two servo classes: analog and digital. Digital servos are more accurate in positioning, so they are the preferred choice. They do have one caveat: a digital servo may not support 180° rotation out of the box.

Servos—especially digital servos—rotate pretty quickly. If the speed is not controlled, a turn from one bin to another from the other side of the disk may spray the tiny 0603s out of the bins. While it's important to limit the rotation speed, it is also important to limit the movement's acceleration. There is one caution: right after power on, the controller cannot know the pulse width at the start because it does not know the servo's beginning position. So, if the servo is not in the position the controller assumes when it starts sending the pulse train, the servo will turn to that position at top speed (and top acceleration). There are several ways to avoid this. I prefer to make the controller assume the neutral position at power on, and to make the carousel move to neutral during various events (e.g., when the controller detects that a disk has been removed).

MAPPING & RFID

A PC utility facilitates the mapping from a component to a carousel bin. For a new disk, fill in the disk layout and its contents in a list. At the very least, the "contents" include the component's value. They typically also include the component's package and other distinctive attributes. After defining a disk, type a value into an edit field, along with any other optional attributes, so the utility can send a command to the carousel to turn the right bin to the pickup location. The carousel communicates with the PC via an RS-232 serial line. Photo 2 is a screenshot of the PC utility.

Since the disks are replaceable, the mapping from the component to the correct bin only works if the carousel disk is the one the PC utility thinks it is. Therefore, I added an RFID reader to the carousel and attached a passive tag to every disk. The white dot on the disk

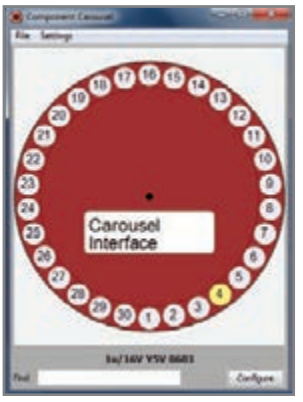


Photo 2— This is a screen shot of the carousel's PC utility.

in Photo 1 is the RFID tag. The carousel transmits a unique ID it has scanned from the RFID tag to the PC utility. The utility can then load the appropriate disk definition.

The RFID reader will only reliably read the tag when it is directly above the reader. Since the tag is glued onto the disk and rotates along with it (the reader stays fixed at its position), there is only a relatively small angular range where the tag is sufficiently well aligned above the reader (i.e., to successfully read a tag, the disk must first be rotated to the appropriate position). Not surprisingly, neutral is the most appropriate position. When a disk removal is detected (or after powerup), the carousel moves to the neutral position, so a new disk is already clipped to the base disk in the neutral position. The RFID tag is only scanned after initially detecting a disk. There is no need to rescan it until a disk removal is detected.

Various "serial servo controllers" exist that drive a servo through serial commands. The PC utility supports a few. Yet, for the extras (e.g., an RFID tag reader capability and a "disk present" optical sensor), I decided to design my own controller. The controller is based on the NXP LPC1100 series and an ARM Cortex M0 core microcontroller. The smallest controller in this series is already suitable. All that is needed is a PWM unit and a UART. Actually, there are two interfaces requiring a UART: the communication between the controller and the PC and the interface to the RFID reader. However, the communication with the RFID reader is only needed immediately after detecting a new disk and, at that moment, communication with the PC is not relevant. So, the LPC1100's single UART is toggled between the two interfaces.

It may appear considerably limiting that the carousel requires a PC to control it, as opposed to it being a self-contained unit, but it isn't, really. The carousel is intended for a workplace setup that already includes a PC. The activities for populating a PCB with components include: picking up each component from its container, identifying the spot to place each component on the PCB, and carefully pressing the component in the solder paste. The carousel makes the pick up efficient. To locate a PCB's component, I use CompuPhase's freely available VisualPlace program. Better yet, there is a link between the carousel software and VisualPlace (by way of a plug-in for VisualPlace). When I click on a VisualPlace component, that part's location is marked on the screen and the carousel turns the bin with that component to the pick-up location. Combined, the carousel and VisualPlace make a PCB's manual assembly more efficient—and appreciably so, in my experience.

Although I have a PC on the workbench, I avoid having a keyboard on it. A keyboard takes up too much space, considering that during assembly I basically need a single key: Next. This function is also handled by the serial servo controller. The two servo controller buttons have their own function, but they can also be forwarded to the PC utility (via the RS-232 line) and then forwarded to VisualPlace. You can also use a three-pin header to connect a pair of external switches (e.g., foot switches) to the controller. As an aside, this article's photos show a prototype of the controller that has three buttons. One of the buttons has the same functionality as the optical sensor, so I removed it from the final design.

The optical sensor sits below the disk, pointing upward. As you can see in Photo 1, the prototype's disk is made of black plastic with a matte finish. This is the worst possible choice for a reflective infrared sensor. (As you may imagine, the idea of using an optical sensor and scrapping one of the buttons came as an afterthought.

Much of the prototype had already been built.) I added a red ring to the disk's bottom so the sensor can actually detect the disk. For my next disks, I will save myself some work and buy the plastic sheet for the disks in a color other than black matte.

FIRMWARE

I used the GNU GCC toolchain and LPC1100 header/interface files downloaded from the NXP website to write the controller's firmware in C. The firmware is relatively simple. It sets up a timer channel for PWM and the UART for communication and creates a millisecond timer as a time base. Servo rotation speed is limited by gradually changing the pulse width from its current "duty cycle" to the target duty cycle. The step size by which the duty cycle changes, combined with the frequency that this step size is added to the duty cycle, determines the rotation speed. Acceleration is limited by gradually increasing the step size. You also need to determine where the rotation should start to decelerate. Some complexity in the inner operation loop is also devoted to keeping movement smooth when a new move is started before the active one finishes. If the direction inverses, a slow out must be forced before the controller starts to rotate in the other direction. The PC utility is compiled in C++ using wxWidgets and runs in Microsoft Windows and Linux. It has a simple user interface that enables you to select a bin or search for a component. The utility creates a listening socket for communication with other applications that remains idle until it is connected by another application.

POSSIBLE MODIFICATIONS

There are many ways to improve the carousel's current design. Instead of a servo, a stepper motor would enable continuous rotation and increase positioning accuracy, provided the "home" position can also be accurately detected by an additional sensor. The component bins are all along the disk's outer edge and the disk's middle section is entirely unused, wasting space. To utilize that space, you could stack two or three disks with decreasing sizes (e.g., a "bottom disk" with 40 bins and a smaller "top disk" with 20 bins) to provide a respectable 60 bins without taking up a lot of space. A type of indicator telling you from which level to choose a component (top or bottom) must also be added. No disk will be large enough to hold all the components for all of your projects, which is why I suggest having several disks (e.g., one per project). As an alternative, have a large drawer cabinet with bins for all the components so you can quickly assemble a project disk by taking the appropriate bins out of the cabinet and clipping them on the disk (i.e., have a disk with detachable bins). This requires a custom design for the bins and the disk, which I did not pursue. But, with modern fabrication techniques (e.g., 3-D printing), it may be feasible and inexpensive.

Thiadmer Riemersma (thiadmer@compuphase.com) develops embedded software for the products produced by his Netherlands-based company, CompuPhase.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/268.

RESOURCES

Antratek Electronics www.antratek.nl.

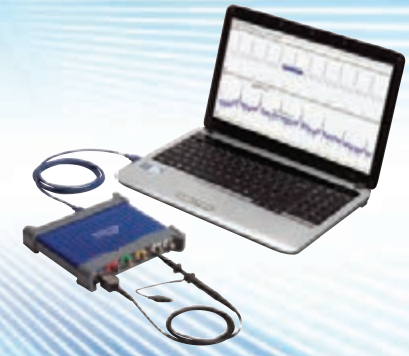
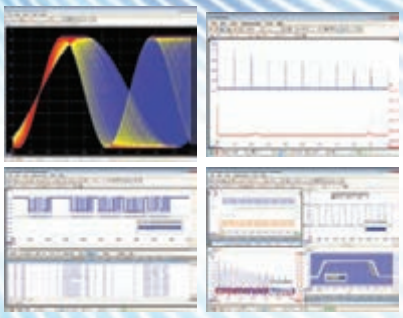
ID Innovations, www.id-innovations.com.

Parallax, Inc., www.parallax.com

SOURCES

VisualPlace Software
CompuPhase | www.compuphase.com

HS-5485HB Servo
Hitec RCD USA, Inc. |www.hitecrcd.com

NEW
pico Technology
PicoScope®
4-CHANNEL
3000 Series
200 MHz bandwidth
128 MS deep memory
1 GS/s real-time sampling
Advanced digital triggering
200 MHz spectrum analyzer
Function generator or AWG


ALL MODELS INCLUDE PROBES, FULL SOFTWARE AND 5 YEAR WARRANTY. SOFTWARE INCLUDES. MEASUREMENTS, SPECTRUM ANALYZER, FULL SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD. FREE UPDATES.
FOR THE FULL PRODUCT RANGE VISIT
www.picotech.com/pco486
or Call: 1-800-591-2796

by Jan Axelson (USA)

Debugging USB Firmware

Tips and Tricks for Developing USB Devices

After writing firmware for a USB device, you must be prepared to debug it. But how? The process will go more smoothly if you have an understanding of USB's layered communications, data transfer types, device classes, and enumeration.

You've written firmware for your USB device and are ready to test it. You attach the device to a PC and the hardware wizard announces: "The device didn't start." Or, the device installs but doesn't send or receive data. Or, data is being dropped, the throughput is low, or some other problem presents itself. What do you do?



Photo 1—The device under test connects to the analyzer, which captures the data and passes it unchanged to the device's host. The cable on the back of the analyzer carries the captured data to the analyzer's host PC for display.

This article explores tools and techniques to debug the USB devices you design. The focus is on USB 2.0 devices, but much of the information also applies to developing USB 3.0 (SuperSpeed) devices and USB hosts for embedded systems.

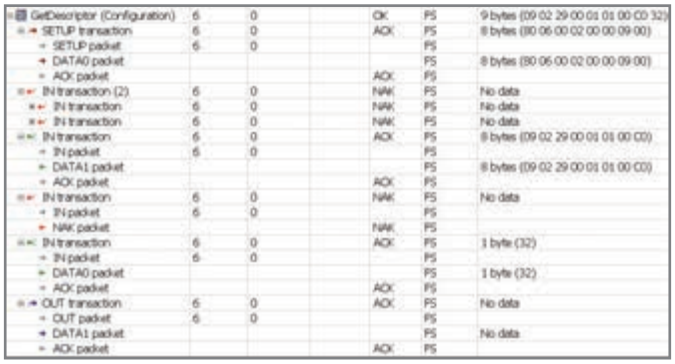
VIEWING BUS TRAFFIC

If you do anything beyond a small amount of USB developing, a USB protocol analyzer will save you time and trouble. Analyzers cost less than they used to and are well worth the investment.

A hardware-based analyzer connects in a cable segment upstream from the device under test (see Photo 1). You can view the data down to each packet's individual bytes and see exactly what the host and device did and didn't send (see Photo 2). An analyzer can also decode data to show standard USB requests and class-specific data (see Photo 3).

To avoid corrupted data caused by the electrical effects of the analyzer's connectors and circuits, use short cables (e.g., 3' or less) to connect the analyzer to the device under test.

Software-only protocol analyzers, which run entirely on the device's host PC, can also be useful. But, this kind



GetDescriptor (Configuration)	6	0		OK	FS	9 bytes (09 02 29 00 01 01 00 C0 32)
SETUP transaction	6	0		ACK	FS	8 bytes (80 06 00 02 00 00 09 00)
SETUP packet	6	0			FS	
DATA0 packet					FS	8 bytes (80 06 00 02 00 00 09 00)
ACK packet				ACK	FS	
IN transaction (2)	6	0		NAK	FS	No data
IN transaction	6	0		NAK	FS	No data
IN transaction	6	0		NAK	FS	No data
IN transaction	6	0		ACK	FS	8 bytes (09 02 29 00 01 01 00 C0)
IN packet	6	0			FS	
DATA1 packet					FS	8 bytes (09 02 29 00 01 01 00 C0)
ACK packet				ACK	FS	
IN transaction	6	0		NAK	FS	No data
IN packet	6	0			FS	
NAK packet				NAK	FS	
IN transaction	6	0		ACK	FS	1 byte (32)
IN packet	6	0			FS	
DATA0 packet					FS	1 byte (32)
ACK packet				ACK	FS	
OUT transaction	6	0		ACK	FS	No data
OUT packet	6	0			FS	
DATA1 packet					FS	No data
ACK packet				ACK	FS	

Photo 2—This bus capture shows the host's request for a configuration descriptor and the bytes the device sent in response. Because the endpoint's maximum packet size is eight, the device sends the first 8 bytes in one transaction and the final byte in a second transaction.

of analyzer only shows data at the host-driver level, not the complete packets on the bus.

DEVELOPMENT STRATEGIES

The first rule for developing USB device firmware is to remember that the host computer controls the bus. Devices just need to respond to received data and events. Device firmware shouldn't make assumptions about what the host will do next.

For example, some flash drives work under Windows but break when attached to a host with an OS that sends different USB requests or mass-storage commands, sends commands in a different order, or detects errors Windows ignores. This problem is so common that Linux has a file, unusual_devs.h, with fixes for dozens of misbehaving drives.

The first line of defense in writing USB firmware is the free USB-IF Test Suite from the USB Implementers Forum (USB-IF), the trade group that publishes the USB specifications. During testing, the suite replaces the host's USB driver with a special test driver. The suite's USB Command Verifier tool checks for errors (e.g., malformed descriptors, invalid responses to standard USB requests, responses to Suspend and Resume signaling, etc.). The suite also provides tests for devices in some USB classes, such as human interface devices (HID), mass storage, and video.

Running the tests will usually reveal issues that need attention. Passing the tests is a requirement for the right to display the USB-IF's Certified USB logo.

LAYERED COMMUNICATIONS

Like networks, USB communications have layers that isolate different logical functions (see Table 1). The USB protocol layer manages USB transactions, which carry data packets to and from device endpoints. A device endpoint is a buffer that is a source or sink of data at the device. The host sends data to Out endpoints and receives data from In endpoints. (Even though endpoints are on devices, In and Out are defined from the host's perspective.)

The device layer manages USB transfers, with each transfer moving a chunk of data consisting of one or more transactions. To meet the needs of different peripherals, the USB 2.0 specification defines four transfer types: control, interrupt, bulk, and isochronous.



Photo 3—This display decodes a received configuration descriptor and its subordinate descriptors.

The function layer manages protocols specific to a device's function (e.g., mouse, printer, or drive). The function protocols may be a combination of USB class, industry, and vendor-defined protocols.

CONTROLLER ARCHITECTURES

The layers supported by device firmware vary with the device hardware. At one end of the spectrum, a Future Technology Devices International (FTDI) FT232R USB UART controller handles all the USB protocols in hardware. The chip has a USB device port that connects to a host computer and a UART port that connects to an asynchronous serial port on the device.

Device firmware reads and writes data on the serial port, and the FT232R converts it between the USB and UART protocols. The device firmware doesn't have to know anything about USB. This feature has made the FT232R and similar chips popular!

An example of a chip that is more flexible but requires more firmware support is Microchip Technology's PIC18F4550 microcontroller, which has an on-chip, full-speed USB device controller. In return for greater firmware complexity, the PIC18F4550 isn't limited to a particular host driver and can support any USB class or function.

Each of the PIC18F4550's USB endpoints has a series of registers—called a buffer descriptor table (BDT)—that store the

Function layer (class, industry, and vendor protocols)
Device layer (control, interrupt, bulk, and isochronous transfers)
USB Protocol layer (USB transactions)

Table 1—USB communications use layers, which are each responsible for a specific logical function.

endpoint buffer's address, the number of bytes to send or receive, and the endpoint's status. One of the BDT's status bits determines the BDT's ownership. When the CPU owns the BDT, firmware can write to the registers to prepare to send data or to retrieve received data. When the USB module owns the BDT, the endpoint can send or receive data on the bus.

To send a data packet from an In endpoint, firmware stores the bytes' starting address to send and the number of bytes and sets a register bit to transfer ownership of the BDT to the USB module. The USB module sends the data in response to a received In token packet on the endpoint and returns BDT ownership to the CPU so firmware can set up the endpoint to send another packet.

To receive a packet on an Out endpoint, firmware stores the buffer's starting address for received bytes and the maximum number of bytes to receive and transfers ownership of the BDT to the USB module. When data arrives, the USB module returns BDT ownership to the CPU so firmware can retrieve the data and transfer ownership of the BDT back to the USB module to enable the receipt of another packet.

Other USB controllers have different architectures and different ways of managing USB communications. Consult your controller chip's datasheet and programming guide for details. Example code from the chip vendor or other sources can be helpful.

DEBUGGING TRANSACTIONS

A USB 2.0 transaction consists of a token packet and, as needed, a data packet and a handshake packet. The token packet identifies the packet's type (e.g., In or Out), the destination device and endpoint, and the data packet direction.

The data packet, when present, contains data sent by the host or device. The handshake packet, when present, indicates the transaction's success or failure.

The data and handshake packets must transmit quickly after the previous packet, with only a brief inter-packet delay and bus turnaround time, if needed. Thus, device hardware typically manages the receiving and sending of packets within a transaction.

Field	Size (bytes)	Description
bmRequestType	1	Direction (bit 7):
		0 = Data stage is OUT (host to device) or no Data stage
		1 = Data stage is IN (device to host)
		Request type (bits 6–5)
		00 = standard USB request
		01 = USB class request
		10 = vendor-defined request
		Recipient (bits 4–0)
		00000 = device
		00001 = interface
		00010 = endpoint
		00011 = other
bRequest	1	Identifies the request. Defined by a USB specification or a vendor driver.
wValue	2	Defined by the request.
wIndex	2	Defined by the request. A common use is to specify an endpoint address or interface number.
wLength	2	The number of data bytes the host will send or the maximum number of data bytes the host is requesting in the transfer's Data stage.

Table 2—Device firmware should fully decode the data received in a control transfer's Setup stage. (Source: USB Implementers Forum, Inc.)

For example, if an endpoint's buffer has room to accept a data packet, the endpoint stores the received data and returns ACK in the handshake packet. Device firmware can then retrieve the data from the buffer. If the buffer is full because firmware didn't retrieve previously received data, the endpoint returns NAK, requiring the host to try again. In a similar way, an In endpoint will NAK transactions until firmware has loaded the endpoint's buffer with data to send.

Fine tuning the firmware to quickly write and retrieve data can improve data throughput by reducing or eliminating NAKs. Some device controllers support ping-pong buffers that enable an endpoint to store multiple packets, alternating between the buffers, as needed.

LOST DATA

In all but isochronous transfers, a data-toggle value in the data packet's packet identification (PID) field guards against missed or duplicate data packets. If you're debugging a device where data is transmitting on the bus and the receiver is returning ACK but ignoring or discarding the data, chances are good that the device isn't sending or expecting the correct data-toggle value. Some device controllers handle the data toggles completely in hardware, while others require some firmware control.

Each endpoint maintains its own data toggle. The values are DATA0 (0011B) and DATA1 (1011B). Upon detecting an incoming data packet, the receiver compares its data toggle's state with the received data toggle. If the values match, the receiver toggles its value and returns ACK, causing the sender to toggle its value for the next transaction.

The next received packet should contain the opposite data toggle, and again the receiver toggles its bit and returns ACK. Except for control transfers, the data toggle on each end continues to alternate in each transaction. (Control transfers always use DATA0 in the Setup stage, toggle the value for each transaction in the Data stage, and use DATA1 in the Status stage.)

If the receiver returns NAK or no response, the sender doesn't toggle its bit and tries again with the same data and data toggle. If a receiver returns ACK, but for some reason the sender doesn't see the ACK, the sender thinks the receiver didn't receive the data and tries again using the same data and data toggle. In this case, the repeated data receiver ignores the data, doesn't toggle the data toggle, and returns ACK, resynchronizing the data toggles. If the sender

mistakenly sends two packets in a row with the same data-toggle value, upon receiving the second packet, the receiver ignores the data, doesn't toggle its value, and returns ACK.

DEFINING A TRANSFER

All USB devices must support control transfers and may support other transfer types. Control transfers provide a structure for sending requests but have no guaranteed delivery time. Interrupt transfers have a guaranteed maximum latency (i.e., delay) between transactions, but the host permits less bandwidth for interrupt transfers compared to other transfer types. Bulk transfers are the fastest on an otherwise idle bus, but they have no guaranteed delivery time, and thus can be slow on a busy bus. Isochronous transfers have guaranteed delivery time but no built-in error correction.

A transfer's amount of data depends in part on the higher-level protocol that determines the data packets' contents. For example, a keyboard sends keystroke data in an interrupt transfer that consists of one transaction with 8 data bytes. To send a large file to a drive, the host typically uses one or more large transfers consisting of multiple transactions. For a high-speed drive, each transaction, except possibly the last one, has 512 data bytes, which is the maximum-allowed packet size for high-speed bulk endpoints.

What determines a transfer's end varies with the USB class or vendor protocol. In many cases, a transfer ends with a short packet, which is a packet that contains less than the packet's maximum-allowed data bytes. If the transfer has an even multiple of the packet's maximum-allowed bytes, the sender may indicate the end of the transfer with a zero-length packet (ZLP), which is a data packet with a PID and error-checking bits but no data.

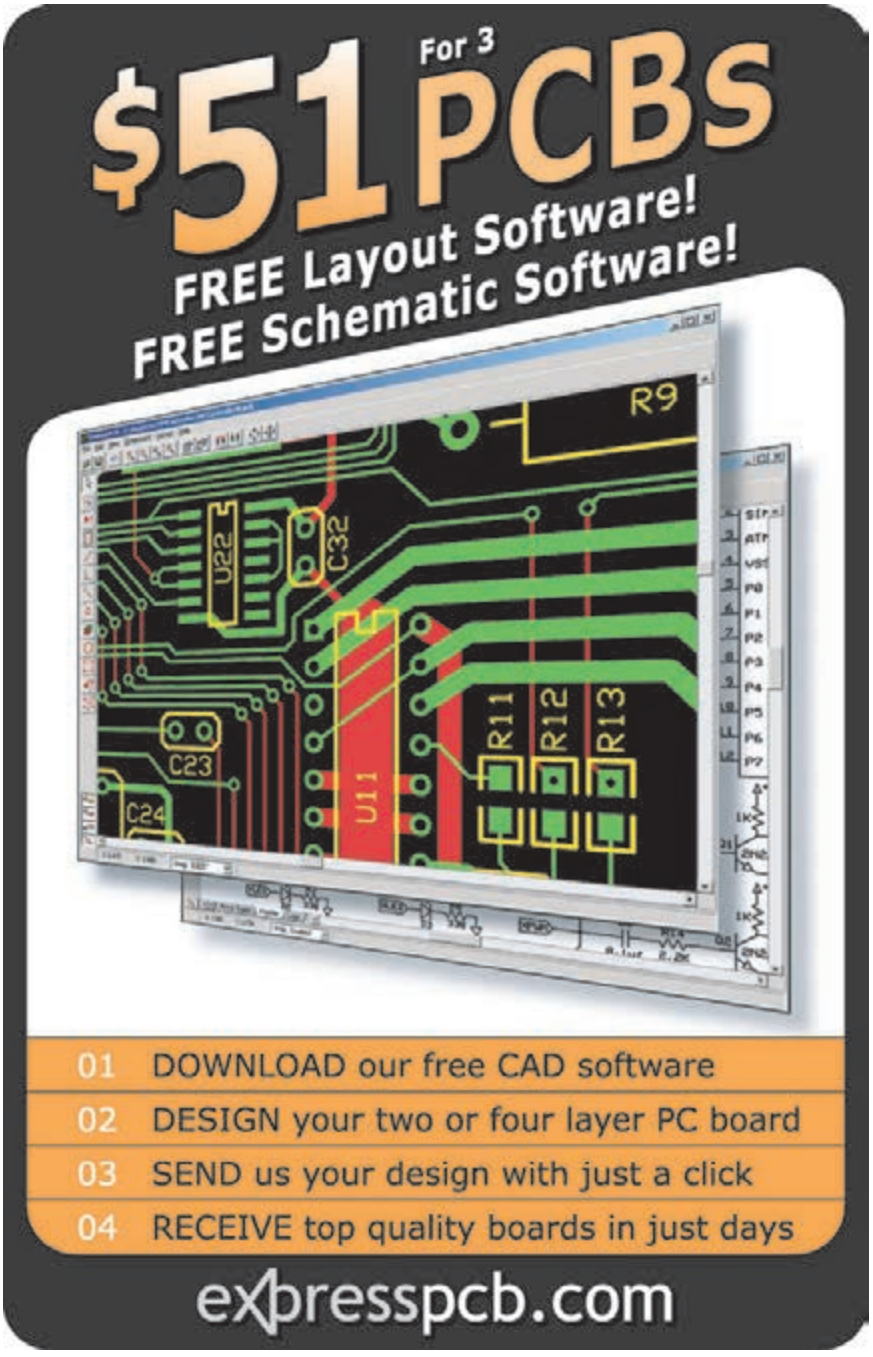
For example, USB virtual serial-port devices in the USB communications device class use short packets to indicate the transfer's end. If a device has sent data that is an exact multiple of the endpoint's maximum packet size and the host sends another In token packet, the endpoint should return a ZLP to indicate the data's end.

DEBUGGING ENUMERATION

Upon device attachment, in a process called enumeration, the host learns about the device by requesting a series of data structures called descriptors. The host uses the descriptors' information to assign a driver to the device.

If enumeration doesn't complete, the device doesn't have an assigned driver, and it can't perform its function with the host. When Windows fails to find an appropriate driver, the setupapi.dev.log file in \Windows\inf\ (for Windows 7) can offer clues about what went wrong. A protocol analyzer shows if the device returned all requested descriptors and reveals mistakes in the descriptors.

During device development, you may need to change the descriptors (e.g., add, remove, or edit an endpoint descriptor). Windows has the bad habit of remembering a device's previous descriptors on the assumption that a device will never change its descriptors. To force Windows to use new descriptors, uninstall then physically remove and



reattach the device from Windows Device Manager. Another option is to change the device descriptor's product ID to make the device appear as a different device.

DEBUGGING TRANSFERS

Unlike the other transfer types, control transfers have multiple stages: setup, (optional) data, and status. Devices must accept all error-free data packets that follow a Setup token packet and return ACK. If the device is in the middle of another control transfer and the host sends a new Setup packet, the device must abandon the first transfer and begin the new one. The data packet in the Setup stage contains important information firmware should completely decode (see Table 2).

The `wLength` field specifies how many bytes the host wants to receive. A device shouldn't assume how much data the host wants but should check `wLength` and send no more than the requested number of bytes.

For example, a request for a configuration descriptor is actually a request for the configuration descriptor and all of its subordinate descriptors. But, in the first request for a device's configuration descriptor, the host typically sets the `wLength` field to 9 to request only the configuration descriptor. The descriptor contains a `wTotalLength` value that holds the number of bytes in the configuration descriptor and its subordinate descriptors. The host then resends the request setting `wLength` to `wTotalLength` or a larger value (e.g., `FFh`). The device returns the requested descriptor set up to `wTotalLength`. (Don't assume the host will do it this way. Always check `wLength`!)

Each Setup packet also has a `bmRequestType` field. This field specifies the data transfer direction (if any), whether the recipient is the device or an interface or endpoint, and whether the request is a standard USB request, a USB class request, or a vendor-defined request. Firmware should completely decode this field to correctly identify received requests.

A composite device has multiple interfaces that function independently. For example, a printer might have a printer interface, a mass-storage interface for storing files, and a vendor-specific interface to support vendor-defined capabilities. For requests targeted to an interface, the `wIndex` field typically specifies which interface applies to the request.

INTERRUPT TRANSFER TIMING

For interrupt endpoints, the endpoint descriptor contains a bInterval value that specifies the endpoint's maximum latency. This value is the longest delay a host should use between transaction attempts.

A host can use the `bInterval` delay time or a shorter period. For example, if a full-speed In endpoint has a `bInterval` value of 10, the host can poll the endpoint every 1 to 10 ms. Host controllers typically use predictable values, but a design shouldn't rely on transactions occurring more frequently than the `bInterval` value.

Also, the host controller reserves bandwidth for interrupt endpoints, but the host can't send data until a class or vendor driver provides something to send. When an application requests data to be sent or received, the transfer's first transaction may be delayed due to passing the request to the driver and scheduling the transfer.

Once the host controller has scheduled the transfer, any additional transaction attempts within the transfer should occur on time, as defined by the endpoint's maximum latency. For this reason, sending a large data block in a single transfer with multiple transactions can be more efficient than using multiple transfers with a portion of the data in each transfer.

DEVICE FUNCTIONS

Most devices' functions fit a defined USB class (e.g., mass storage, printer, audio, etc.). The USB-IF's class specifications define protocols for devices in the classes.

For example, devices in the HID class must send and receive all data in data structures called reports. The supported report's length and the meaning of its data (e.g., keypresses, mouse movements, etc.) are defined in a class-specific report descriptor.

If your HID-class device is sending data but the host application isn't seeing the data, verify the number of bytes the device is sending matches the number of bytes in a defined report. The device should prepend a report-ID byte to the data only if the HID supports report IDs other than the zero default value.

In many devices, class specifications define class-specific requests or other requirements. For example, a mass storage device that uses the bulk-only protocol must provide a unique serial number in a string descriptor. Carefully read and heed any class specifications that apply to your device!

Many devices also support industry protocols to perform higher-level functions. Printers typically support one or more printer-control languages (e.g., PCL and Postscript). Mass-storage devices support SCSI commands to transfer data blocks and a file system (e.g., FAT32) to define a directory structure.

The higher-level industry protocols don't depend on a particular hardware interface, so there is little about debugging them that is USB-specific. But, because these protocols can be complicated, example code for your device can be helpful.

In the end, much about debugging USB firmware is like debugging any hardware or software. A good understanding of how the communications should work provides a head start on writing good firmware and finding the source of any problems that may appear.

Jan Axelson (jan@lvr.com) is the author of USB Embedded Hosts, USB Complete, *and* Serial Port Complete. *Jan's PORTS web forum is available at www.lvr.com.*

RESOURCES

Jan Axelson's Lakeview Research, "USB Development Tools: Protocol analyzers," www.lvr.com/development_tools.htm#analyzers.

USB Implementers Forum, Inc., www.usb.org.

SOURCES

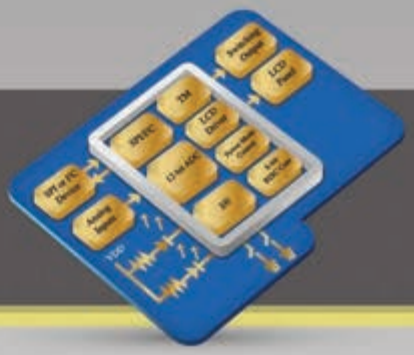
FT232R USB UART IC
Future Technology Devices International, Ltd. | www.ftdichip.com

PIC18F4550 Microcontroller
Microchip Technology, Inc. | www.microchip.com

HOLTEK
www.holtek.com

All In One

Ultra Low Power Consumption - Highly Functionally Integrated Flash MCU Solutions



The Holtek Flash MCU range of devices encompass an extensive range of peripheral functions making them suitable for use in a diverse application area such as health care products, instrumentation products, household appliances, industrial control, consumer products, automotive peripheral products to name but a few. With one range types also including Holtek's unique TinyPower™ technology, the ability is provided to meet the demands of today's environmentally conscious products.

- **HT68Fxx** Internal Multi-function Timer Modules. Internal oscillators with four frequency selections. Multi-function I/O pins with re-mapping function. 4 LCD SCOM outputs for direct driving of LCD panels. Multiple Communication Interfaces.
- **HT66Fxx** Includes all the HT68Fxx functions with an additional 12-bit ADC and internal reference voltage source.
- **HT67Fxx** Includes all the HT66Fxx functions but also uses Holtek's TinyPower ultra low power technology and also includes an internal R-type/C-type LCD driver. Also includes dual SPI or I²C interfaces which can be used simultaneously as well as internal Data EEPROM.

STD Flash MCU HT66/67/68Fxx Series

- Wide Operating Voltage Range: 2.2V~5.5V
- Multiple Low-power Operating Modes
- Fast Wake-up Function
- Industrial Specifications: -40°C ~ +85°C
- Multiple External Communication Interfaces
- Internal Accurate High/Low Speed Oscillators
- High Noise Immunity and Excellent ESD protection



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

HOLTEK SEMICONDUCTOR INC.

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

by Richard Lord (USA)

Digital Camera Controller (Part 2)

Code, User Interface, and Timing

The microcontroller-based Photo-Pal system is an electronic flash-trigger camera controller with four modes of operation. This article details the project's code, six-button user interface, and internal timing.

The first part of this article series, "Digital Camera Controller (Part 1): Hardware and Construction" (*Circuit Cellar* 267, 2012), covered the basic concepts, applications, and hardware of the Photo-Pal, which is a camera controller I created to explore high-speed flash photography. In Part 2, I present some aspects of the Photo-Pal's code development that have broader applications for other projects.

Today's trend is to create software applications for smartphones and tablets. More and more hardware projects are being designed around 32-bit processors in 200-pin, fine-pitch packages. This means you have to buy a whole preassembled system that won't fit in your handheld case and has a board with features you'll never use. Despite this trend toward prepackaged solutions, I still believe there is value in projects based on pocket-sized devices you can build yourself with less than $40 worth of parts. These devices are often a better fit for the intended purpose. What's more, successfully building them provides personal satisfaction and a sense of accomplishment. Besides, there are a lot of simple applications that don't need 32-bit microcontrollers. An 8-bit microcontroller can often adequately do the job. Eight-bit microcontrollers cost less than $5, draw less current, and are available in DIP packages that enable simple hand-tool construction.

The Photo-Pal controller is probably the 30th gadget I've designed since I started playing with 8-bit microcontrollers a few years ago, so, even an old dog like me has picked up a few new tricks. Most of these gadgets have required some sort of human interaction beyond flipping on the power switch, so I've tried various user interfaces with varying degrees of satisfaction. With the Photo-Pal project, I finally made significant progress in balancing hardware simplicity, compact design, intuitive operation, and significant capability.

SOFTWARE-DEFINED BUTTONS

When building a pocket-sized gadget, big fingers and lots of small buttons aren't a good combination. Besides, more buttons can mean losing precious I/O pins or adding more interface parts. One way to solve this dilemma is to give each button more than one purpose. I've seen many interfaces with three tiny printed labels next to each button (sometimes printed in three different colors) showing the button's function in each of the three modes. Aside from needing a magnifying glass to read the labels, the real problem is determining which mode you are currently in and what happens if three modes are insufficient.

If your gadget includes some sort of display, there is hope. It makes sense to locate several of the push buttons below the display and to use the lower line of the display to define each button's current function. This type of display eliminates confusion about which label is applicable because there is only one label displayed next to each button at a time. Changing the display gives the button a new function.

Compact, two-line, 16-character displays that can fit inside small plastic cases are available for less than $7. Thus, there are many inexpensive projects where character display becomes practical. However, 16-character-per-line displays present some interesting communication challenges, especially when limited to two lines. Four soft buttons are about the maximum that can be defined with one 16-character line. That is four characters (including white space) for each button. If the buttons are [IN OUT UP DOWN] you are probably safe. Changing them to READ, WRITE, RESET, and ENTER makes a little more work, but

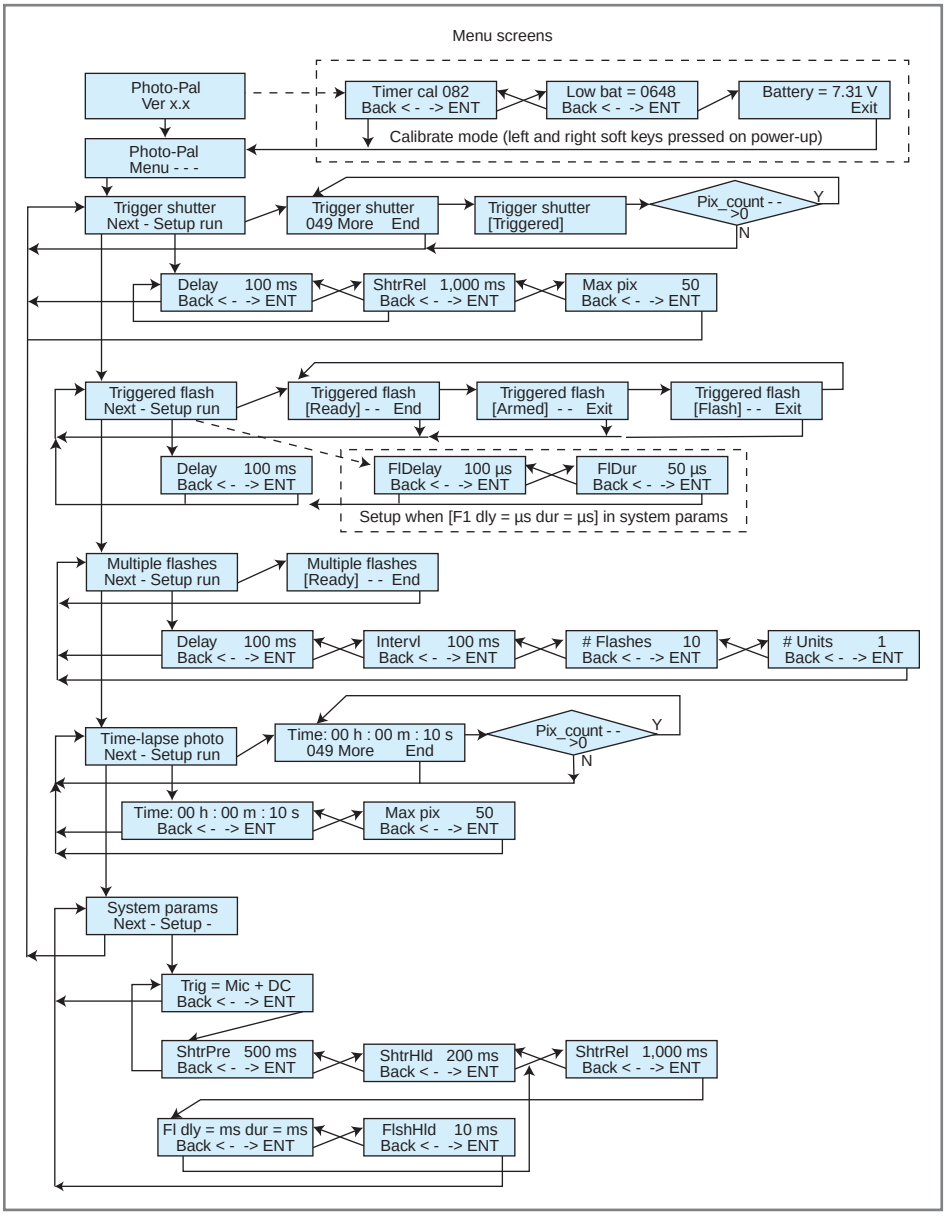


Figure 1—The Photo-Pal's screen map enables you to develop a logical flow pattern from screen to screen.

most people can manage [RD WR RESET ENT] without too much difficulty. The challenge is to find intuitive abbreviations for words that don't have simple, well-known abbreviations. Imagine dealing with the menu TRANSPOSE, TRANSFORM, INVERT, and REVERSE. Well, maybe you won't see those four abbreviations in a gadget with a 16-character display, but you get the idea.

One way around this problem is realizing you don't have to define and use all the buttons all the time. What if the previous problem was solved with a screen that displayed `[Transform YES NO]` followed by `[Invert YES NO ]` if the No button was pressed. The point is, you can do a tremendous amount with just four software-defined buttons and a 16-character display line, but sometimes you have to put some thought into the way you define and present the choices. I see it as a fun challenge (a combination of Sudoko and a crossword puzzle) to design a logical menu system that enables you to navigate between screens with software-defined keys.

From a programming perspective, there are several ways to manage the software screens and menus. For the Photo-Pal, many screens were handled on a case-by-case basis, but the data-entry screens were managed with subroutine calls to a standardized value input routine that returned with flags for the Back and Enter keys. For my digital photographic pan-head design ("Panning Control: A Digital Indexing Panoramic Tripod Head," *Circuit Cellar* 248, 2011), I used a vectored structure and assigned each screen a numeric constant that became an offset into tables of prompt strings and tables of vectors responding to each possible key press and mapping the screen links. This made it easier to change linkages if I decided to rearrange the order in which screens were presented. One of the other advantages of the vectored approach is the pan-head controller can be remotely operated by an RS-232 link able to remotely simulate a front-panel soft-key press. The remote controller can request the current screen's ID and can immediately display the same soft-key menu choices.

Before implementing the screens in software, create screen maps to show the user how to navigate from screen to screen. I developed a screen map for the Photo-Pal that helps you develop a logical flow pattern from screen to screen (see Figure 1). If your arrows inconsistently move around your map, you probably need to rethink how the screens should be linked.

PARAMETER ENTRY

The real test of a simplified human interface is the ability to easily enter data. The Photo-Pal is a classic case where a number of parameters have numeric values that can extend to five significant digits. When there is room for a full keyboard or a numeric keypad, data entry is a matter of pressing the correct digits and typing in the number. However, for a simple pocket device, a full keypad takes up a lot of room and uses many I/O pins. The other extreme is to only use Increase and Decrease keys to set the value. Obviously, no one wants to push the Increase key 10,000 times to enter a five-digit value, so there are various strategies to speed up the process. The simplest improvement is for the key to

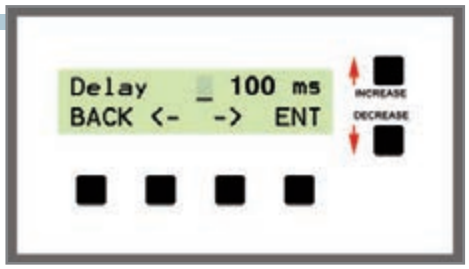


Figure 2—Two software-defined keys (labeled "<-" and "->") are used to move the data-entry point cursor left or right.

auto repeat if it is held down for 1 s. This is fine for one- or two-digit numbers, but it takes a long time to get from zero to 59,999 using this method. A more sophisticated approach speeds up the increase or decrease rate the longer the key is held. I used this method for my digital pan head's four-button interface. The problem is, once the number increases by hundreds of counts per second, it is nearly impossible to stop at the desired value without overshooting it and having to backtrack.

My solution for the Photo-Pal was to add two more push buttons to the user interface that are dedicated to increasing and decreasing the parameter's entered value. I then use the middle two software-defined keys (labeled "<-" and "->") to move the data-entry point cursor left or right (see Figure 2). The Increase and Decrease keys auto repeat at a 10-digit-per-second rate, but this occurs at the cursor position. If the cursor is moved two places to the left, the value increases or decreases by hundreds. I decided to let this overflow into the next digit, so the next step after 900 is 1,000 because I felt this was more intuitive. The movable cursor gives you a lot more data-entry flexibility. I used leading-zero suppression to make the displayed number easier to read. If the previously stored value shows only three digits, it may not be obvious that the entry can run to five digits. However, the entry-point cursor can only be moved as many places to the left as there are significant digits in the parameter.

Not all parameters are five-digit numbers, nor do they all start at zero. For example, the number of separate flash units can only be set between one and four. At the beginning of the program, I defined constants for the minimum and maximum value and the number of significant decimal digits (the parameter's "size") for each parameter (see Listing 1). You can't increase beyond the maximum value, you can't decrease below the minimum value, and you can't move the cursor left beyond the defined number of digits. The parameter's size definition also includes a flag to indicate if leading zeros should be displayed or suppressed when the parameter's value is displayed.

As an example, for a parameter with a maximum value of 199, you can only move the cursor two places to the left and the leftmost digit can only be incremented to "1." The next time the hundreds digit is incremented, the value goes to maximum. Thus, if you start at zero and move the cursor to the hundreds position, the first increase goes from zero to 100 and the second Increase key press jumps to the maximum value of 199. The Decrease key takes the value from 199 to 99, and then to the minimum value of zero.

Rather than write a separate data input routine for each parameter, I have a single `get_value` subroutine. To display and modify a parameter, I first set variables `max`, `min`, and `size` equal to that parameter's defined constants, then

Listing 1—At the beginning of the program, constants for each parameter were defined for the minimum and maximum value and the number of significant decimal digits.

```
;       Program Parameter DEFAULTS, LIMITS, and SIZES
;
FLSH_DLY EQU     D'100'   ; default "flash-delay" from trigger
FDLY_MAX EQU     D'49999' ; max "flash delay" in ms
FDLY_MIN EQU     D'0'             ; min "flash delay"
FDLY_SIZ EQU     0x45             ; unsigned 5 digits with zero suppress
;
FLSH_INTVAL      EQU     D'100'   ; default "flash-repeat interval"
FINT_MAX EQU     D'49999' ; max "flash repeat interval" in ms
FINT_MIN EQU     D'0'             ; min "flash repeat interval"
FINT_SIZ EQU     0x45             ; unsigned 5 digits with zero suppress
;
PHOTO_CT EQU     D'50'            ; default "photo count"
PHOTOCT_MAX      EQU     D'199'   ; max "photo count"
PHOTOCT_MIN      EQU     D'1'             ; min "photo count"
PHOTOCT_SIZ      EQU     0x43             ; unsigned 3 digits with zero suppress
;
FLSH_CT  EQU     D'10'            ; default "count of repeat flashes"
FLSHCT_MAX       EQU     D'99'            ; max "count of repeat flashes"
FLSHCT_MIN       EQU     D'1'             ; min "count of repeat flashes"
FLSHCT_SIZ       EQU     0x42             ; unsigned 2 digits with zero suppress
;
UNIT_CT  EQU     D'1'             ; default "count of flash units"
UNITCT_MAX       EQU     D'4'             ; max "count of flash units"
UNITCT_MIN       EQU     D'1'             ; max "count of flash units"
UNITCT_SIZ       EQU     0x01             ; unsigned 1 digit
;
;       System Parameter DEFAULTS, LIMITS and SIZES
;
SHTR_PRE_DLY     EQU     D'500'   ; default time from half to full press
SHPRE_MAX        EQU     D'19999' ; max "shutter pre-release time" in ms
SHPRE_MIN        EQU     D'0'             ; min "shutter pre-release time"
SHPRE_SIZ        EQU     0x45             ; unsigned 5 digits with zero suppress
;
SHTR_HOLD_DLY    EQU     D'200'   ; default time shutter button is held
SHHLD_MAX        EQU     D'19999' ; max "shutter hold time" in ms
SHHLD_MIN        EQU     D'0'             ; min "shutter hold time"
SHHLD_SIZ        EQU     0x45             ; unsigned 5 digits with zero suppress
;
SHTR_RLS_DLY     EQU     D'1000'  ; default delay after shutter is released
SHREL_MAX        EQU     D'49999' ; max "shutter release time" in ms
SHREL_MIN        EQU     D'0'             ; min "shutter release time"
SHREL_SIZ        EQU     0x45             ; unsigned 5 digits with zero suppress
```

copy the parameter into the value variable. The parameter's name is written into the display's upper line, then the get_value subroutine is called. The parameters' actual values are stored in binary in a single byte for parameters with a maximum value of 255 or less. They are stored in a 16-bit word for parameters with larger ranges. Single-byte parameters are copied into the least-significant byte of the value variable and the upper byte is cleared.

The get_value subroutine first displays the value variable then uses the Increase, Decrease, and cursor keys to manipulate it. The cursor keys increment or decrement the digit variable's value between zero and the size value and set the blinking cursor position in the display. The Increase key's code calls the inc_value subroutine that tests the digit value and determines whether to add decimal 1, 10, 100, 1,000 or 10,000 to the value variable each time it is called. Increasing the value past max sets it to max. A timer is set to the "auto-repeat wait time" when the key is first pressed. If the time expires while the key is still being pressed, the inc_value subroutine is called again, and the timer is set to the quicker "auto-repeat time" before looping back to see if the key is still being held down. The Decrease key's code works the same way, subtracting 1, 10, 100, 1,000, or 10,000, depending on the digit value and setting the value variable to min if it goes below it.

If either the Back or Enter key are detected in the get_value subroutine, it returns to the code for the parameter being modified. A flag is set if the Back key was pressed. If the return was caused by the Back key, the parameter entry subroutine exits and the caller uses the flag to move back to the previous screen. If the flag wasn't set, the ENT key was pressed, so the value variable's contents are copied into the parameter variable and written to the data EEPROM before returning. On subsequent powerup, the parameters are restored from the EEPROM's values.

KEEPING TIME

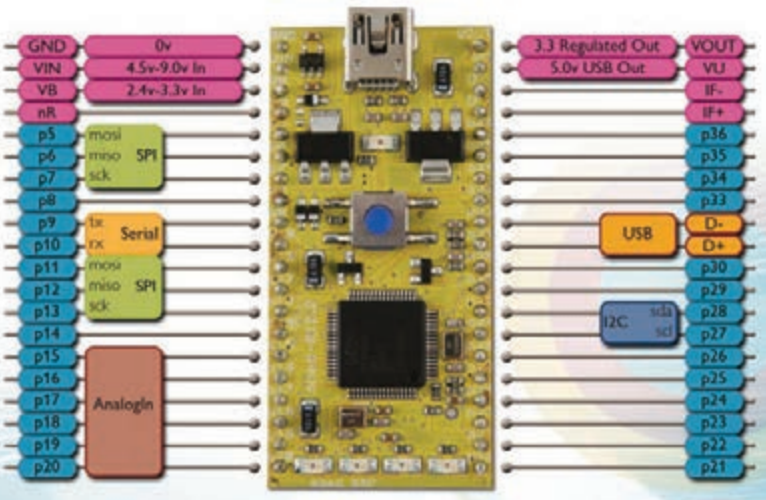
Unless greater resolution is selected for the flash delay or the flash output width, all internal timing resolutions are in milliseconds. One of the microcontroller's hardware counter/timers is configured to generate an interrupt once every millisecond. If the MSTIME flag is set in the flags variable, the interrupt handler tests the 16-bit variable msec_time. If it is nonzero, its value is decremented. When the value reaches zero, the MSTIME flag is cleared. Each time a delay is needed, the software writes the delay time into msec_time and sets the flag. It then tests the flag and waits until it is cleared.

The millisecond interrupt also decrements a "divide by 10" counter that is reloaded every time the count reaches zero, which occurs every 10 ms. After reloading the count, the 8-bit variable time10ms is tested and decremented if nonzero. This variable is used as a general-purpose delay timer for the auto-repeat function and various other situations where a 2-s or less delay is needed. When a delay is needed, the program writes the delay time (in tens of milliseconds) into the time10ms variable and tests to see if the value is zero. For auto repeat of the Increase and Decrease keys, the test is part of a loop that also tests for the key release.

The Photo-Pal provides the option of specifying the delay from trigger to flash and the width of the flash output with a microsecond resolution. When the delay is timed in microseconds, the delay time is preloaded into the microcontroller's 16-bit counter timer, which is configured to be clocked once every microsecond. When the comparator interrupt occurs signifying the trigger event has been detected, this timer is started. When this timer reaches zero, its interrupt starts the flash output. This provides resolution to within 1 µs. However, a number of instructions must be executed from the moment the trigger is detected until the timer starts. More instructions are needed after the timer interrupt is detected before the flash output can be set. This is the system's latency, and the delay cannot be less than this latency, which is around 40 µs.

mbed

mbed NXP LPC11U24 Microcontroller



Rapid prototyping for USB Devices, Battery Powered designs and 32-bit ARM® Cortex™-M0 applications

http://mbed.org

SEALEVEL®
Computing/HMI | Serial | I/O

Data Acquisition for Distributed I/O.

Seal/O™ data acquisition modules provide powerful digital, analog, and serial expansion to any computer. Connect to the host via wireless, Ethernet, USB, RS-485, or RS-232 to add the functionality required for your particular application. Multiple units can be daisy chained using convenient pass-through connectors to create a versatile distributed control and monitoring network.



Seal/O Solutions Offer:

- Wireless, Ethernet, RS-485, USB and RS-232 Connectivity
- Daisy Chain Expansion for Up to 247 Modules
- Optically Isolated Inputs, Relay Outputs, TTL, and Analog I/O
- Removable Screw Terminals to Simplify Field Wiring
- DIN Rail or Table Mount
- Sealevel SeaMAX Software for Windows® and Linux Operating Systems

Seal/O modules are perfect for a wide variety of applications including process control, data acquisition, broadcast communications and remote environmental monitoring.


Wireless


Ethernet


USB


RS-485


RS-232


Expansion

sealevel.com > sales@sealevel.com > 864.843.4343

Learn more about Seal/O Data Acquisition Modules at **sealevel.com/cir/seaio** or scan this QR code with your smart phone.

© 1986-2011, Sealevel Systems, Inc. All rights reserved.

The timeout is the first interrupt tested in the general-interrupt handler. All other interrupts are suppressed during this timer's countdown, so this latency is essentially constant. If the other interrupts weren't suppressed, the timeout could occur while the millisecond timer interrupt was in progress and the microsecond timeout wouldn't be detected until the millisecond interrupt completed, so it is important to suppress other activity to prevent a 20-to-30-µs jitter in the microsecond timing. With the other interrupts suppressed, the only other source of jitter could occur if the interrupt happens while the microcontroller is executing an indivisible two-cycle instruction (e.g., a conditional branch), in which case there will be an additional microsecond of delay for the interrupt to take place.

RESOLUTION VS. ACCURACY

An old riddle asks: "Which is more accurate, a clock that loses a minute per day or a clock that is broken and unable to run?" The stopped clock shows the precise time twice per day while the slow-running clock reads the correct time only once every two years. The Photo-Pal can specify delays to resolutions of up to one part in 60,000 µs or one part in 60,000 ms, but it doesn't mean a 10,000-µs delay is exactly 10 ms long. Maybe that delay is actually 9.997 ms. An inexpensive 4-MHz resonator is used to clock the microcontroller, so the frequency is not precise and can vary with temperature. In a piece of test equipment, that difference could be important, so laboratory equipment usually incorporates a precisely manufactured quartz crystal inside a temperature-controlled oven.

However, the Photo-Pal isn't being used to calibrate anything. It provides a delay between a trigger and the electronic flash firing. Why bother to provide five-digit resolution if there isn't five-digit accuracy? Whatever its accuracy, the delay is repeatable, and that is important for this application. What if you want to capture a picture of a water droplet the moment it strikes a surface? The droplet passes through a photodetector and, say, it takes around 45 ms

to reach the surface. The photograph taken with the Photo-Pal set to 46 ms may be different than the photograph set at 45 ms. If you try taking a picture with the Photo-Pal set at 45,500 µs (45.5 ms), you may get almost what you want, but not quite. With the Photo-Pal set at 45.75 ms, you get the photo you wanted. It doesn't matter whether the 45,750-µs number is precisely that many microseconds. What matters is the result is highly repeatable and the value is just enough delay to get the picture you want.

When it comes to the Photo-Pal's time-lapse mode, people tend to notice when one-hour intervals are only 55 min. long. Therefore, there is a calibration-tweaking parameter that adjusts how many Photo-Pal microseconds are in a real second. Another parameter sets the voltage below that which the "low battery" warning will display. These two calibration parameters are accessed and adjusted by holding down the leftmost and rightmost soft keys during powerup.

THE NEXT EVOLUTION

The six-button user interface, the four software-defined buttons, and the menu scheme I developed for the Photo-Pal device are intuitive and easy to use. The ability to quickly set and change values up to five digits long makes this interface attractive for several other projects I have in mind, so the Photo-Pal project has provided value beyond its specific implementation.

It seemed worthwhile to create a PCB to provide a generic microcontroller and six-button interface that could be mated with the peripheral circuitry needed to implement each new application. The next part of this article series will cover the design and implementation of this generic front-panel controller, which evolved from my original Photo-Pal project.

Richard Lord (rhlord@comcast.net) holds a BS in Electrical Engineering and an MS in Biomedical Engineering. During his career, he has designed digital electronics for an aerospace company and several telecommunication test equipment manufacturers. Working as a consultant in the 1980s, Richard designed several medical pulmonary test instruments and the electronics for an autonomous underwater robot. His interests include digital electronics, photography, jazz, and river conservation.

RESOURCES

R. Lord, "Panning Control: A Digital Indexing Panoramic Tripod Head," *Circuit Cellar* 248, 2011.

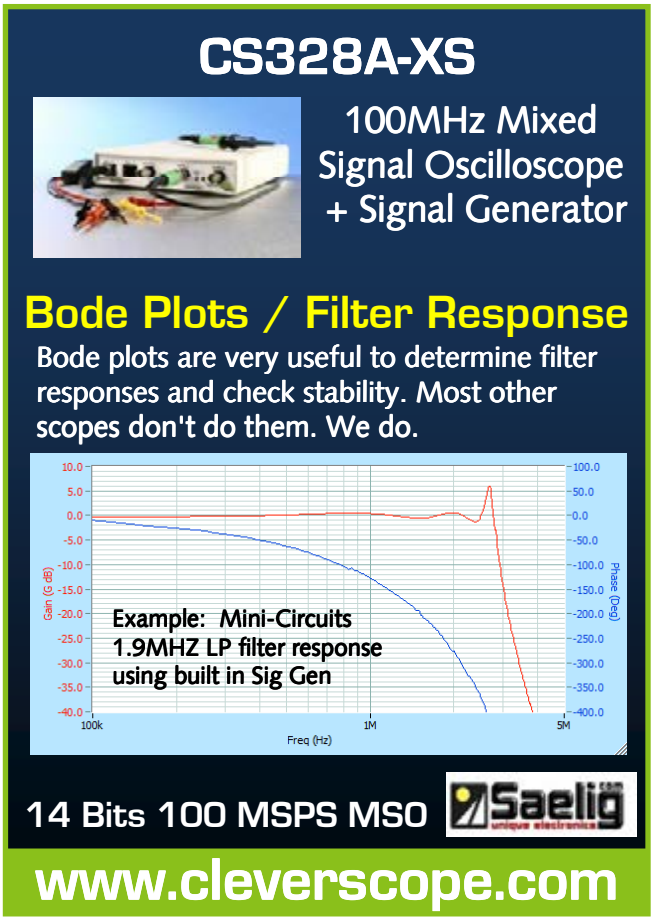
———, "Digital Camera Controller (Part 1): Hardware and Construction," *Circuit Cellar* 267, 2012.

SOURCES

PIC16F873A Microcontroller
Microchip Technology, Inc. | www.microchip.com

LM78L05 Regulators and ULN2803A Darlington transistor array
Texas Instruments, Inc. | www.ti.com

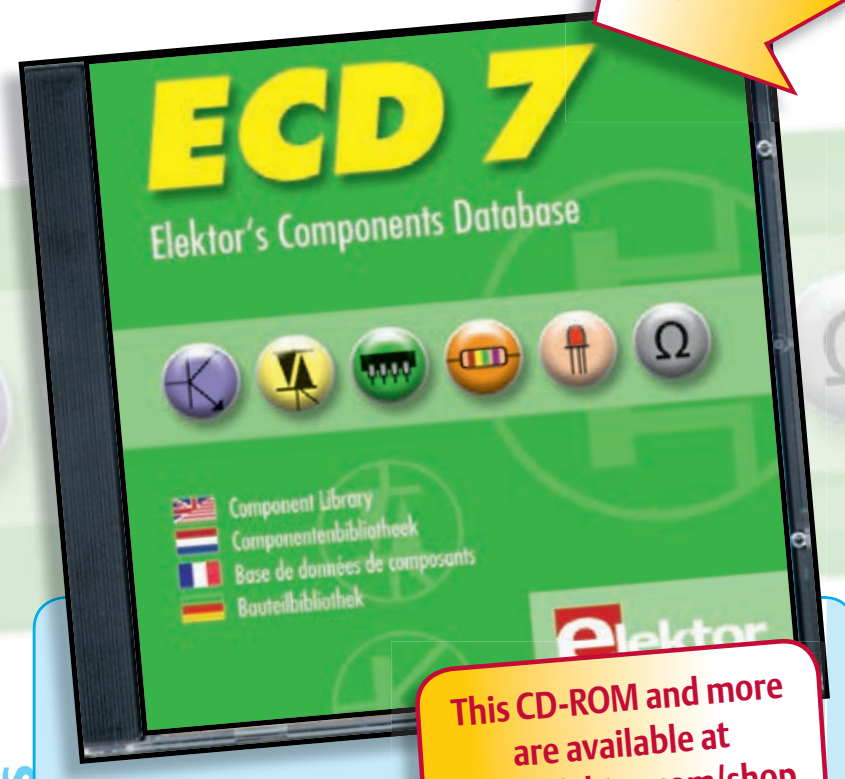




Elektor Shop

The world of electronics at your fingertips!

CD/DVD-ROMs



This CD-ROM and more are available at www.elektor.com/shop

More than 75,000 components

CD Elektor's Components Database 7

This CD-ROM gives you easy access to design data for over 11,100 ICs, 37,000 transistors, FETs, thyristors and triacs, 25,100 diodes and 2,000 optocouplers. The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. A colour band decoder is included for determining resistor and inductor values. All databank applications are fully interactive, allowing the user to add, edit and complete component data. This CD-ROM is a must-have for all electronics enthusiasts!

ISBN 978-90-5381-298-3 • $40.20



Circuits, ideas, tips and tricks from Elektor

CD 1001 Circuits

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

ISBN 978-1-907920-06-6 • $55.70



A whole year of Elektor magazine onto a single disk

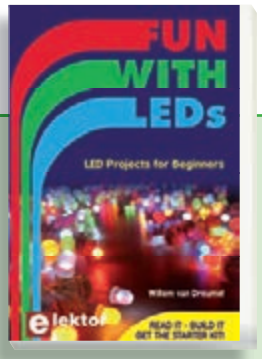
DVD Elektor 2011

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

ISBN 978-90-5381-276-1 • $37.90

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

Books



Associated 60-piece Starter Kit available

Fun with LEDs

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

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



Free Software CD-ROM included

Elementary Course BASCOM-AVR

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

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



LabWorX 2: Straight from the Lab to your Brain

Mastering Surface Mount Technology

Mastering Surface Mount Technology takes you on a crash course in techniques, tips and know-how to successfully introduce Surface Mount Technology in your workflow. Even if you are on a budget you too can jumpstart your designs with advanced fine pitch parts. Besides explaining methodology and equipment, attention is given to parts technology and soldering technique. Several projects introduce you step by step to handling surface mounted parts and the required technique to successfully build SMT assemblies. Many practical tips and tricks are disclosed that bring surface mounted technology into everyone's reach without breaking the bank.

282 pages • ISBN 978-1-907920-12-7 • $47.60

Elektor is more than just your favorite electronics magazine. It's your one-stop shop for Elektor Books, CDs, DVDs, Kits & Modules and much more!

www.elektor.com/shop

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



Embedded Linux Made Easy

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

Populated and tested Elektor Linux Board

Art.# 120026-91 • $93.30



USB Isolator

If your USB device ever suffers from noise caused by an earth loop or if you want to protect your PC against external voltages then you need a USB isolator. The circuit described in Elektor's October 2012 edition offers an optimal electrical isolation of both the data lines as well as the supply lines between the PC and the USB device.

Populated and tested Board

Art.# 120291-91 • $101.40

Kits & Modules



Hands-On Innovation

An Interview with David Penrose

For David Penrose, the best way to investigate a new technology is to start working with it. From the first time he assembled a transistor radio kit to his more recent work with MCU-based development kits, he has always taken a hand-ons approach to learning and innovation. David and I recently discussed his background in the aerospace industry and his creative approach to microprocessor design.—Nan Price, Associate Editor

NAN: Where are you located?

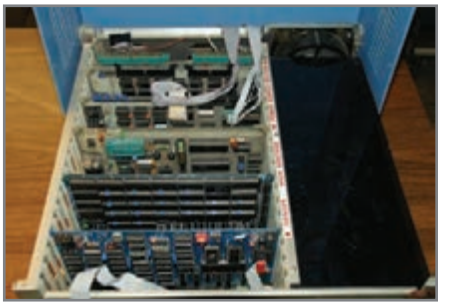
DAVID: I live in Bedford, NH. I was born in California. I grew up in the Monterey Bay area and later moved to the San Jose area, where I worked for 15 years. My company offered me the opportunity to move to Austin, TX to start a new operation, so I found myself going from a state with one season (mild and beautiful) to a state with two seasons (hot and not quite so hot). After about 13 years in Texas, the company again offered me the opportunity to move, this time to New Hampshire, a state with four very distinguishable seasons. This theme of weather and seasons has a lot to do with the microprocessor development I have done, since a number of my systems have been designed to capture seasonal data and display the current conditions using many devices.

NAN: How did you become interested in electronics?

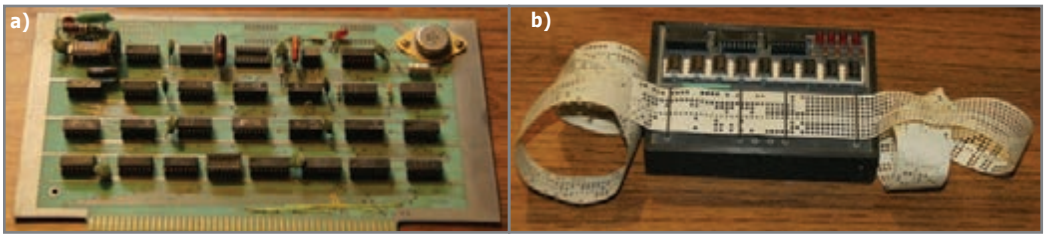
DAVID: My father was one of the first radio amateurs and used a "spark-gap" transmitter to communicate with people around the globe. I was always fascinated by his electronics equipment and soon found myself tearing apart surplus radio equipment from the military. I learned a lot from those times and can still remember the craftsmanship and artistry that went into those devices. This passion for tinkering continued as transistors appeared and I had the opportunity to assemble one of the first transistor radio receiver kits. I was really hooked at this point, because you could dig around in electronic devices without the dangers of being fried by the lethal voltages the tube devices used. I attended a community college to get certified as an electronic technician but was immediately intercepted by my instructor (a retired Navy technician) who told me to get out of his class and head for a degree instead. I took his advice and began to study mathematics and physics. I remember the "Aha! moment" quite well when mathematics actually explained how the physical world worked. It wasn't all theorems and proofs after all. I went on to obtain a math degree from the University of California at Berkeley and began graduate work at (then) San Francisco State college. They had an IBM 360 computer and one of my classes involved programming a computer using punched cards—my appetite was whetted. I later got a Master of Computer Science degree from the University of Santa Clara, and as part of that classwork had to design a microprogrammed pocket calculator. There was no going back after that.

NAN: Tell us about the first microprocessor you worked with.

DAVID: Microprocessors have been my way of capturing data around me and displaying it. It didn't start this way, however. Following my discovery of computer programming at college, I was offered a job at Lockheed Missiles and Space Company. I was a computer programmer using Fortran to run data analysis programs on complex systems being developed by brilliant systems engineers. I would punch up the program on cards, submit the card deck to the massive central computer, and wait a day or so to get the run. Each key punch or logical error meant at least a day wasted. I look back at those days and I am amazed at what we accomplished. It was about this time that a friend gave me a broken programmed data processor (PDP) computer. I found out the driver transistors were bad in the core memory (yes—little donuts of magnetic material with three wires strung through them). I replaced the transistors and had a working computer at home. As great as this sounds, it wasn't much use without peripherals, and back in those days, all interfaces were proprietary. It was still fun to use and I knew I was going to get a real computer with working I/O devices. That plan fell into place with the introduction of the IMSAI 8080 kit computer. I immediately ordered one and when it arrived began happily soldering together motherboards and processor cards. Silicon Valley was a great place to



David began his microprocessor work with the IMSAI 8080.



One of David's earliest projects used a Tarbell cassette tape reader to store and load programs (a). A paper tape reader was used to read prepunched programs (b).

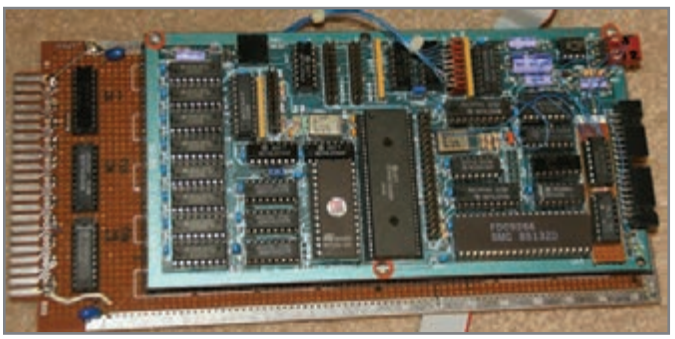
be at that time. I could drive down to 2102A Walsh Avenue and buy a kit from Solid State Music to assemble a 4-K memory card using 2102A memory chips—the poetry of it all. A trip to Halted Specialties or Haltek Electronics always yielded the latest devices discarded by the aerospace and electronics industry. The first I/O device I acquired was a surplus Navy teletype from a ship's weather system. It was gold plated on the inside and as long as I kept it slathered in oil, it clanked away at 110 bps using Baudot coding. There were a few minor problems since all the special characters were designed for weather reports. If I needed a "<" sign, it printed the symbol for thunderstorm. Reading a program listing meant interpreting the symbols for "thunderstorm scattered cirrus" to be "less than or equal." This wasn't a big problem and it made life fun. I added a Tarbell cassette tape reader to store and load programs, a paper tape reader to read prepunched programs, a floppy drive card, a Z80 CPU card to replace the original 8080 processor, and a PROM burner to burn my own PROMs. Prior to that, you could ship your data to a firm that would burn a PROM for you. It usually took a few round trips to get it right, but it beat keying in the data by hand. I wrote a one-chip (256-byte 1702 PROM) monitor for the computer that I sold at a computer hobby store in Palo Alto. That store was close to Stanford and the Homebrew Computer Club, which was an unbelievable forum for all things computer oriented.

As the industry moved along, the SB180 computer became available. Here was everything you could want from a Z80 and more. It shrunk my massive IMSAI down to a single card. (Note: If any appropriate institution would be interested in obtaining the IMSAI computer, I would love to donate it.) The ZCPR3 system running on CP/M became my standard for development and Pascal and C became available at reasonable costs. It was about this time that I decided to build my own computer to record and display weather data. The computer consisted of a motherboard, a Z80 processor board, an analog data board, a floppy and hard drive interface board, an interrupts processor, a time board, and standard I/O boards. All of this was encased in an IBM PC case and interfaced to a VIC-20 to display weather information on my TV set. I could turn to Channel 3 on any TV in the house and receive page after page of my weather data. If I wanted to know the temperature on the house's north, south, east, or west side, it was all there and recorded on floppy disks. This system ran for about eight years in Texas before it was packed up for the move to New Hampshire. There it was replaced with smaller single-board computers and USB drives. The industry had moved on.

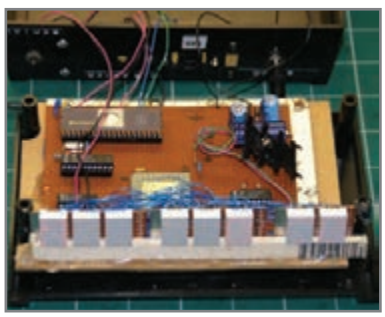
I haven't answered your question yet. So far, I've progressed from massive mainframes to smaller computers to microcomputers, but no microprocessors. Microprocessors entered the picture with Intel's 8748. Here was a chip that didn't need external memory, memory address decoding chips, parallel port chips, interrupt priority chips, DMA chips, and so forth. It left me wondering what I was going to do with all my breadboarding expertise. I had built a few projects with the 8048, but that chip still required external memory for the program. The 8748 made the big leap for me, since it had an internal user-programmable memory. The first project I did was a simple rehash of some of the work in the weather system—a wind speed and direction device that used numeric readouts. This project reused the case from a Heathkit Wind Meter. The transition to these devices was simple since the I/O logic was similar to Intel standalone chips and the instruction set was very similar to the 8080. Motorola's 6811 was also used in a few homebrew projects, but when Philips introduced the LPC line of processors, I began to focus. I started with the one-time programmable chips and progressed (thankfully) to the flash-programmable devices. It's amazing how many one-time programmable tombstones you can accumulate even when you think everything is checked and ready to go.

I've learned a lot of technical details in my years of putting projects together, but I learned the main lesson early on: Never automate something your spouse will have to fix when you are away. That clever thermostat project looks neat until it is the middle of winter, the furnace doesn't come on, and you are miles away. Try explaining the beauty of that project to your soul mate when you return.

Another important lesson when a project doesn't work or stops working is: Integrated circuits will probably never fail. Look to your programming first, second, and always. Look for the failed power supply or the poor solder



The SB180 consolidated David's IMSAI computer down to a single card.



David designed a wind speed and direction device that used numeric readouts.

connection. Never doubt the ICs meet the manufacturer's specifications until you have really exhausted every other possibility (check the errata sheet if there is one), and don't expect to find failed semiconductors unless you have done something really bad with the power or have failed to eliminate dataline spikes. And always add more decoupling capacitors than you think you may need. I use open-frame sockets and hide a –0.1-µF capacitor in each one.

NAN: You're now retired from the aerospace industry. Tell us a bit about your background and experiences.

DAVID: I am a very lucky person because of the great companies I have worked for and the fantastic people with whom I have worked. From processing massive data collections on a large central computer I took a job testing the Agena space vehicle. Now we had our dedicated computer (CDC 3200) and peripherals. Programs were still entered on punched cards, but you put the cards in the reader and immediately had results. No more overnight runs. Once the programs were checked, they were stored on large magnetic tapes or punched paper tapes then used to exercise the spacecraft. They would stimulate subsystems in the vehicle and collect response data. This data was analyzed to verify the systems were operating properly. Once the spacecraft was operating properly, it was shipped to the Vandenberg Air Force Base for mating to the boost vehicle, then it was launched. We got to operate in the blockhouse as we once again verified the spacecraft was operating properly and mated to the launch vehicle and payload. Any changes to the programs at this point meant you had to leave the blockhouse and move next door to the support building where the card punches were kept. You made the change and then ran the card back to the blockhouse to continue checkout. If the blockhouse was locked down for safety, you had a manual card punch. To punch an "A" you simultaneously pressed the 0, 2, and 8 keys and square holes appeared on the card. If you made an error, you had to start all over—or, if you were skilled enough, you could lick your finger and pick up one of the punched chads. The chad would swell because of the moisture on your fingertip and you could force it back into the card to erase the mistake. Thinking back, I have to give a tremendous amount of credit to all the engineers and programmers. This was before the use of any structured language and most of this programming was performed in Assembly language. The CDC machine had two primary registers, A and Q. A frequent lament heard when the program died was the old "load A store Q" mistake. I repeat this even today, only the registers have changed to A and B. The real-time operating system (RTOS) was also programmed for the project and the one programmer (yes, one person did the entire RTOS) who produced it had programmed in a halt operation that was the landing pad for any unresolved or unexpected errors in the program. The comment on that line was, "Should never get here." Guess which location was most frequently executed?

This was also the time before naming conventions, and as a tribute to the project manager, Rohrkemper, a routine was named after him. Unfortunately, it proved to be a troublesome routine, and it was always being reported in the morning status meeting that Rohrkemper had died again. The humor quickly wore thin.

Christmas was a special time with this group because of one of the talented programmers. The CDC machine had a speaker that could be toggled by one bit in one register for diagnostic purposes. The gifted programmer (on his after-hours time) managed to use this speaker, the high-speed, character-based, chain-line printer (a noisy thunk with each line printed), the vibrations from the vacuum card reader, and the whirr of four massive tape drives spinning backward and forward, to play a dramatic version of "Little Drummer Boy." You have never experienced such a Christmas if you haven't been in a freezing computer room at 2:00 A.M. (programmers always got the off shift), surrounded by spacecraft equipment and hearing and feeling "Little Drummer Boy" belted out by technology. As the song progressed and the line printer kept the drum beats, it was printing out a graphic scene of a little drummer boy in a snow storm. You had to look carefully to see that all of the scenes were built from the ASCII characters then used by the printer. The program was so good at exercising the equipment in a coordinated fashion the CDC technicians adopted it as a system-level checkout tool. Now that's programming!

All fun must come to an end. As I was promoted through management, these simple pleasures gave way to the challenges and rewards of getting teams of people to work together on large projects. I progressed through project management to program management and then to general management. All this while I was working on some of the greatest, most rewarding programs anyone could hope for. And, as I mentioned earlier, I was always surrounded by the most amazing people. During this time, the joys of homebrewing microprocessor projects helped me keep grounded in the technology. I remember one Thanksgiving where the routine was: check the turkey, run upstairs and make a few test runs on the weather computer, fix the dressing for the turkey, rewrite the code, set the table, and reboot and run the new software. What a great way to celebrate a holiday. It was also during this time that I was traveling frequently and decided to spend the air time studying code for my Ham radio license. As other folks listened to pop music on their Walkmans, all you could hear from mine was a series of dots and dashes. I got some funny stares from fellow passengers. It worked though—I tested for all classes of licenses (novice through extra) in one sitting and was finally a licensed amateur radio operator. My father had passed away by then, but I was very proud to think I was following in some of his footsteps.

I retired from aerospace and two weeks later I found myself working for a small startup company during the dot-com period. I was in charge of its engineering group and had all the fun of management, but I also got to be involved in all the hardware and software development. The company's goal was to perform identity verification for credit card companies, but unfortunately, it ran into a lot of public protest against maintaining a large database of driver's license information. The technology was fascinating and the people working there were all top-notch individuals, but the world was not ready for that type of company. It was bought out and moved to Georgia, so I went back to being retired.

Not long after, I was asked to return to aerospace to help



COMPLETE ENGINEERING SOLUTIONS

Start here.

At Newark element14, all your engineering needs come together in one source—vast product range from world-class brands, fast online search, seamless purchasing tools, resources and services, one-on-one support, and a community of experts. Here, you'll find simpler, smarter and faster ways to do business.

"The navigation and ordering process are easy to work. Thanks."

– Richard, Newark element14 customer

Newark element14

HOW MAY WE HELP YOU TODAY?

COMMUNITY: element14.com
WEBSITE: newark.com
PHONE: 1.800.463.9275
LEARN MORE: newark.com/together

manage my former company's Space Division, which now belonged to BAE Systems. I was back in heaven surrounded by the experts in the space technology field where I had previously worked. This business group built some of the most sophisticated spacecraft our country has ever produced. It also built its own space-qualified digital semiconductors in a foundry in Virginia. These processors, memories, and support chips have enabled the Mars rovers and the spacecraft transits of the planets in our solar system. The Space Division also had access to BAE Systems's specialized Microwave Electronics Center in New Hampshire, which built sensitive and very specialized analog semiconductor devices for both air and space receiver systems. The Space Division also operated a foundry in Massachusetts that built specialized sensitive infrared optical sensors for spacecraft that monitor our weather from space. The technology used to build these specialized integrated circuits was amazing to me. And the people working in these areas were the top specialists in their fields. All this, and they were paying me to work in these areas! Things change, however, and as BAE Systems reorganized for a smaller space market, I decided to retire again. My time is now spent working on my home projects, biking locally, and visiting with all the great friends I have made over many years in a wonderful industry.

NAN: Some of your projects have placed well in design challenges. Tell us about your creative process.

DAVID: *Circuit Cellar* (and *Byte* before that) have been my touchstones. Sometimes I think it is a gigantic plot to publish a magazine just for me. The contests I enter have mostly been those in *Circuit Cellar*. I select them based on a number of factors: What will I learn? What do I have to offer? Will the experience last?

Let me explain the last factor. Most systems today are complex and require considerable time to learn and develop proficiency. If the development environment is proprietary or if it is offered under a restricted time or size license, I generally pass on the opportunity. Being a hobbyist, I can't afford to purchase most of the expensive licenses to keep a development environment alive. I can't justify spending a large amount of time to learn a new system only to bump into a size restriction or have a license expire. I understand the need to recover costs spent to provide a development system, but there is little joy in developing a project to have it turn into a dead end when the IDE evaporates or restricts its growth.

I enjoy learning, so most of my projects are focused on exploring some new device or concept. I generally attack problems that are control oriented rather than algorithm oriented. If the project is going to require a complex algorithm, I generally pass on it. I have a lot of experience in some very narrow fields. If a project requires that I leave this comfort zone, I generally think it over for quite a while before starting. I know this sort of contradicts the learning desire, but I am also a realist and recognize that I have only so much time and talent.



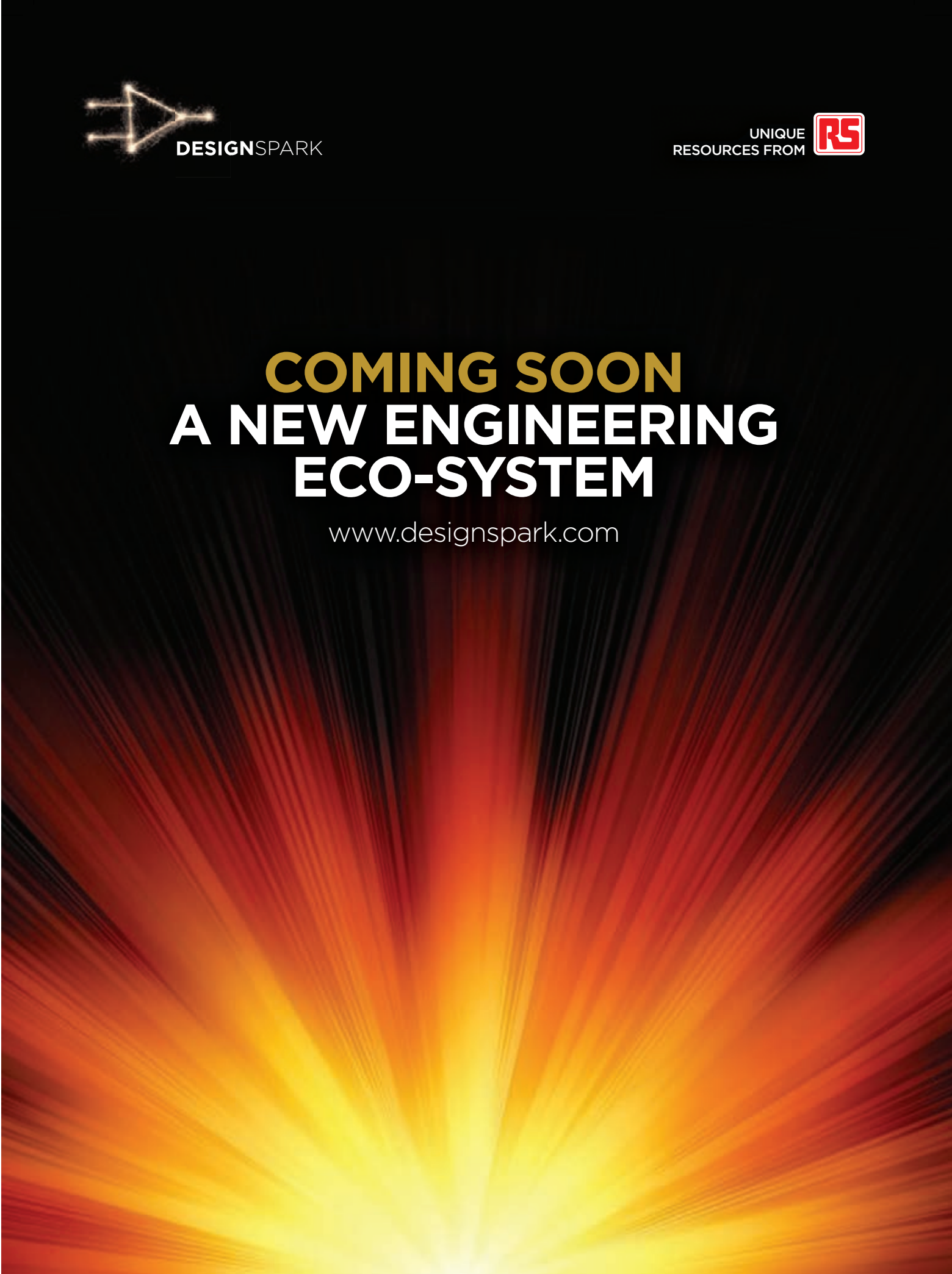
The Geo-Sentry senses ground vibrations and displays events.

NAN: You wrote five articles for *Circuit Cellar* since 2001. Tell us what inspired you to create the Geo-Sentry (*Circuit Cellar* 126, 2001), which senses ground vibrations and displays events on a fluorescent display screen.

DAVID: This was a fun system. The inspiration came from a purchase of a number of geophone sensors used in the oil field industry to record vibrations from ground contact air cannons to help map the underlying strata. I also had a surplus device from the military that used a similar sensing device to provide unattended monitoring of an area. This military device was a complex system with bunches of discrete parts that processed the sensor input and generated a radio signal based on detected motion. It seemed to me that I could make a simple device if I ignored fidelity and concentrated on just sensing any vibration. The project used a simple op-amp to square up the signal from the geophone and generate a one or zero condition based on signal level. The 451 processor did additional processing on the ones and zeroes to determine if a reportable event had occurred. The 451 processor had enough I/O pins that I could add any number of sensors and still drive the display, serial output, and debug pins. The display was one of a surplus pair I had bought. It is a very bright display that will actually light up my basement. I had the unit installed with about four geophones buried along my house's driveway and walkway. It worked great for quite a while until we had some landscaping done and the crew managed to sever all the wiring I had run along the driveway and walkway. The pieces of the project now sit in the basement awaiting resurrection.

The other display from the pair was incorporated into a project that monitored systems in my car. It also used a smaller processor but processed inputs from a road speed sensor, a compass, and a tilt indicator. It would record the direction and gradient as the car was driven. It was a great idea and looked good—and very professional—installed just below the radio in the console. There was just one problem: The display was so electrically noisy that if the unit was on, you couldn't pick up any radio stations. Now I understand why they do electromagnetic interface (EMI) and electromagnetic compatibility (EMC) testing on systems. The system was removed from the car when the road speed sensor managed to come loose from its mounting and wrap around the front axle shaft. The real world is a difficult environment to survive. The system also demonstrated quite convincingly the relationship between acceleration and apparent pitch. As the car accelerated, the pitch sensor thought it was going uphill. This would have been an interesting problem to solve had the car not eaten the speed sensor.

NAN: Your remote humidity control system (*Circuit Cellar* 247, 2011) calculates


DESIGNSPARK
UNIQUE
RESOURCES FROM
RS
COMING SOON
A NEW ENGINEERING
ECO-SYSTEM
www.designspark.com

water vapor pressure from temperature and humidity readings. What's the background on that project?

DAVID: The humidity control system was a great learning experience. I had not done much with TCP/IP or Internet protocols, so when the WIZnet computer was unveiled with a 8051 core and prepackaged support for TCP/IP, it became the perfect opportunity. I had gained some good experience with I^2C and temperature/humidity sensors, so it made sense to implement multiple remote sensors using this technology. My basement is the hub for all my spare parts, my CNC system, and a large assortment of wood- and metal-working equipment. Being a basement, it gets humid when the ground is thawed and the outside humidity is high. To deal with this rust- and corrosion-inducing environment, I run a fairly large dehumidifier. There are times during the year when it is very dry outside and the outside air would do a great job taking the humidity out of the basement, hence the inception of the humidity control system. The WIZnet processor monitors indoor and outdoor sensors and calculates water vapor pressure to determine when it can "open up the basement" and enable the outside air to reduce the humidity rather than the dehumidifier. Since the processor has TCP/IP capability, it was a natural step to put all the environmental and control information on an HTML page to let me see what was happening. This also enabled me to override any decisions the system was making from any computer behind my firewall. Using the Vdrive subsystem to add a USB drive enabled me to make a record of all the data at 4-s intervals for years and years. The big inducement was *Circuit Cellar* and WIZnet were offering me contest money to have fun. The winnings from this project bought me a nice carbon fiber road bike that is great for exercise.

NAN: Tell us about your timeserver system project (*Circuit Cellar* 258, 2012). Why did you design it?

DAVID: Almost all my projects incorporate an I^2C real-time battery backed-up clock. As the saying goes "a person with two watches never knows what time it is." Sure enough, even though these clocks are crystal driven, they all march to their own drummer and drift over time. I had initially included logic in these clocks to adjust for daylight saving time, but when the government decided to change the dates, a number of my projects were keeping the wrong time. There are a number of inexpensive clock chips that solve these problems using WWVB, which is the National Institute of Standards and Technology (NIST) time signal radio station based in Colorado. Where I live in New Hampshire, that signal is only reliable at night and then mostly at night during winter. Errors in misreading this signal turn a reliable clock into a real headache. I decided it was not worth changing to these clock sources because of these problems. The solution then became a master clock in my house that all my projects could use to update their own free-running clocks. If I was going to have just one master, it made sense to make it as accurate and reliable as possible. So the GPS time signal was selected as the master source. I dug up an old GPS receiver that was used with a serial interface to a PC and built a microprocessor to read this data, adjust for daylight saving time, and transmit this time to remote receivers throughout my house. Since there are a number of these remote receivers, I wanted to keep costs to a minimum. I used some inexpensive LINX OOK transmitter/receivers. Once the receiver has a valid time word, it transfers it to the project's I^2C clock by assuming a temporary master condition on the bus. The time signal transmitter also incorporates its own clock that is slaved to the GPS and enables the system to freewheel through GPS outages. The problem of keeping a single time throughout my house is solved, at least for my projects. Other commercial devices are still trying to keep their own time, and with every power outage I still have to make a pilgrimage through the house resetting clocks. Living in a heavily forested state with an abundance of wild weather means many power outages, so I've developed a definite routine to handle them. At least I don't need to worry about my projects any longer.

NAN: What is the "next big thing" in the embedded design industry?

DAVID: When I was working in Sunnyvale, CA, one of my projects was to interface to a much larger project. This larger project had computer systems spread in seven different 40' vans and controlled a triad of sensor aircraft and a number of support aircraft. The program manager for this massive project was brilliant and one of the nicest individuals you could hope to work with, but he was not a computer engineer. One day while we were talking, he mentioned he was going to buy one of the new IBM PCs. This surprised me a bit, and I asked him what he was going to do with it. He wasn't a programmer (as I was) so what could he possibly want it for? He said he wanted to run spreadsheets and play games. I chuckled and told him that I stood by to purchase the "boat anchor" from him as soon as he found out how useless it was unless he could program. Needless to say, I never had the opportunity to purchase this dead-end technology. This story is meant to illustrate how totally inept I am in predicting the future, but nevertheless I will do a bit of introspection.

Smartphones and tablet devices are the ruling technology today, and any embedded device will have to talk to these devices if they are going to be accepted as consumer devices. Bluetooth seems to be the easiest and least expensive technology, although near-field communications and RFID seem to have a contribution to this area. Just as the microprocessors of old absorbed the serial communication chips. (Remember the 8251 IC?) I think new embedded devices will absorb these communications protocols and hardware and provide a single-chip solution that integrates seamlessly to smart handheld devices and to each other. I would love to be able to configure the communication channel in my next project and have it ready to communicate to the outside world with no additional hardware or software. Until then, I am purchasing a bunch of Bluetooth transceivers and, from my comfortable workstation, I have begun replacing some of my serial interfaces with these cards.

Editor's note: The rest of the interview with David Penrose is available online at CircuitCellar.com/interviews.

A BETTER PLACE FOR ELECTRONICS KITS

It's only a great project if I can build it myself!

- **A wild collection of maker-designed electronics projects**
- **All projects packaged as step-by-step kits available for sale**
- **Kit designers earn a commission on every sale**
- **Request a project and help influence the design process**
- **DIY inspiration**

Dust off that workbench, fire up the soldering iron and get back in the garage. Visit www.ClubJameco/Fireup to join the fun.



by Patrick Schaumont (USA)



True Random Number Generation

A true random number generator implements the equivalent of a coin flip in hardware. The design challenge is to ensure the coins are unbiased and the coin flips are unpredictable. This article demonstrates a hardware implementation using ring oscillators. The Random Number Generator has a built-in self-test to check the quality of the randomness.

Random numbers are extensively used when implementing cryptographic protocols. During communication link initialization, they are used as nonces (i.e., numbers-used-once) to make the protocol resistant against replay attacks. Public-key cryptography requires random numbers to create key pairs and signatures. Also, casinos would be unable to build their businesses without good random number generators.

Defining what exactly makes a good random number generator is challenging. There's a Dilbert comic in which he visits the accounting department's "random number generator," which turns out to be a dog uttering the infinite sequence, "nine, nine, nine…" Upon asking if that sequence is actually random, Dilbert is told, "with randomness, you can never be sure!" In a true random number generator, indeed, no sequence is completely improbable. However, it's clear that a constant sequence of numbers is unlikely, as is using dice to throw an infinite series of sixes.

Random number generators used for cryptography (and slot machines, for that matter) have two fundamental requirements. First, the random number sequence should have good statistics. Second, the sequence should be unpredictable. The first property can be verified with statistical tests, and this article will describe and implement several of them. The second property, unpredictability, must be guaranteed by the random number generator's design. In particular, it requires you to precisely identify the source of randomness. Indeed, *unpredictable* and *unknown* are two different things!

In this article, I'll describe a full-hardware implementation of a true random number generator, including an online testing facility that continuously checks the random number stream's quality. There's a good reason to prefer a hardware implementation. A software program (e.g., C) is fundamentally deterministic. A true random number generator must be unpredictable, so it's not possible to write one in C

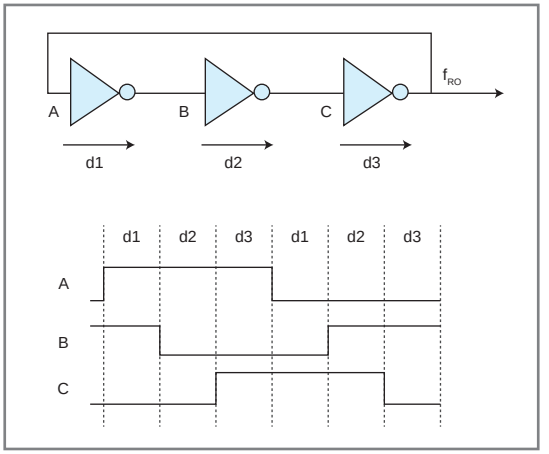


Figure 1—A ring oscillator is built using an odd number of inverters. It oscillates with an average period of twice the propagation delay through the inverters.

alone. Let's first look at the different types of random number generators and the available sources of randomness in digital hardware.

PRNG & TRNG

There are two major types of random number generators, pseudorandom number generators (PRNGs) and true random generators (TRNGs). There's an important difference between them.

A PRNG is a finite state machine that moves through a long sequence of states. Each state corresponds to a random number. Eventually, the finite state machine gets back to the initial state and the sequence of random numbers restarts. The bottom line, though, is that a PRNG is fully determined through the state register's value. Once the state is known, the entire random sequence is fixed. The random number generator in the standard C library works in this way. That random number generator is controlled through two functions, `rand()` and `srand()`. Calling `rand()` returns the next number from the pseudorandom sequence. It advances the finite state machine to the next state. Calling `srand(seed)` uses the `seed` value to initialize the finite state machine's state. For a given value of `seed`, the same sequence will appear from `rand()`.

A TRNG uses a different approach. A TRNG attempts to implement the equivalent of a coin flip. You can think of a coin flip as binary random variable: an event with two outcomes (heads or tails), each with the same 0.5 probability. By encoding a head as "1" and a tail as "0," you can make a TRNG that produces one random bit per coin flip. A 50% probability is the ideal, most preferable case. If you have a biased coin (e.g., one that gives 10% heads and 90% tails), the random bitstream will also be biased. There will be nine "0s" for every "1," on average. A biased random variable is less suited to create a TRNG because the random bitstream becomes predictable. Thus, in an ideal TRNG, each output bit should be completely independent of the others, and should be "0" or "1" with equal probabilities. This property, which is one method of evaluating the TRNG quality, can be estimated with a statistical test.

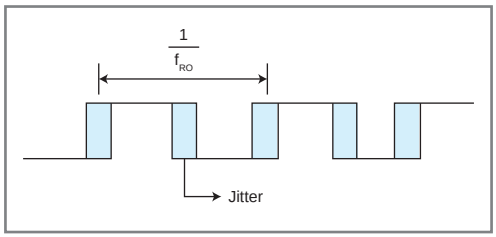


Figure 2—Ring oscillation clock jitter is the clock transition's time uncertainty. Jitter is caused by electrical and thermal noise in the ring oscillator. With the accumulation of jitter, a ring oscillator's clock phase becomes a random variable.

Anything that can be treated as a random variable can potentially be used as a source of randomness in a TRNG. For example, the RANDOM.ORG website quantizes atmospheric noise, presumably at an A.M. or F.M. frequency near Dublin, Ireland, and converts this into random numbers as a web service. A second example, the Linux random number generator (which is a PRNG) is seeded with randomness extracted from keyboard and mouse events, disk events, and system interrupts.

In a small embedded system, the access to external random variables may be limited. There may be no web connection or few random events occurring. In that case, the embedded system may be better off using a built-in random variable, a source of physical noise. This is the approach I will take for my design.

JITTER AS A SOURCE OF RANDOMNESS

A digital TRNG requires a random variable in digital logic. Digital systems are, of course, deterministic. Hardware designers go to great lengths to ensure digital hardware's behavior is predictable. Luckily (for TRNGs) there are a few properties in digital hardware that are nondeterministic. One is flip-flop metastability and another is oscillation clock jitter. My design uses oscillation clock jitter, which is the uncertainty in a digital clock's transitions.

Figure 1 shows a ring oscillator, which is a feedback structure built using an odd number of inverters. It's easy to verify that the circuit has no stable state, therefore the circuit must oscillate. The oscillation period is determined by the propagation delay through the inverters. With fewer inverters, the oscillation frequency is higher.

The ring oscillator's transitions are not as fast and sharp as shown in Figure 1. The reality looks more like what is shown in Figure 2. A clock signal's exact transition time contains an uncertainty, determined by the circuit's thermal and electrical noise. This uncertainty is called jitter. The region of uncertainty is very small. Typical values in FPGA ring oscillators are 1% to 2% of the clock period. However, over multiple clock periods, the uncertainty accumulates, and eventually it spreads out over the entire clock period.

This way, the ring oscillator's clock phase becomes a random variable. Besides jitter, a ring oscillator can also be affected by other types of noise (e.g., drift). All of this noise contributes to randomizing the oscillator clock phase.

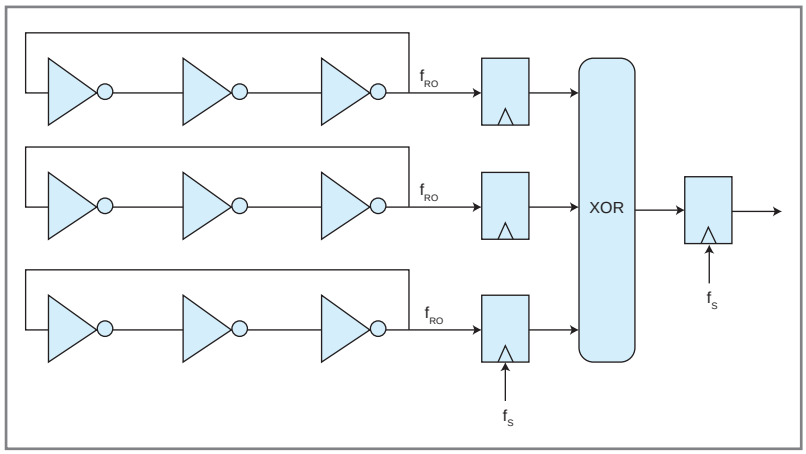


Figure 3—Here is a TRNG using three ring oscillators built with three inverters. Each ring oscillator's output is sampled at f_S, typically much lower than f_{RO}. The sampled bits are XORed together and captured at the output. When at least one ring oscillator is in the jitter window, the circuit's output is a sampled random variable. The number of ring oscillators and the number of inverters per oscillator are design parameters.

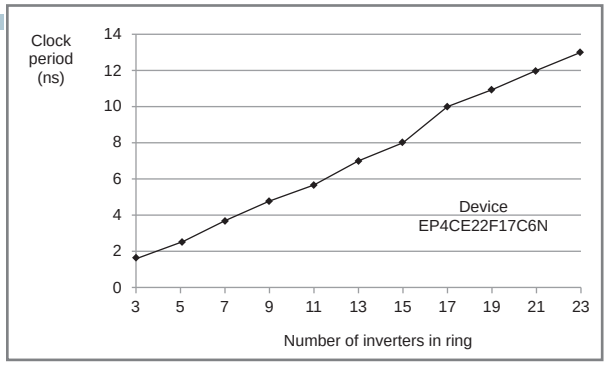


Figure 4—I used an Altera Cyclone IV EP4CE22F17C6N FPGA to produce this graph, which shows the ring oscillation clock period as function of the number of inverters in a ring. Note that a three-stage ring oscillator runs at 600 MHz (1.6-ns period)! In the TRNG design, I used a nine-stage oscillator, which runs at 210 MHz (4.76-ns period).

TRNG ARCHITECTURE

After a ring oscillator is sampled, you need to wait for several clock periods before the clock phase is random again. The sample interval may be shortened by combining multiple ring oscillators. Figure 3 shows how this can be accomplished. That design uses three ring oscillators. Each oscillator output is periodically sampled in a flip-flop. Due to drift and process variations, each ring oscillator's phase is different. Therefore, the bits sampled from the oscillator outputs using a fixed clock can be randomly distributed over the ring oscillator period. Given enough oscillator rings, there's a good chance that at least one of the ring oscillators is within the jitter window when its output is sampled. The TRNG's output is computed by XORing all sampled bits together. Therefore, the random bit will propagate to the output. This design has several parameters: the number of rings, the number of inverters per ring, and the frequency of the sample flip-flop, with respect to the oscillation frequency. This design's exact analysis is complex, but it's clear that a larger number of ring oscillators will generate more true randomness.

What if an oscillator violates the sample flip-flop's setup time? Indeed, this is a possibility, since the ring oscillator output is asynchronous with respect to the sample clock. However, for the purpose of building a TRNG, this will not hurt the operation. The worst that can happen is a metastable flip-flop, which will inject additional randomness in the TRNG output.

My design is implemented in an Altera Cyclone IV EP4CE22F17C6N FPGA. To determine a reasonable value for the design configuration, I implemented and measured ring oscillators of various lengths in an FPGA (see Figure 4). For the final design, I'm using a nine-stage oscillator, which oscillates at 200 MHz. The sample flip-flops run at 50 MHz.

RANDOMNESS TESTING

To test the a TRNG's quality, the output bitstream is fed into a random-number testing suite, which works by analyzing a large number of output bits and judging their randomness by evaluating several properties. For example, you could count the number of "1" bits in a long sequence and require the resulting count to fall within a certain limit of the expected value, half of the sequence length. A random-number test can be done offline, as a post-processing, or online while the TRNG is running. In this case, I will build an online random number test. This will raise an alert as soon as something is going wrong with the TRNG—something that is relevant when security (or casino revenue) is at stake.

Two popular test suites are provided by the National Institute for Standards and Technology (NIST) and the Marsaglia Random Number CDROM, which was created by George Marsaglia of Florida State University. However, their complexity makes them less suited for an online implementation, integrated into an embedded system. Instead, I will make use of an older version of the NIST testing suite, FIPS 140-1, which was first proposed in 1994. While it is not as thorough as NIST or Marsaglia, it is compact

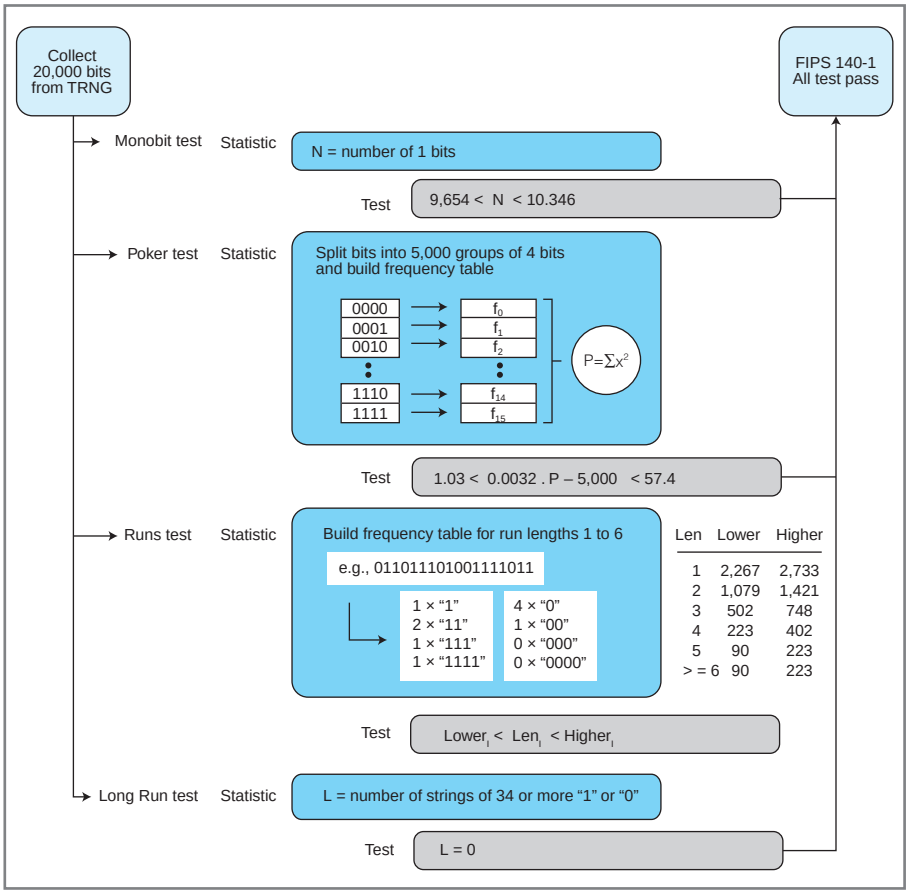


Figure 5—The FIPS 140-1 standard specifies four different tests (monobit, poker, runs, and long run) on a set of 20,000 bits. To be FIPS 140-1-compliant, a TRNG must pass all four tests.



▶ CUWIN6300

▶ CUWIN5500

▶ CUWIN5200

Windows CE 6.0 Touch Controller

CUWIN

CE FC

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

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



C-Programmable Modular Industrial Controller

MOACON

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

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

BASIC with LADDER LOGIC CONTROLLER



New

Integrated CUBLOC Controller and I/O Board

CB210

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

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

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

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

CUPC

CE FC

Industrial Touch Panel PC with ATOM Processor



▶ CUPC-P200

▶ CUPC-P220

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

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

CUSB

Integrated Industrial Controllers



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

enough to be implemented in hardware, together with the TRNG ring oscillator.

Figure 5 shows the test suite's concept. The test collects 20,000 bits from the TRNG output and performs four different tests. The Monobit test evaluates the bias in the bitstream. There should be, on average, an equal number of 1s and 0s in the output, so there should be approximately 10,000 "1" bits among 20,000 output bits. The Poker test splits the bits in 5,000 groups of 4-bit patterns and counts the number of times each 4-bit pattern appears (e.g., "0000," "0001," etc.). Naturally, each pattern should be equally likely, so the average variation among the pattern counts should be small.

The Runs test evaluates the number of times each repeating bit pattern appears (e.g., "0," "1," "00," "11," "000," "111," etc.). Short bit patterns should be more likely than long bit patterns. Finally, the Long Run test evaluates if there are long strings of repeating bits. This test ensures there are no strings of more than 34 identical bits in any 20,000-bit block.

TRNG WITH BUILT-IN TEST

The resulting TRNG, together with its tests, can be integrated into a single module (see Figure 6). The TRNG with ring oscillators is at the module's core. The module's output is downsampled and fed into a 32-bit shift register. The design is parameterizable. I used 20 oscillator rings with nine inverters per ring, and I downsampled the result to a 50-kbps stream. Hence, the 32-bit random register is refreshed 1,562.5 times per second. The random bitstream is also processed in a hardware implementation of the FIPS 140-1 test. Each test is implemented as two modules. One module derives the test statistic (e.g., the number of "1" bits in a block of 20,000 bits). The second module performs the actual check (e.g., checking if the 1s count falls within FIPS 140-1 limits). The result of the statistical test can be captured in 4 bits, all of which should be 1. The module is configured to repeat the test for each 65,536 bits produced.

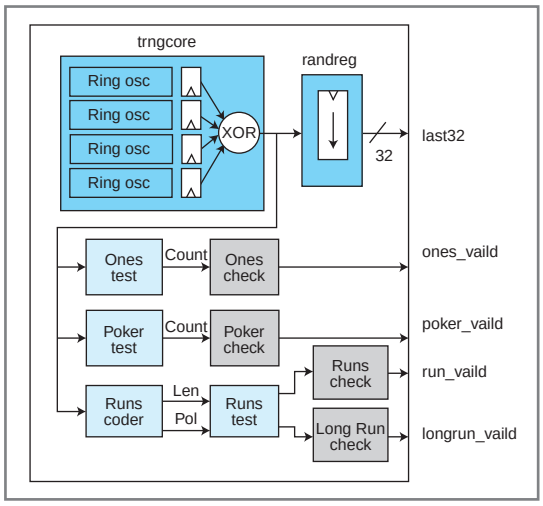


Figure 6—This is the TRNG's architecture with a built-in FIPS 140-1 test. A TRNG core with a parametrizable number of oscillators and sample frequency creates a random bitstream. A shift register chops the output in blocks of 32 output bits. The bitstream is also analyzed by hardware modules that implement the four tests included in FIPS 140-1: monobit, poker, runs, and long run. Four output flags return the test result. The test automatically repeats for every 64-Kb output bits produced.

Listing 1—This is the Verilog code for the TRNG module. Special synthesis directives ensure the ring oscillators are not removed by the synthesis tools.

```
1. module inv_cell(input wire  a,
2.                 output wire q);
3.    wire         aw;
4.    lut_input  lut_a(a, aw);
5.    lut_output lut_q(~aw, q);
6. endmodule // inv_cell
7.
8. module nand_cell(input wire  a,
9.                  input wire  b,
10.                 output wire q);
11.   wire         aw;
12.   wire         bw;
13.   lut_input    lut_a(a, aw);
14.   lut_input    lut_b(b, bw);
15.   lut_output lut_q(~aw | ~bw, q);
16. endmodule // nand_cell
17.
18. module ro(input wire en,
19.               output wire q);
20.   parameter STAGES = 3;
21.   wire [0:(STAGES-2)] ni;
22.   wire                nandout  /* synthesis keep = 1 */;
23.   wire [0:(STAGES-2)] no  /* synthesis keep = 1 */;
24.   nand_cell nc(en, no[(STAGES-2)], nandout);
25.   inv_cell  ic[0:(STAGES-2)] (ni, no);
26.   assign ni[0]             = nandout;
27.   assign ni[1:(STAGES-2)] = no[0:(STAGES-3)];
28.   assign q            = no[(STAGES-3)];
29. endmodule
30.
31. module rngcell(input  wire clk,
32.               output wire qw);
33.   reg          q;
34.   parameter STAGES = 9;
35.   wire         ro_en, ro_q;
36.   ro #(STAGES) myro(ro_en, ro_q);
37.   assign ro_en = 1'b1;
38.   always @(posedge clk)
39.      q <= ro_q;
40.   assign qw = q;
41. endmodule
42.
43. module trngcore(input wire clk,
44.               output reg q);
45.   parameter CELLS = 20;
46.   parameter STAGES = 9;
47.   wire [0:(CELLS-1)] tq;
48.   rngcell #(STAGES) t[CELLS] (clk, tq);
49.   always @(posedge clk)
50.      q <= ^tq;
51. endmodule
```

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

HUMANDATA® FPGA / CPLD Boards from JAPAN

SAVING COST=TIME with readily available FPGA boards

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

ALTERA FPGA Board

Cyclone IV E F780 FPGA board

ACM-204 series NEW

Cyclone IV E | SDRAM

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

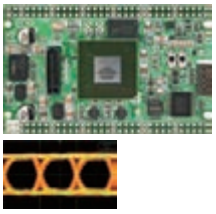


Arria II GX F572 FPGA board

ACM-025 series

Arria II GX | DDR2

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



CycloneIV GX F484 FPGA board

ACM-024 series

Cyclone IV GX | DDR2 | SIF40

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



Cyclone IV E F484 FPGA board

ACM-107 series

Cyclone IV E | MRAM

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



XILINX FPGA Board

Spartan-6 FGG484 FPGA board

XCM-018/018Z series

Spartan-6 | MRAM | DDR2

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



Spartan-6 FGG484 FPGA board

XCM-110/110Z series

Spartan-6 | MRAM | DDR2

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



Virtex-5 FFG676 FPGA board

XCM-011 series

Virtex-5 | FRAM | SDRAM

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



Virtex-5 LXT FFG665 FPGA board

XCM-017 series

Virtex-5 | SDRAM | RocketIO | SIF40

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



FPGA/CPLD Stamp Module PLCC68 Series

Easy and Quickly Mountable Module

FPGA Module IC socket mountable



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



XILINX PLCC68 Series

Spartan-6 PLCC68 FPGA Module

XP68-03



NEW | Spartan-6 | PLCC 68

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

Spartan-3AN PLCC68 FPGA Module

XP68-02



Spartan-3AN | PLCC 68

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

ALTERA PLCC68 Series

Cyclone III PLCC68 FPGA Module

AP68-04



NEW | Cyclone III | PLCC 68

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

Cyclone III PLCC68 FPGA Module

AP68-03



Cyclone III | PLCC 68

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

MAX V PLCC68 CPLD Module

AP68-02



MAX V | PLCC 68

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

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

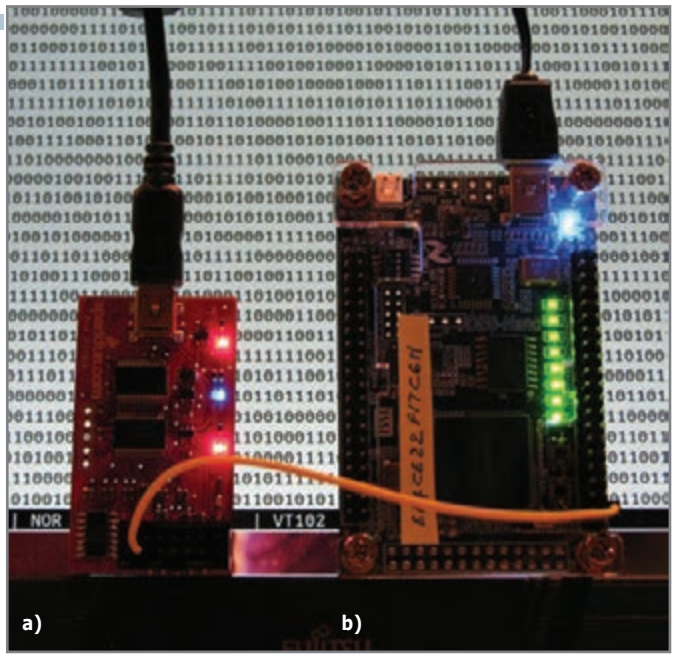


Photo1a—The bitstream is available in ASCII format over a UART connection, implemented with a Bus Pirate module. The terminal in the back shows the bitstream. **b—**The DE0-Nano is running the random number generator. The lower four LEDs indicate the results of the FIPS 140-1 tests. The upper four LEDs show the lower nibble to the random register.

Listing 1 shows a Verilog implementation of the TRNG core. Of particular interest in this design is the ring oscillators' implementation. Asynchronous structures (e.g., rings of inverters) tend to confuse the synthesis tools. The oscillators are, therefore, implemented from the bottom up. The oscillator's inverters are captured as low-level FPGA primitives (see lines 1–16). These modules are then assembled into a ring (see lines 18–29). The directives on lines 22 and 23 (`/* synthesis keep = 1 */`) tell the hardware synthesis tools to implement the design precisely as shown. Several ring oscillators outputs are XORed together to produce a random bit (see lines 31–51). There are flip-flops at each oscillator's output, and at the output of the XOR tree. This prevents high-frequency switching within the XOR tree.

The random tests (e.g., monobit, poker, runs, and long run) are implemented as independent logic modules. Most of them involve simple bit counting and range checking. Perhaps the most complex one, in terms of hardware resources, is the Poker test. Given 16 frequency counts f_i, one for each bit pattern from 0000 to 1111, the Poker test evaluates:

$$1.03 < \frac{16}{5{,}000} \times \sum f_i^2 - 5{,}000 < 57.4$$

The test statistic evaluation requires complex and expensive floating-point computations. This can be easily avoided by scaling the formula with a 5,000 factor:

$$5{,}120 < 16 \left\{ \sum f_i^2 - 1{,}562{,}500 \right\} < 287{,}000$$

Furthermore, instead of computing all f_i^2 in parallel (i.e., 16 hardware multipliers), I can use a single multiplier to compute them in 16 clock cycles. For the Verilog implementation of this module, refer to the source code, which is available on *Circuit Cellar*'s FTP site.

An interesting point in the random number test implementation involves debugging and simulation. Obviously, when Listing 1 (the TRNG core) is loaded into a Verilog simulator, you will not get a random bitstream. Indeed, the Verilog simulator does not simulate jitter! This makes it difficult to verify the proper operation of the random number tests. To handle this issue, I replaced the TRNG core in the simulation with a PRNG, which can be simulated in a Verilog simulator. In addition, the output of a good PRNG will pass FIPS 140-1, so it can be used to debug the random number testing hardware.

The prototype is implemented on the same Terasic DE0-Nano kit platform I used in my article "Hardware-Accelerated Encryption" (*Circuit Cellar* 266, 2012). The design can be synthesized using Quartus (10.1 or later). I integrated the TRNG module in the FPGA, together with a UART, to send ASCII-format bits from the random register to a terminal. The output pins from the random number tests are wired to a few LEDs (see Photo 1).

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

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/268.

RESOURCES

G. Marsaglia, "The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness," Florida State University, 1995, http://stat.fsu.edu/pub/diehard.

National Institute of Standards and Technology (NIST), "FIPS 140-1: Federal Information Processing Standards, Security Requirements for Cryptographic Modules," 1994, http://csrc.nist.gov/publications/fips/fips1401.htm.

———, Computer Security Resource Center (CSRC), "Guide to the Statistical Tests," 2000, http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html.

RANDOM.ORG, www.random.org.

P. Schaumont, "Hardware-Accelerated Encryption" (*Circuit Cellar* 266, 2012

SOURCES

Cyclone IV EP4CE22F17C6N FPGA
Altera Corp. | www.altera.com

Bus Pirate open hardware tool
SparkFun Electronics | www.sparkfun.com/products/9544

DE0-Nano Development and Education Board
Terasic Technologies, Inc. | www.terasic.com.tw

2013 International CES
THE GLOBAL STAGE FOR INNOVATION

PRODUCED BY CEA

Be the first to see what tomorrow will be talking about.



TUESDAY, JANUARY 8 – FRIDAY, JANUARY 11, 2013 * LAS VEGAS, NEVADA * REGISTER AT CESWEB.ORG



REGISTER NOW

by George Novacek (Canada)



Product Reliability (Part 1)

Reliability Prediction

Reliability, maintainability, and safety (RM&S) are important activities during product development. This article focuses on the critical task of reliability prediction. Without reliability prediction, the other RM&S activities cannot be performed.

In my next few articles, I will discuss how to ensure product reliability and safety. This important engineering activity is largely based on statistical analysis and must be understood as such. That means while a certain failure's statistical probability may be infinitesimally small, there is no guarantee the failure won't occur in the next moment. This is especially important for the customer to understand. One product's early failure does not mean the contract terms have not been met.

Most safety analyses are based on the knowledge of the components' failure rate. Thus, reliability prediction is a good start for this series.

BATHTUB CURVE

Figure 1 is a well-known "bathtub curve" showing a product's failure probability over its lifetime. In this article, I focus on stress-related failures, which are statistical probabilities that are constant during a product's lifetime (see the green line in Figure 1). Initial quality failures (i.e., infant mortality) should be eliminated by environmental stress screening (ESS) before the product ships. I described this in my article, "Environmental Stress Screening" (*Circuit Cellar* 255, 2011).

Wear-out failures are mainly considered a problem with mechanical and electromechanical parts. But there are solid-state devices (e.g., power semiconductors) that exhibit limited life due to wear out caused by physical and thermal stresses. Wear-out time is difficult to predict. It depends on your design experience. Wear out must never occur during the product's expected lifetime.

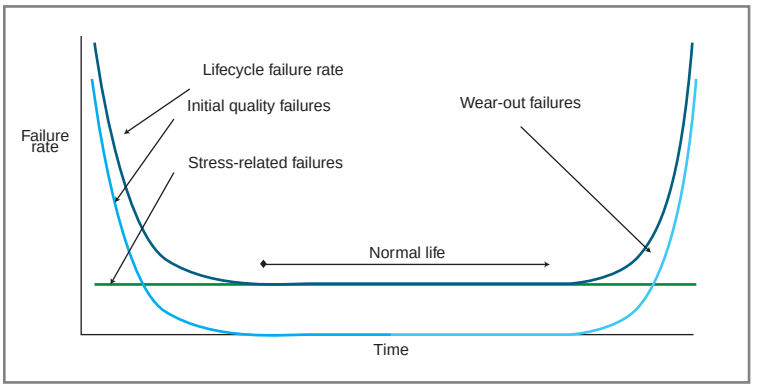


Figure 1—This "bathtub curve" shows reliability during a product's lifetime.

SOFTWARE RELIABILITY

Software is not subject to stress, does not wear out, and does not age with use. Software execution faults due to parts failures or external effects, such as electromagnetic interference (EMI), are hardware related. Hardware faults are physical faults. Software faults are design faults. Some measure of a product's reliability—which includes software—is needed, but it's like comparing apples to oranges.

While there are methods to calculate software reliability failure rates so they can be combined with the traditional hardware failure rates, I am not aware of any universally accepted method. For the moment, I'll only concentrate on hardware reliability prediction. I will address software reliability in a separate article.

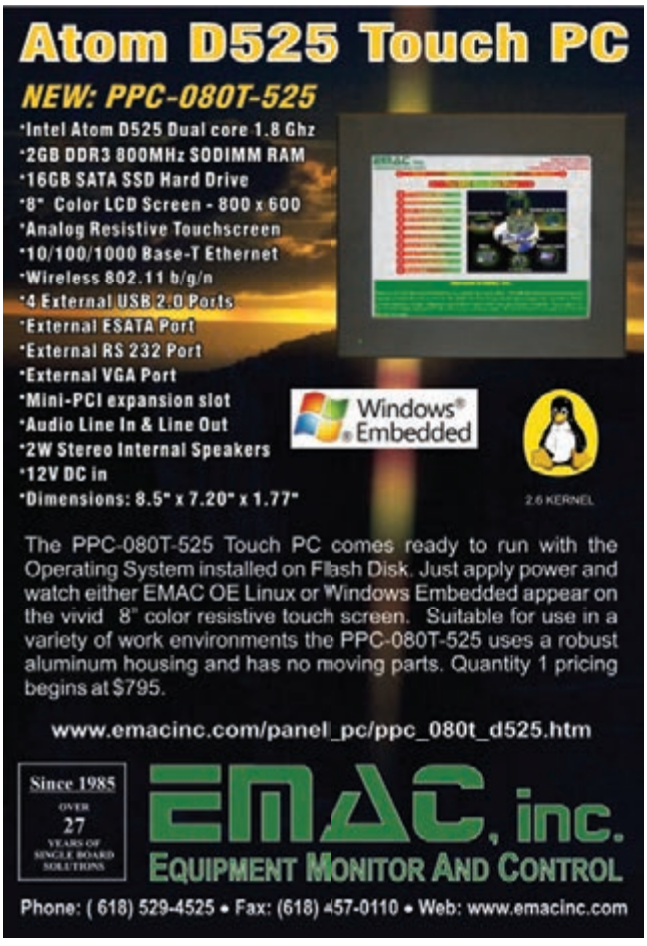
MATHEMATICAL MODELS

There are various mathematical models and tools to calculate electronic products' reliability (e.g., MIL-HDBK-217, Bellcore, Telcordia, etc.). The MIL-HDBK-217 component count model is popular worldwide. The U.S. military developed it in the 1950s. The military handbook MIL-HDBK-217 describes the process, which can be downloaded from numerous sources (e.g., Quanterion Solutions) for free. The latest revision, F, was originally released in 1991 and last updated in 1995. The MIL-HDBK-217 reliability prediction calculations are mathematically simple, but don't let that mislead you. There is a lot of research behind those formulas. Reliability modeling is a science in its own right.

Over time, MIL-HDBK-217 became somewhat outdated, as it was developed during the early days of electronic components manufacturing. Manufacturing processes and quality controls were not on the level we are used to seeing today. Components to achieve a military level of reliability had to be screened and, therefore, were expensive. The same commercial-rated component with a tenfold price was not an exception.

By the end of the 1980s, manufacturing processes became so consistent, in many cases screening became unnecessary. Following the Perry Initiative in 1994, use of commercial-off-the-shelf (COTS) components began in established reliability equipment. Following the MIL-HDBK-217, guidelines to calculate failure rates for COTS components resulted in unrealistically low reliability estimates. The Reliability Information Analysis Center (RIAC) RIAC-HDBK-217Plus handbook, which was published in 2006, addressed this discrepancy. The handbook costs approximately $1,400; however, you can use the free MIL-HDBK-217F and adjust the quality factors λ_Q to a level closer to reality. With experience, you will develop a pretty good feel for it. Better yet, download SoHaR's free MTBF calculator.

Reliability prediction calculation is straightforward, but it is a menial, time-consuming task if it is manually performed. There are programs to automate the process, such as Parametric Technology's Windchill (formerly known as Relex), ReliaSoft, SoHaR, T-Cubed, Advanced Logistics Development (ALD), and so forth. These programs are expensive. SoHaR offers a suite to





generate reliability, maintainability, safety, and other related analyses. ALD also offers a free program to calculate reliability, which includes a database of many components. It enables you to calculate reliability using different models and to select different environments. The program can be downloaded from ALD's website. I tried it and recommend using it for small projects. It helps you get the feel for reliability prediction before you spend a lot on a full-featured program.

TOTAL FAILURE RATE

As I mentioned, the calculations are straightforward. To better understand it, I will explain the process commercial programs hide from you. For every component, determine its failure rate λ, then add all the lambdas together to obtain the total failure rate, which reflects the expected number of failures per 1 million hours:

$$\lambda_{PRODUCT} = \left(n_{R1} \times \lambda_{RESISTOR\ 1\ k\Omega}\right) + \left(n_{R2} \times \lambda_{RESISTOR\ 2.2\ k\Omega}\right) + \left(n_{C} \times \lambda_{CAPACITOR\ 0.1\ \mu F}\right) + \text{etc.}$$

Here, n is the number of the same components under the same stress. Reliability programs contain databases of parts' failure rates, which make the calculations much quicker. Equipment manufacturers should collect their reliability data and maintain their own databases. You should be using as many identical components under the same stress as possible. For example, all resistors should be derated by 50%. Components with existing historical data in the company's database should be preferable. Be careful when choosing components without established pedigree or experience.

When historical data does not exist in the available reliability database, the component's failure rate can be calculated. Following the "217Plus" handbook, a resistor failure rate is:

$$\lambda_{RESISTOR} = \pi_G \left(\lambda_{OB}\pi_{DC0}\pi_{T0}\pi_P + \lambda_{EB}\pi_{DCN}\pi_{TE} + \lambda_{TCB}\pi_{CR}\pi_{DT}\right) + \lambda_{SJB}\ \pi_{SJDT} + \lambda_{IND}$$

All electronic components have similar empirical formulas derived from tests. The terms are calculated with equations presented in the handbook, while the constants are listed in tables therein. Here, for example, they reflect the resistor type, its rating, power dissipation, operating duty cycle, operating temperature, cycling rate, and so forth. All you really need is a hand calculator, or you can set up the equations in a spreadsheet. You can imagine how confusing the manual calculations can become with many different components working at different stress levels. Instead, download SoHaR's MTBF calculator.

THE NUMBERS EXPLAINED

At the end of your effort, whether it's manual or computer assisted, you will have the product's predicted failure rate, λ. That is the statistical probability of a failure over 1 million hours. You may be more familiar with its reciprocal, mean time between failures (MTBF). In the next article in this series, I'll explain the numbers and how they can be used.

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

AP CIRCUITS
PCB Fabrication Since 1984

As low as...
$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB

RESOURCES

Advanced Logistics Development, "Free MTBF Calculator," www.aldservice.com/en/reliability-software/free-mtbf-calculator.html.

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

Quanterion Solutions, Inc., "MIL-HDBK-217F: What is MIL-HDBK-217F?," http://quanterion.com/Publications/MIL-HDBK-217/index.asp.

The Reliability Information Analysis Center (RIAC), "217Plus: RIAC's Reliability Prediction Methodology," www.theriac.org/productsandservices/products/217plus.

ReliaSoft Corp., www.reliasoft.com.

SoHaR, Inc., "Free MTBF Calculator," www.sohar.com/reliability-software/free_mtbf.html.

T-Cubed Systems, Inc., www.t-cubed.com.

P. A. Tobias and D. C. Trindade, *Applied Reliability*, CRC Press, 1995.

SOURCE

Windchill
Parametric Technology, Inc. | www.ptc.com

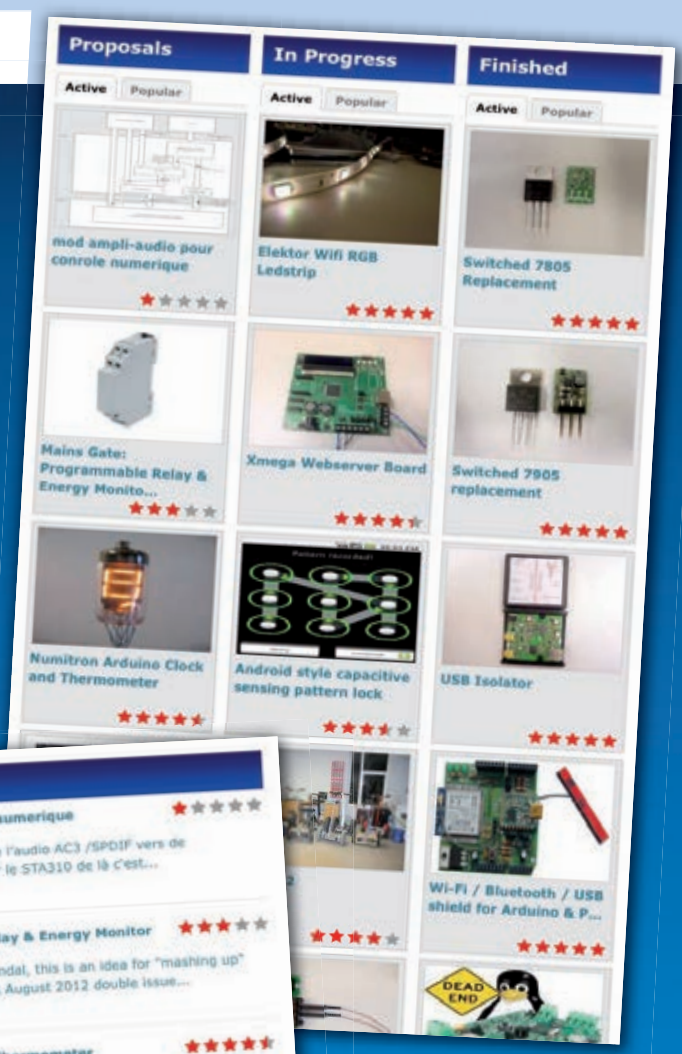
www.elektor-projects.com

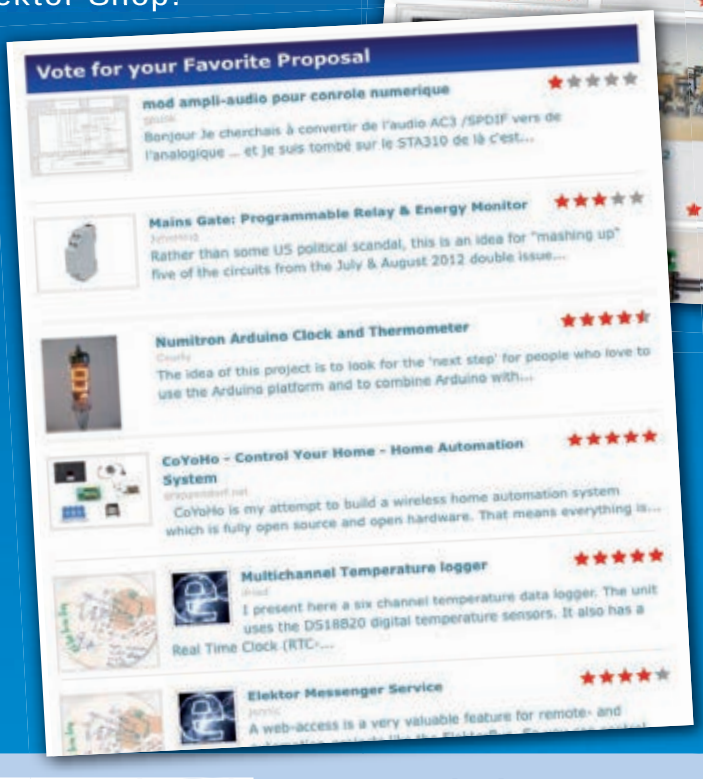
Get Elektorized

Sharing Electronics Projects

Elektor Projects is an online community for people passionate about electronics. Here you can share your projects and participate in those created by others. It's a place where you can discuss project development and electronics.

Elektor's team of editors and engineers assist you to bring your projects to a good end. They can help you write an article to be published in Elektor magazine or even develop a complete product that you can sell in the Elektor Shop!





JOIN NOW!

elektor PROJECTS

Get elektorized too! Check www.elektor-projects.com

by George Martin (USA)



Tachometer Design

A tachometer is a handy device if you're interested in machining parts with a lathe and a mill. It enables you to determine revolutions per minute (RPM) and better manage cutting speeds. This article details the process of building a tachometer and provides tips on choosing proper sensors, incorporating a microcontroller, developing software, and more.

I am in the process of winding down my electronics design career and starting up a new endeavor. In my spare time, thinking about electronics in general, and *Circuit Cellar* in detail, I see a rapidly changing picture. It's easy to choose off-the-shelf components (both hardware and software) to create a product. Many sources discuss potential products, but *Circuit Cellar* actually shows you how to accomplish the design. As contributors, our task is to present the details in an interesting manner. I hope we're doing a good job.

I need a few custom instruments. My new business is very mechanically oriented. I'm ordering small parts, machining some items, and molding others. I'm applying the processes I've described in many of my Circuit Cellar articles.

As my startup progresses, I've begun receiving some machinery in house, including a mini mill and a mini lathe. I use this machinery to support customer changes to standard products. Since I have no experience or training using these machines, I'm anticipating a steep learning curve. For example, last month, I could shape aluminum rods quite well. This month, all I get is chatter. Something is different, to say the least. And that brings me to my need for an instrument to help me understand what's happening.

The mini mill and a mini lathe are small machines controlled by variable-speed DC motors with forward and reverse capabilities. When machining different materials, the amount of material removed in a set amount of time becomes the basis of your cost and is limited by the physical properties of the tools and materials you are machining. The key controlling parameters become the cutter's feed rate and the material's speed (i.e., "feed and speed"). The material's speed is proportional to the material's diameter for rotational cutting or the cutting tool's diameter for a milling-type operation. Knowing the rotating speed is the basis for all your other decisions, such as cutting feed rate, tool type (e.g., high-speed steel or carbide), and cooling type (e.g., none, air, or fluid).

WHAT IS A TACHOMETER?

Both of my machines have reversible DC motors with variable speed controls. As an initial starting point, I would like to know the revolutions per minute, or RPM (i.e., speed), produced. That way, whenever I get results, I can compare my settings to the recommended settings and repeat them. Also, from this reading, I can determine the cutting speed with which I'm actually operating, thus my need for a tachometer.

A tachometer measures and reports rotation speed. To keep the tachometer's setup simple, put a piece of tape on the rotating shaft for an index marker and use an optical pickup to obtain one pulse per revolution. The tape can be either reflective or absorptive, either changes an optical sensor. Measure the time between pulses and convert that to RPM, then send that reading to a four-character

digital LCD. My tachometer will have a 0-to-2,000-RPM starting range. I could use a 9-V battery and have an On/Off switch, a magnetic mount, an off-the-shelf plastic case, and some sort of gooseneck device to get the sensor close to the rotating shaft. Ta-da! I would have product and a job well done! Oh I forgot, I need to explain how to do all this.

I searched online to ensure this product is not available off the shelf. I searched for tachometers and found many mechanical devices for automobiles. They have interesting-looking displays, but nothing was close to what I wanted. Refining my search to optical tachometers, I found some optical tachometers close to my requirements and ordered one ($60) for further evaluation. While it may be close to what I want, I'll probably need other features that are not included.

> "A tachometer measures and reports rotation speed. To keep the tachometer's setup simple, put a piece of tape on the rotating shaft for an index marker and use an optical pickup to obtain one pulse per revolution. The tape can be either reflective or absorptive, either changes an optical sensor. Measure the time between pulses and convert that to RPM, then send that reading to a four-character digital LCD. "

DIFFERENT SENSORS

The rotation measuring sensor will be the tachometer's input side. There are several choices, from magnetic to optical. To keep it simple, versatile, and easy to move among different machines, I selected an optical-type sensor. More specifically, I selected an optical sensor with a transmitter and receiver that operates in the infrared (IR) spectrum. With IR, the tachometer will be less influenced by stray light and I can more easily use a piece of tape to generate an index mark once per revolution.

I excluded all forms of magnetic sensors. These all require close positioning between the sensor and the magnet. I've discovered that as the speed is reduced, the input signal changes shape. At high RPMs, these types of transducers resemble square waves. At low RPMs, they look more like a sine waves. It's more difficult to find a sine wave's edge than a square wave's edge. I would like a square wave at the input to precisely detect when the measurement device passes the sensor.

In my search, I discovered a family of solutions that uses stroboscopic light or stroboscopic shutters to measure RPM. These devices operate their shutters at a user-set rate. If the rotating device appears stationary, you are operating the shutter at a multiple of the rotating device's speed. And, if that multiple is one, you are at the device's RPM. These types of devices look interesting but require too much operator interaction for my requirements.

SYSTEM DESIGN

Figure 1 is a block diagram of the system including a power supply, display, CPU, sensor, and an On/Off switch. As I select components, build software interfaces, and package everything, I plan to divide the work into several sections. The first section is basic operation (i.e., generic operation). These groups of decisions and interfaces are basic. Decisions about the second section or group will be specific to the components I've selected

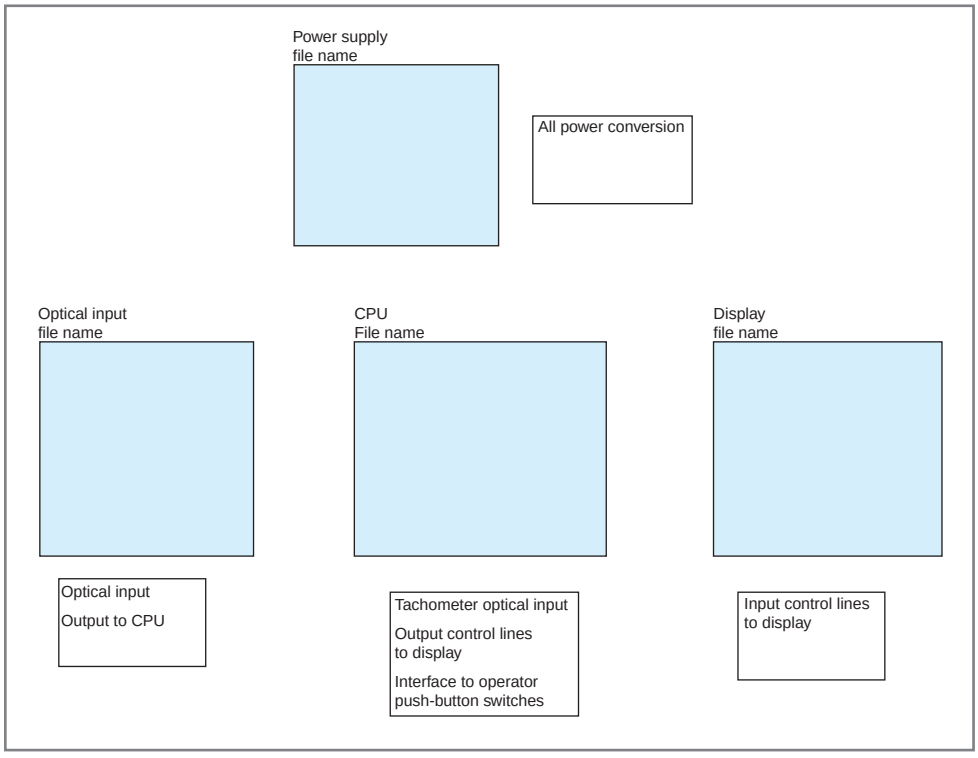


Figure 1—This tachometer block diagram includes a power supply, display, CPU, sensor, and an On/Off switch.

RPM Analysis						
			0.1	Jitter (ms)		
RPM	Rate (s)	Rate (ms)	Jitter+	Jitter–	RPM+	RPM–
10	6	6,000	6000.1	5999.9	10	10
60	1	1,000	1000.1	999.9	59.99	60.01
100	0.6	600	600.1	599.9	99.98	100.02
600	0.1	100	100.1	99.9	599.4	600.6
1,000	0.06	60	60.1	59.9	998.34	1001.67
2,000	0.03	30	30.1	29.9	1993.36	2006.69

Table 1—Introducing a 100-µs jitter (i.e., noise) has a different impact on several RPM readings.

for this design. If, for example, I select a different display, I would only need to modify this second hardware and software section. This type of design should be done for all products. What if a device becomes obsolete and needs to be replaced? With the design properly partitioned, the work would be minimized and you would be a star! It's good to have goals. With this division in mind, I'll explore the components.

The CPU and the main power supply will need to be either 3.3 or 5 V. Designing a 3.3-V system will result in lower power operation. But, I will probably need to power the display and sensor with higher voltages to the contrast and signal-to-noise that will make for an acceptable unit. So, I will use a 5-V basic system voltage. (Remind me to revisit this decision down the road to see if it was well made.)

DISPLAY

My perfect display would be a large four-digit LCD. One-inch characters would be great, similar to a digital multimeter (DMM) display. In a real production design, I would drive the segments (i.e., 4 digits × 7 segments = 28 total segments) directly displayed from the CPU. Some modern CPUs have the LCD segment drive built in, but I'm going to build one or two of these units. So, the number of wires is convincing me to choose a different display type. It's still an LCD, but it's a graphic-type display.

Newhaven Display International's NHD-12232AZ-FSW-GBW graphic LCD features 32 × 122 dots with a 16 mm × 53 mm (0.63" × 2") display size. With this unit, I could easily display four large digits. The LCD uses an 8-bit parallel bus with one address, one read/write, and two chip-select lines for control. That's a total of 12 control lines, which is not too many. I think even I could wire this prototype. And, it only costs $15.50. The LCD also features 5 VDC and 2 mA on the power side. During the design process, I need to consider backlighting issues, which goes hand in hand with the reflective or transflective display type.

CPU

It makes sense to use a microcontroller to measure the speed and operate the display. It doesn't make sense for me to decide on a microcontroller and force you to use it. So, let's just consider the CPU a component in the system and make it easy to isolate it and replace it in the design's hardware and software. First, I'll show you some simple math for RPM and see how noise might affect the information.

In Table 1, I listed several RPMs and calculated the pulse rate that would come to the microcontroller every revolution. Next, I introduced a 100-µs jitter (i.e., noise) to all the readings. As you can see, at 2,000 RPM, 100-µs jitter is disastrous to the signal, which is because 2,000 RPM is 30 ms per revolution, and 100 µs is a big part of the 30 ms. I want the last displayed digit to be accurate, so, the CPU must connect an input pin to a timer function. An input pin change generates an interrupt, and the timer's value is read. For example, 60 RPM produces a new RPM reading every second and 2,000 RPM produces a reading every 30 ms, which is quite a range. If the display is updated for every 60-RPM reading, it's probably a reasonable image. If you try to update the display every 30 ms at 2,000 RPM, it just becomes a blur. A constant update rate provides an opportunity to filter the input values. I'll leave the filtering details for later, but they will be built into the requirements.

POWER SUPPLY

The design's last major component is the power supply. There are several choices, including using an USB connector to provide 5 VDC. However, I'm not sure I'll always be able to connect a USB to a computer. Also, it could be a messy environment with cutting and cooling fluids everywhere, so I don't think the USB option will work. Next, I'll look at batteries. Standard 9-V batteries have 500-mAh capacitance. Burning 25 mA, I could achieve 20 h of operation with 50% efficient power supplies. If I were designing a buck-type switching regulator, I could achieve 85% efficiency, which would extend the operating time. I'm adding a lot of complexity for the battery operation. I could use a rechargeable Li-ion battery and a charger. They are reasonably priced and supplied by most electronics distributors. I'm also toying with the idea of using a cordless drill's battery pack. This project is not burning so many watts that I would need much power, but it might be nice to have that design in my bag, as I may want to take a different instrument into the field at some point.

DESIGN MILESTONE

I have a first pass at the design, which is a milestone. Perhaps it's not a milestone I could publish or invoice to a client. The project is small, so this milestone seems little. Next I could start drawing schematics that show more detail than my block diagram, but I think it's more important to look at the software. I need to read the datasheet on my display and see how I might design the code to interface to that particular device.

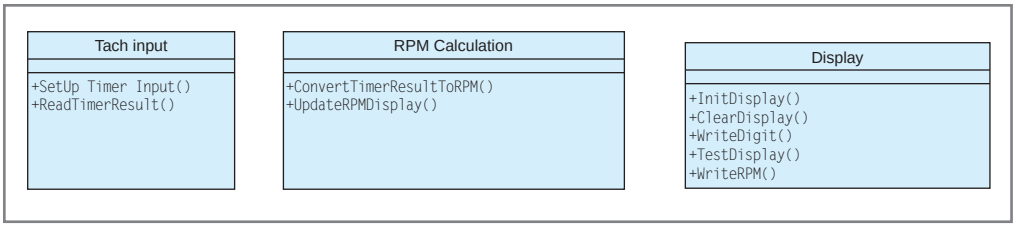


Figure 2—My initial understanding of the tachometer's code includes three main code modules.

CONNECT WITH

Circuit Cellar

For people who are passionate about hardware and software design, embedded development, and computer applications. Awesome projects, tutorials, industry news, design challenges, and more!



Connect.
Follow us on Twitter.
Like us on Facebook.

www.circuitcellar.com

@CircuitCellar
@editor_cc

circuitcellarmagazine

"The software is where the real design work starts. (It's about time!) In selecting the hardware components, all the details were buried in the device and many assumptions were made as to what would be done in software. With the software, I started my UML program, made a new project, and was surprised I didn't know what to do next. That's typical. Don't worry when you find yourself in the same situation."

And while I'm at it, exploring the software with respect to the RMP calculations might also uncover some surprises.

SOFTWARE

The software is where the real design work starts. (It's about time!) In selecting the hardware components, all the details were buried in the device and many assumptions were made as to what would be done in software. With the software, I started my UML program, made a new project, and was surprised I didn't know what to do next. That's typical. Don't worry when you find yourself in the same situation. I'll start with what I know.

I suspected I would need some code for the tachometer input, RPM calculations, and display control. I'll start with these three and see how the design unfolds. My initial understanding of the code is shown in Figure 2. There are three main code modules with several routines in each module. I promised to keep the hardware specific routines separate from the more general routines, so it will be straightforward to implement this design on my hardware. So the three code modules might split into more modules to support the separation of hardware specific routines.

The Tach Input module has two routines. The Setup Timer Input routine is hardware specific. I envision connecting the sensor pulse to a timer/counter input pin on the CPU. That pin generates an interrupt every time the index marker passes. A timer register will be read in the Read Timer Results routine and placed in a queue for later processing. An interrupt is generated, and the timer data is also saved in a queue for later processing.

The RPM calculation routines are interesting. The first identified routine converts the raw timer results to RPM units. Remember from Table 1, there is a wide range of readings and noise, which can cause great problems. I envision the Convert routine to just do a conversion and to get RPM readings in engineering units. The Update RPM Display routine needs to smooth the data to reduce the noise component and handle the update rate. If data is coming in at 2,000 RPM, I don't want to display every reading. The display would just be one great flickering mess. I should stop updating the RPM when it gets to slow. It is better to remove instead of displaying an outdated RPM reading. Everywhere in between, I need to make adjustments, as necessary. The RPM Calculation routines will need access to a real-time timer. The Update RPM Display is the result of all this work.

The display routines fall into two categories. The first category, housekeeping, includes the `InitDisplay`, `ClearDisp`, and `TestDsip` routines, which can be used to understand how the display operates. The `WriteRPM` and `WriteDigit` routines actually write the RPM data to the display.

I can imagine other routines (e.g., a `hello world` routine that displays your name, copyright information, and version numbers). But I'll save that for a later presentation. I'm off to order parts. You should round up your favorite CPU and get ready to implement the design.

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

REFERENCES

element14, "Freescale Freedom Development Platform," www.element14.com/community/community/knode/dev_platforms_kits/element14_dev_kits/kinetis_kl2_freedom_board?view=overview.

H. Smeitink, "TFT LCD," *HMS Projects*, 2012, www.hmsprojects.com/tft_lcd.html.

SOURCE

NHD-12232AZ-FSW-GBW Graphic LCD
Newhaven Display International | www.newhavendisplay.com

NEED-TO-KNOW INFO

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

Electronic Gear Control
Add Electronic Gears to a Metal Lathe
by John Dammeyer
***Circuit Cellar* 196, 2006**

John upgraded his metal lathe with a PIC18F2620-based gear control system. In this article, he explains how to design and write code for your own electronic gearbox. Topics: Microcontroller, Lathe, Electronics

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

Time is Running Out

to get your Special Edition 25th Anniversary Hat and Robot

The Entire *Circuit Cellar* Magazine Archive on a Limited-Edition 25th Anniversary USB drive!

CIRCUIT CELLAR 25 YEARS OF EMBEDDED INSIGHT



Here's what's included:

- Special anniversary USB drive or our traditional CC GOLD USB drive. ***Your choice!***
- PDFs of all *Circuit Cellar* magazine issues in print through date of purchase
- Article code
- Design Challenge projects*
- Two years of *Elektor* magazine issues in PDF format (2010–2011)
- Two years of *audioXpress* magazine issues in PDF format (2010–2011)
- A USB memory upgrade from 16 GB to a whopping 32 GB!
- *Circuit Cellar* 25th anniversary hat

Time is running out to receive the blue tin robot featured here!

Order Now at www.cc-webshop.com

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

by Jeff Bachiochi (USA)



Mechanical Gyroscope Replacement (Part 2)

Gravity and Acceleration

In the first part of this series, I introduced a quadcopter that featured a mechanical gyro as an input device to keep the flying platform level with the ground. Due to the gyro's limitations, it can become unstable and deliver faulty data to the vehicle's motor controller. This article covers the topics of gravity, acceleration, velocity, and bearing.

Gravity plays a large role in our lives, but we usually take it for granted. We accept the fact that we are being pulled toward the earth's center by the mutual attraction of these two masses. Since the majority of our experiences take place on the earth's surface, we've come to accept this as the force of 1 g (gravity). If you stood on a trap door's opening shaft extending through the earth's center, you'd begin to fall toward the center. In 1 s you would be traveling at a velocity of 32'/s. As long as you continued to fall, your velocity would continually increase (ignoring other forces) until you passed through the center, in which case you would begin to decelerate again. Eventually, you would slow down until you were again moving at a 0'/s velocity. Presumably, you would be on the opposite side of the earth, and if a precisely timed trap door closed beneath your feet, you could then step back onto solid ground.

While gravity's effect causes a 1-g force toward the earth's center, 1 g of force can be experienced in any direction. If you can jump up with a force greater than 1 g, you can counteract gravity's force and travel away from the earth's center. When driving around a corner, you can experience a 2-g sideways force. Drive an Indy car, and the side force can exceed 3 g. Bobsledders can experience 5 g, and astronauts can experience 10 g. Human bodies tend to dislike anything greater than 7 g.

For practical purposes, we stick with 1 g as gravitational force when measuring acceleration, even though this value changes as we move further from the earth's surface. No matter how far we are from the earth's center, earth's gravity will always affect us. For the most part (until we are truly exploring our universe), we can use earth's 1-g gravitational influence as the primary means of determining orientation (i.e., down). The accelerometer measures this linear force for us. With 3-D orientation, it is common to use three accelerometers, one oriented to each of the three dimensions (i.e., x, y, and z), to find the gravity vector. When you toss a ball into the air, it can freely rotate in any direction. All three axes must be measured to determine where "down" is at any point in time.

REPLACEMENT SPECIFICATIONS

In the first part of this article series, "Mechanical Gyroscope Replacement (Part 1): A Microelectromechanical Systems (MEMS) Solution," (*Circuit Cellar* 267, 2012), I introduced a quadcopter vehicle that uses a mechanical gyro as an input device to keep the flying platform level with the ground. The gyro's limitations can cause it to become unstable and present wildly false input to the vehicle's motor controller. This routinely causes unwelcome maneuvers, which frequently end in a crash. The mechanical gyro plugs into the motor controller

and is easily removed. This enables a solid-state replacement to be retrofitted as long as the interfacing signals can be duplicated. The present mechanical system provides 5-V power through this interface and returns two voltages proportional to the gyro's x and y positions. The gyro is aligned with the vehicle with the x axis indicating roll, and the y axis indicating pitch. When the gyro is level, each output is centered at 2.5 V. These outputs vary a maximum of ±1 V with a maximum 30° tilt.

I'm not concerned with this vehicle's makeup. It could be a balancing robot, just as easily as a quadcopter. If I can mimic the interface's function and eliminate any bad side effects, then the solid-state solution can be safely implemented. The mechanical gyro is a two-axis (i.e., x and y) device. If I don't care which direction the vehicle is pointed (yaw), the third (z) axis is unnecessary.

While there may be cost savings associated with using a two-axis device, a three-axis device offers much more to the mix, especially if the circuitry can be used for other projects. For this reason, I am using a Pololu MinIMU-9, which has a three-axis rate gyro, accelerometer, and magnetometer all accessed via an I²C interface (see Photo 1).

Allow me to eliminate some confusion often associated with the term "gyro." A gyroscope's output is its position or angle. Most solid-state devices listed as "gyros" are not gyroscopes. These MEMS devices output a rate of rotation (ROR) that is the instantaneous angular rate at any given point in time (not actual position).They can be used to estimate position by keeping track of the ROR over some period of time. This is similar to accelerometers that indicate an amount of linear force (acceleration) applied at any point in time. This output can be used to estimate linear velocity by keeping track of acceleration over time. The magnetometer is the only device that directly outputs position because it directly measures the earth's magnetic field.

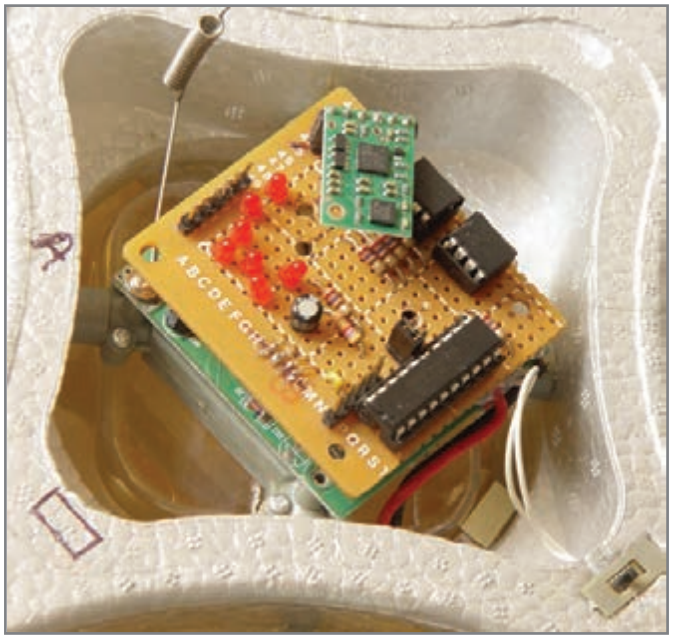


Photo 1—The Pololu MinIMU-9's sensor axes are aligned with the mechanical gyro so the x and y output pitch and roll, respectively. The prototype PCB interfaces with the quadcopter through a two-row by five-column, 1-mm² pin header. The microcontroller handles reading the sensors and calculating the appropriate output for the two DACs.

(Refer to Part 1 of this article series for more information on these MEMs devices.)

From my opening remarks, I hope you've concluded that the accelerometer is the best device to use to replace the mechanical gyro. Its ability to determine the gravity vector's direction makes it useful for determining the x- and y-axes' tilt levels. I'll be developing a couple of voltage outputs for this solid-state gyroscope replacement, but first I will explain the MinIMU-9's sensor outputs. I added a UART connection to the circuitry in the first part of this project, which I will use as a pop-in replacement for the mechanical gyro. This is not necessary for the project, but it enables the sensor outputs to be analyzed by running an external application on my PC. This will enable me to more easily visualize each sensor's value. I can do calculations and change algorithms more easily in BASIC than continuously rewriting microcontroller code.

Thus far, the microcontroller code initializes the sensors and continuously reads each sensor's outputs via I²C routines and sends them out to the UART in `!GYRO:X,Y,Z`, `!ACC:X,Y,Z`, and `!MAG:X,Y,Z` format at a 10-Hz rate.

GRAVITY & ACCELERATION

Whether you're trying to sense a single movement, put your position in perspective, or figure out how to get from here to there, it all begins with the same basic sensors. Each sensor adds information to help accomplish a task. An output equaling −1 g on the z axis and 0 g on both the x and y axis is expected while at rest and perfectly positioned on the gravity vector. Consider "down" (i.e., below level) as any negative z output and "up" (i.e., above level) as any positive z output. So, at rest, the gravity vector is as it should be, indicating straight downward. At rest, the x axis is level (i.e., 0° pitch angle, 90° to the z-axis). At rest, the y axis is level (i.e., 0° roll angle, also 90° to the z-axis). Any misalignment of the sensor to the gravitational vector and the outputs will share that 1 g together in proportion to their angular relationships. You can calibrate any misalignment by taking a reading while at rest (i.e., no movement and level). Any difference between the expected axis output and the actual axis output can then be added to any future readings creating a corrected output.

Once calibrated, if the sensor is rotated along the y axis (i.e., pitch) up or down, the gravity vector will be shared between the x and z axes. The new value output by the z axis will use arcsin to yield the pitch angle. Equally, should the sensor rotate on the w axis (i.e., roll, wingtip, up, or down), the gravity vector will be shared between the y and z axes. So, at rest, the accelerometer can use the gravity vector to determine pitch and roll. Gravity is always attempting to accelerate the sensor toward the earth's center. The earth's surface counteracts the gravitational force while objects are in contact with it. For an object to rise, some external force must replace that of the earth's surface. A 1-g thrust provides this. Any more than 1 g, and the object begins to accelerate upward; any less and it accelerates downward (assuming it is not already on the surface). From the accelerometer's point of view, this will be added to or subtracted from the 1 g it already sees due to gravity.

I expect the accelerometer will always measure 1 g unless it is accelerating/decelerating. Unless I pay attention to past input,

I can't determine which measurement is caused by gravity and which is caused by acceleration. As long as any changes take longer than my sampling interval, I should be able to predict which is which. Therefore, it is important to begin from a known place (in this case, a level and stationary position). I could just assume the sensor is level and will output 1 g on the z axis or run a calibration routine. Suppose you wanted to rise to a 8' level. You would want to add thrust in the upward direction greater than 1 g. As long as this thrust was greater than 1 g, the device would accelerate upward and you would notice a change in the z-axis measurement. The difference in the two measurements over a period of 1/10 s indicates the linear acceleration for that period. If the difference is 1 g (a thrust of 2 g minus the 1 g of gravity), you could calculate a rise in altitude of 3.2' (32'/10 approximately 1 g). If the thrust is uneven (i.e., some unbalance between rotors, or even a breeze), there will be a change in the x-or y-axis indicating a pitch or roll. Some portion of the thrust is now causing acceleration in a direction other than up. The vectors' combination indicates a tilt degree. This can be calculated from the measurements. Linear motion can also be calculated. This could be expressed in 3-D space as altitude with a bearing and distance. If it were only this easy! We don't even know which way is north.

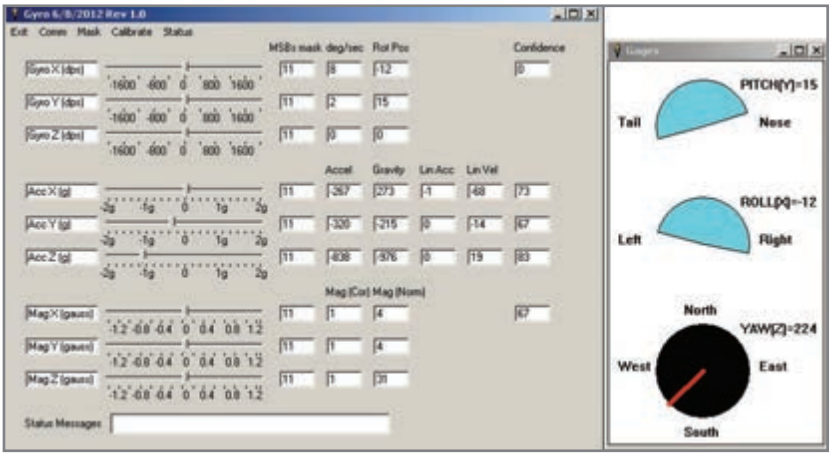


Photo 2—Once my prototype PCB's microcontroller could read the sensors, a serial connection was used to output the readings. This enabled them to be more easily manipulated while learning how the sensors operate. A Liberty BASIC application can be used to help visualize the sensor outputs both as live data and in a more pictorial gauge format, as seen here. BASIC makes it easy to edit calculations and try different approaches without having to experiment with microcontroller code.

BEARING

The MinIMU-9's magnetometer indicates bearing (direction relative to north 0°) by measuring the earth's directional magnetic flux, which runs between poles. While the field's direction runs mostly magnetic north/south, this field's strength can vary by location as well as its proximity to magnetic interference, so it must be calibrated. Calibration is done by continuously taking measurements as it's rotated 360° (while level). The x and y measurements will go through positive and negative field strengths in a sine-cosine-axis relationship. Trigonometry can again be used to determine bearing only if the magnetometer is level. You can find more information on magnetometers in my 2009 article, "Location Notification: A Look at Anisotropic Magnetoresistance Sensors" (*Circuit Cellar* 227). I used a magnetometer (to calculate bearing) and an accelerometer (as a level indicator) to produce a compass. In reality, tilt can be used to compensate the magnetometer's measurements for non-level orientations, but I'm not implementing that here as I don't need a bearing for this project.

ANGULAR VELOCITY

Finally, there is the rate gyro sensor. The mechanical gyroscope I want to replace is outputting an indication of tilt position in two axes, pitch and roll. The MEMS gyro outputs the rate each axis is moving. While static (i.e., landed or hovering), I'd like the gyro to output zero on all axes. However, there is a tendency to slowly creep in one direction, similar to how a mechanical gyroscope's position might creep from the drag of less than perfect bearings. Over time, each axis's measurement indicates a change in the number of degrees per second. By comparing and totalizing measurements (i.e., integrating) over time, a new position (i.e., tilt) can be estimated, based on the last position. Using a three-axis gyro, you can estimate the new position for each axis at each time interval. In the nominal operating position, the z axis indicates a change in bearing, the x axis indicates a change in roll, and the y axis indicates a change in pitch.

Nominal flight characteristics will be less than 100°/s, while gusts or crashes can easily exceed 10 times this. In this project, as a mechanical gyroscope replacement, I have no idea whether the sensor is affected by the pilot's input or some external force as there is no feedback. The duty here is to report what is "seen" or is that "felt?"

BEST GUESS

To keep things as simple as possible, I will introduce a confidence value for each sensor. Since the gyro rate is most effective during movement, I can use the movement amount as an indicator of trust in the sensor. The accelerometer can best predict the gravity vector while it is stationary. Any change in the acceleration measurements may suggest either a new tilt if the axis vector sum is 1 g, or a combination of tilt and acceleration if the total has changed. While thrust is applied in the z axis by pilot control, differences in rotor output will apply angular velocity to tilt the platform (with respect to gravity vector) and enable thrust to be shared by multiple axes for changes in altitude and horizontal velocity. Again, I would like to avoid having the changes in altitude and or horizontal velocity affect the tilt estimations. Without requiring the magnetometer in this application, the gyro's rotational position and the accelerometer's tilt position can be used to complement the magnetometer.

DISPLAY APP

Visual input enables you to interpret words and absorb pictures. And this is a situation where pictures "speak louder than words" when conveying information. Looking at a bunch of

Assembly Language Essentials

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



Author Larry Cicchinelli provides readers with:

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

and more!

On Sale NOW!

$47.50

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

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

numbers coming from sensors can be boring, if not confusing. By digesting these values and drawing a graphic based on the values and/or computations, I can get a feel for what's occurring. I began this step by displaying just the sensor data for the three sensors. Since I had sensor measurements produced by the microcontroller, I needed to capture this data in an application. I used Shoptalk Systems's Liberty BASIC to write this application for my PC as it's user friendly and easy to edit. The microcontroller's data format is "Gyro X Y Z," "Acc X Y Z," and "Mag X Y Z." So, each received string identifies a sensor and the three integer values associated with it. This enables all the data to be presented close to real time. While seeing the numbers roll might be impressive, not much wisdom goes with this.

If I could see the pitch, roll, and bearing represented by these readings represent, I would have more information. Simple graphics might be adequate. I used the "piefilled" graphics commands to create three gauges. Pitch and roll use a sky-colored hemisphere to indicate tilt. The Bearing gauge uses a black circle-filled graphic with a line graphic as a compass needle. Each graphic uses a degree value as input. Pitch and roll use a 0° value as level (i.e., at the 3 o'clock position) and the 0° bearing value of north (i.e., at the 12 o'clock position). Once the graphics routines were correctly presenting this data, I could move on to the real problem, which was deciding how the sensor data would be transformed into degrees for each gauge. After visualizing the data and constantly tweaking the calculations, I've determined values that work consistently well.

The magnetometer was fairly straightforward. It needs calibration, which is a collection of maximum and minimum data points for a level sensor. This is a good thing to do each time it's going to be used. Since it requires rotating the sensor 360°, I entered the results from one calibration event as constants that can be used each time the application is run. This eliminates having to repeat this process. To simplify interpreting sensor readings into degrees, I require the sensor to be level, since this will be a normal characteristic of the quadcopter's operation. Small tilts can have a large effect on the bearing calculation, which is not consistent at every position. An alternative to tilt correction using the z-axis is to do calculations while the sensor is level. How can you tell when you are level? When stationary, you can use gravity's effect on the accelerometer. However, once moving, these measurements include acceleration from other sources and the gravity vector is obscured.

When the accelerometer is stationary and level, you can expect to see zero output from the x- and y-axes, and −1 g on the z-axis. Whenever any axis is outputting these values, you can consider this axis has no movement and give it a 100% confidence level. The further from these ideal values, the lower the confidence level. The total confidence level will be the lowest of any of these. This will enable you to determine when the accelerometer is stationary. When stationary, you can believe the gravity vector and know the tilt (i.e., roll, or x and the pitch, or y) is zero. On the other hand, the gyro does not do well while stationary, as it wanders and drifts. You can associate the gyro output with a confidence level as well. While the accelerometer is stationary, your confidence level is low, but as you see larger outputs, your confidence level increases. The gyro's data output enables you to determine the present rate (in degrees) of a rotating axis. If you know where you started (at zero), you can calculate where each axis is currently located based on the data's sampling rate and the rate measurement in degrees per second. Once a new axis position is known, this can be verified by a change in the accelerometer's gravity vector. If the combined accelerometer vectors equal 1 g, you know you have rotated to a new position and the sensor is not linearly moving. Any deviation from 1 g indicates the sensor is measuring an additional linear acceleration in some direction.

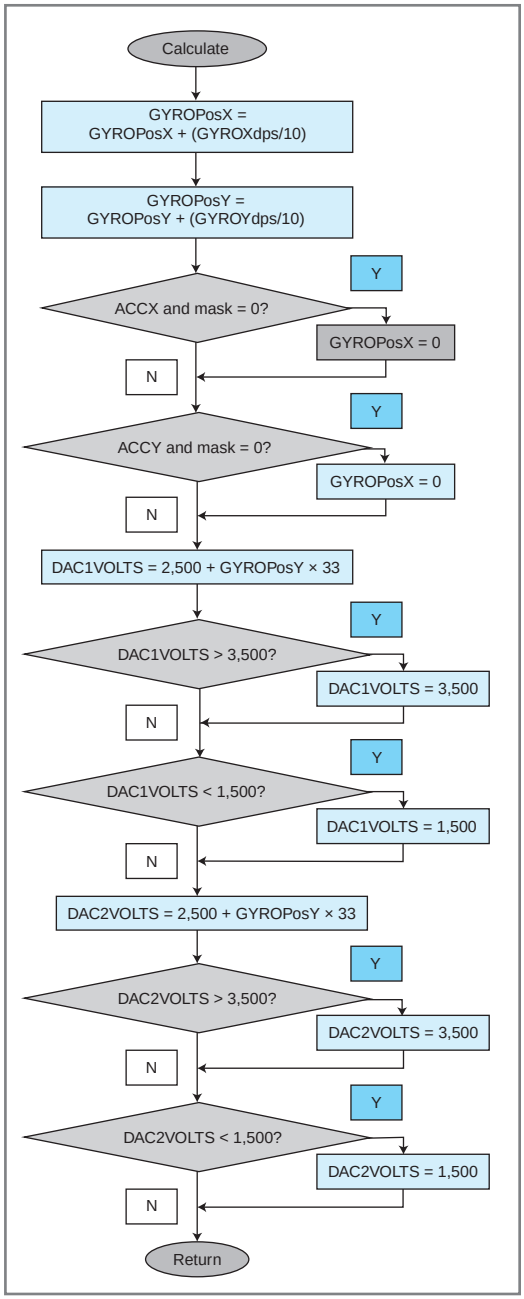


Figure 1—This flowchart calculated the output voltages that correspond to tilt (i.e., pitch and roll) measurements by the accelerometer and the rate gyro.

To prevent the gyro's additive drifting over time, I use the accelerometer's axis confidence level to zero the gyro's associated axis position whenever appropriate. This way, the gyro always stays relevant to zero. To ensure the magnetometer's (simple) calculation is accurate, I use the total accelerometers confidence level to indicate when the bearing can be trusted. This is accomplished by drawing the compass needle in red, yellow, or green to indicate confidence. Photo 2 shows a typical view of the data received from my prototype module.

AN EFFECTIVE ALGORITHM

With what I learned through the easily edited BASIC application, I can go back to my prototype and develop

the code necessary to accomplish my intended task. I purposely stayed away from using any floating-point calculations to reduce the programming requirements for my microcontroller's code. In fact, by scaling the trigonometry functions, I can use integer math and lookup tables to speed computations.

I can use one jumper on the prototype to enable and disable the gyro. With the gyro disabled, both the x and y outputs produce a constant 2.5 V, which tells the quadcopter's controller all is level. With the gyro enabled, any axis's tilt will lower or raise the appropriate 2.5-V output up to ±1 V, which indicates a maximum ±30° tilt. This provides a way of determining any improvement in flight dynamics when adding the solid-state sensor's input to the flight control system.

Figure 1 shows the calculations done at a 100-ms rate to determine output voltages to mimic the mechanical gyroscope I'm replacing. A constant 2.5-V output indicates a level flight capable of hovering in one spot. A tilt of either pitch and/or roll (with sufficient thrust) enables the quadcopter to move in a horizontal direction with a sharing of vertical and horizontal thrust.

I still don't know how the flight (i.e., rotor) controller utilizes the user's input (i.e., radio controller) and the gyro input to produce requested directional thrust yet compensates for unwanted changes in vehicle position. The next improvement for this vehicle will most likely require redesigning the flight control system to enable autonomous flights. I could then make use of the magnetometer and even add GPS for more sophisticated positional awareness. This opens the door to additional sensors that can sense the horizon or the distance to ground. Sometimes a project can become the tip of a great iceberg. I might just have to revisit this at some point in the future. So much to learn, so little time.

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.*

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/268.

RESOURCES

J. Bachiochi, "Mechanical Gyroscope Replacement (Part 1): A Microelectromechanical Systems (MEMS) Solution," *Circuit Cellar* 267, 2012.

———,"Location Notification: A Look at Anisotropic Magnetoresistance Sensors," *Circuit Cellar* 227, 2009.

SOURCES

MCP4*xx*1 DAC and PIC16F1829 Microcontroller
Microchip Technology, Inc. | www.microchip.com

MinIMU-9 Inertial measurement unit
Pololu Corp. | www.pololu.com

Liberty BASIC
Shoptalk Systems | www.libertybasic.com

Get PUBLISHED. Get NOTICED. Get PAID.

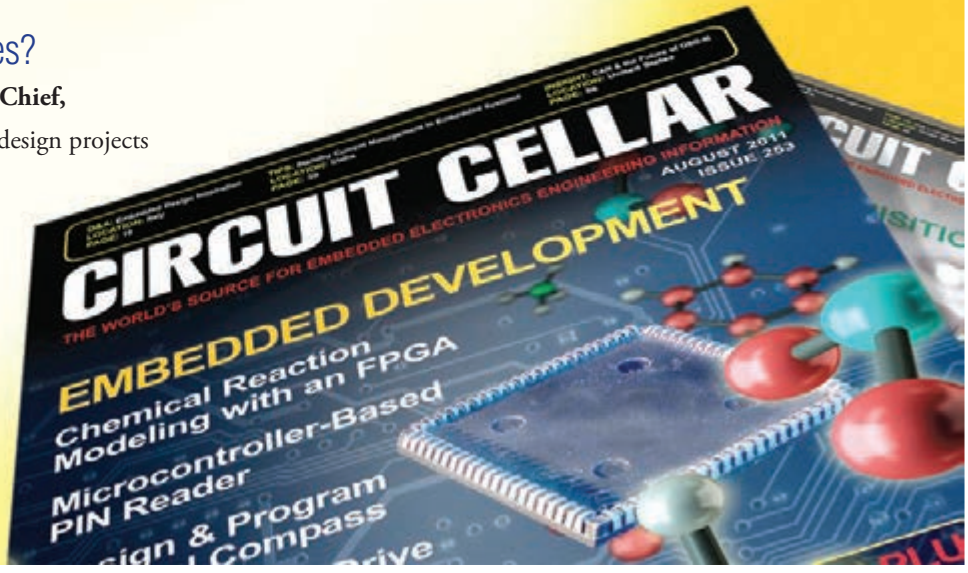
Circuit Cellar feature articles are contributed by professional engineers, academics, and students from around the globe. Each month, the editorial staff reviews dozens of article proposals and submissions. Only the best make it into the pages of this internationally respected magazine.

Do you have what it takes?

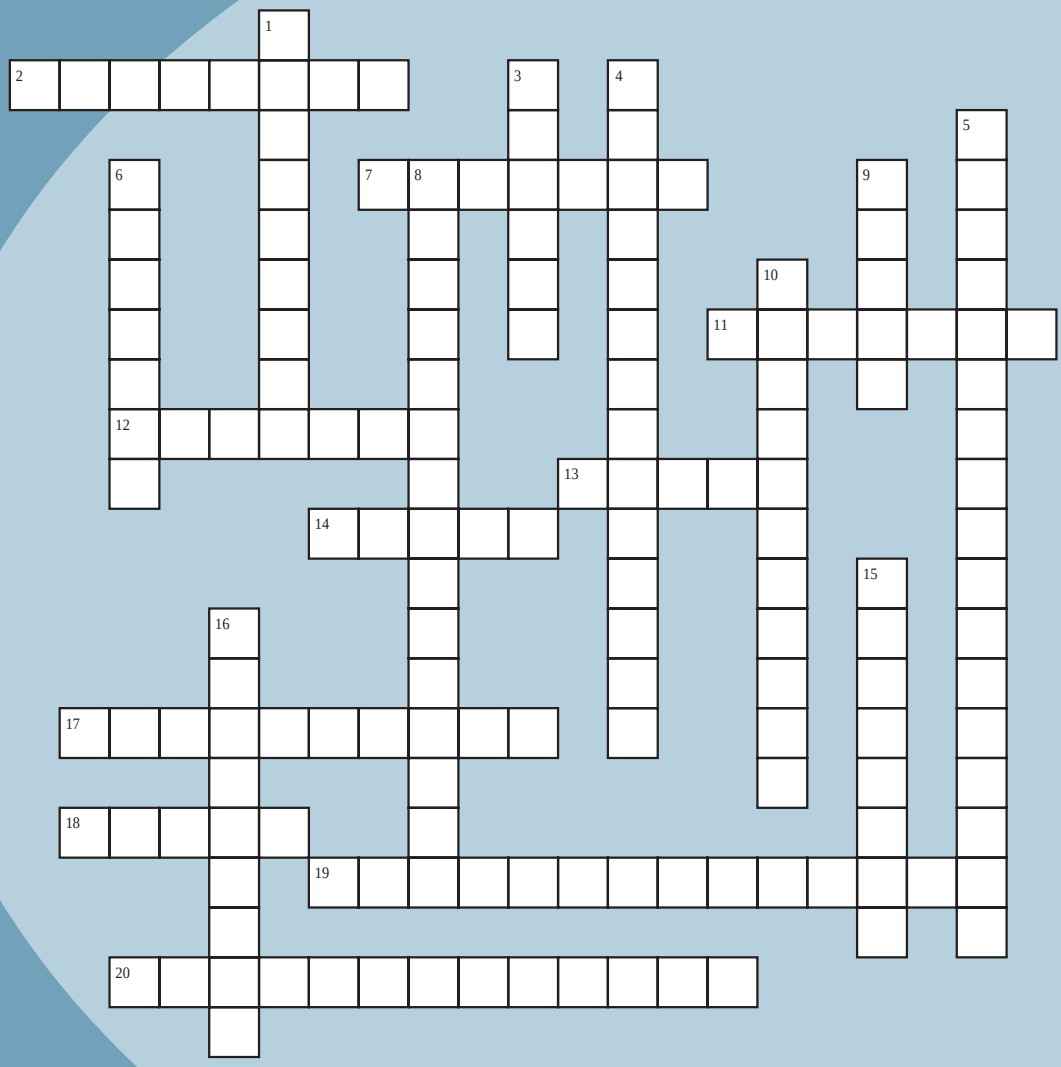
Contact **C. J. Abate, Editor-in-Chief,** today to discuss the embedded design projects and programming applications you've been working on and your article could be featured in an upcoming issue of *Circuit Cellar* magazine.

editor@circuitcellar.com

CIRCUIT CELLAR



CROSSWORD



Across

2. Columnist Jeff Bachiochi taught readers how to use this graphical programming language in his recent article about flowcharting (*Circuit Cellar* 266, 2012)
7. This type of transform is similar to Fourier, but expresses functions into moments as opposed to vibration
11. Channel to hold wires, cables, and so forth
12. *Circuit Cellar*'s 250th issue (2011) focused on Measurement and this other topic
13. A metallic contact area
14. Interviewee (*Circuit Cellar* 253, 2011) who designed the "Witness Camera," a self-recording surveillance camera
17. This type of pair can be produced using individual transistors or purchased as a single device, as in a 2N6301
18. A slice of semiconductor material upon which monolithic ICs are produced
19. The insulating material that makes up the cable's center through which the conductors are run [two words]
20. A synthetic, flexible mixture of rosins used as an insulating material

Down

1. In his article "Hardware-Accelerated Encryption" (*Circuit Cellar* 266, 2012), columnist Patrick Schaumont said AES encryption's real secrecy comes from the periodic additions of these
3. A type of planar tube, similar to the lighthouse tube, which has cooling fins
4. In the 1970s, *Circuit Cellar* founder Steve Ciarcia wrote his first article for *BYTE* about this topic
5. An oscillator controlled by voltage input; there are usually two types: harmonic and relaxation [two words]
6. Describes compromising emanations
8. In a communications system, the time interval required to attain synchronism [two words]
9. Company credited with making the first single-chip microprocessor
10. How one device communicates with one or more other devices, at a predetermined speed
15. aka "varicap"
16. Active filter, two-pole

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

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

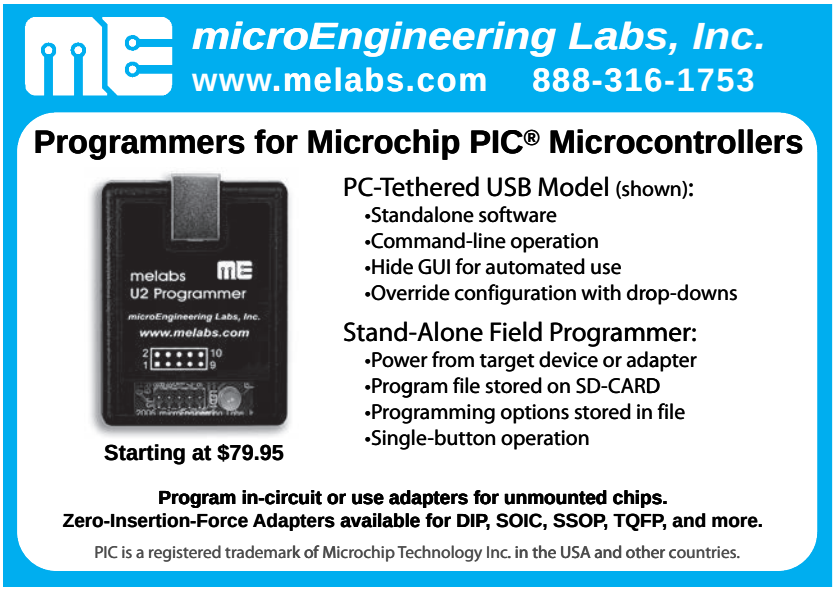
AD FORMAT: Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/advertise). ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter Wostrel at 978.281.7708 or peter@smmarketing.us.

The Vendor Directory at www.circuitcellar.com/vendor
is your guide to a variety of engineering products and services.









HUSB™

OEM $79

- Add a high speed USB port to a TERN controller.
- High performance USB stack chip (FT232H, FTDI)
- Ready to use, royalty free USB drivers
- USB 1.1 and USB2.0 compatible
- Data transfer rate to 8 MB/sec with D2xx driver
- 2.1"x1.3"



100+ Low Cost Controllers with ADC, DAC, UARTs, 300 I/Os, solenoid, relays, CompactFlash, LCD, Ethernet, USB, motion control. Custom board design. Save time and money.

TERN INC. 1950 5th Street, Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181 www.tern.com • sales@tern.com

NOW AVAILABLE!

CC Electronic Toolbox App

Databases, schematics, calculators, & more!

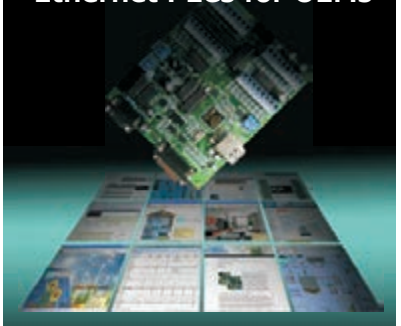


For iPad, iPhone, & iPod Touch

Available on the App Store

LISTEN TO YOUR MACHINES

Ethernet PLCs for OEMs



FMD88-10 and FMD1616-10

Integrated Features :
- ETHERNET / Modbus TCP/IP
- 16 or 32 digital I/Os
- 10 analog I/Os
- RS232 and RS485
- LCD Display Port
- I/O Expansion Port
- Ladder + BASIC Programming

$229 and $295
before OEM Qty Discount

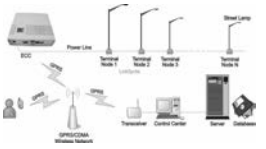
tel : 1 877 TRI-PLCS
web : www.tri-plc.com/cci.htm

TRI TRIANGLE RESEARCH INTERNATIONAL

BPS BusBoard Prototype Systems

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

Street Lighting System, Power-Line Communication, Long Range RFID Reader, Arduino



■Power-Line Communication Street Lighting System
■WiFi Street Lighting System

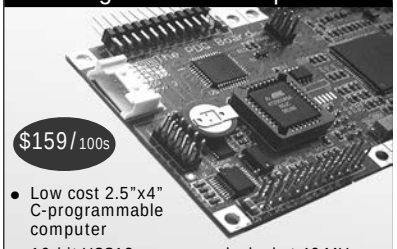


■Power-Line Communication Modules: Spyder and McLaren
■Arduino with WiFi: Diamondback, Redback
■Arduino Shields: Cupperhead and Juniper WiFi, PLC, GPRS, MIDI, Touch, gameduino
■Long Range UHF Gen2 RFID
■JPEG Serial Cameras
■4TV
■Robotics



www.linksprite.com
www.linkspritedirect.com

PDQ Board™ - A Fast I/O-Rich Single Board Computer



$159/100s

- Low cost 2.5"x4" C-programmable computer
- 16-bit HCS12 processor clocked at 40 MHz
- 8 PWM, 8 counter/timer, and 8 digital I/O
- 16 10-bit A/D inputs
- Dual RS232/485 ports, SPI and I²C ports
- 512K on-chip Flash, 512K RAM with Flash backup
- Plug-in I/O expansion, including Ethernet, Wi-Fi, GPS, 24-bit data acquisition, UART, USB, Compact Flash card, relays, and more ...

Mosaic Industries Inc.
tel: 510-790-1255 fax: 510-790-0925
www.mosaic-industries.com

ALL ELECTRONICS CORPORATION

Electronic and Electro-mechanical Devices, Parts and Supplies. Wall Transformers, Alarms, Fuses, Relays, Opto Electronics, Knobs, Video Accessories, Sirens, Solder Accessories, Motors, Heat Sinks, Terminal Strips, L.E.D.S., Displays, Fans, Solar Cells, Buzzers, Batteries, Magnets, Cameras, Panel Meters, Switches, Speakers, Peltier Devices, and much more....

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

Disk On Chip
Ready for Delivery



From 16MB to 128MB Available!
Call 530-297-6073 Sales@jkmicro.com
www.jkmicro.com

jk microsystems, Inc.
International Orders Welcome

BGA & QFN Sockets
Quick-Turn Custom Designs



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

I²C™/SMBus



- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools

MCC Micro Computer Control
www.mcc-us.com

PIC-SERVO
MOTION CONTROL

MOTION CONTROLLERS FOR BRUSH, BRUSHLESS AND STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com
JEFFREY KERR, LLC

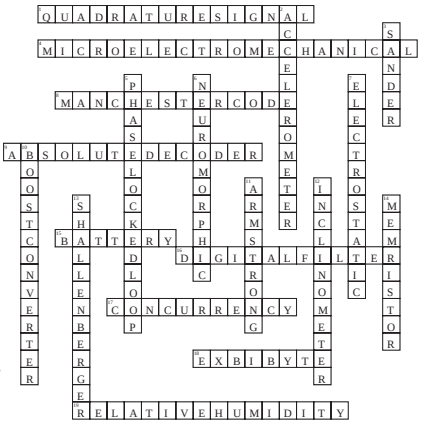
CROSSWORD ANSWERS from Issue 267

Across

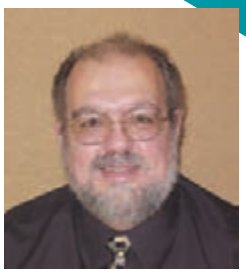
1. **QUADRATURESIGNAL**—Can be produced using two sensors spaced at odd half-slot multiples around a single track [two words]
4. **MICROELECTROMECHANICAL**—This type of system's size ranges from 20 µm to 1 mm
8. **MANCHESTERCODE**—A low-to-high transition means "0" and a high-to-low transition means "1" [two words]
9. **ABSOLUTEDECODER**—Because these devices have only one track per bit of resolution, they can require large diameters, which gives them a nonvolatile and unique output for each position [two words]
15. **BATTERY**—Italian physicist Alessandro Volta (1745–1827) is credited with inventing the first one of these in the 1800s
16. **DIGITALFILTER**—A piece of software, firmware, or logic circuit that takes a digital data flow as an input and provides a filtered version of this signal on its output [two words]
17. **CONCURRENCY**—Topic of columnist Bob Japenga's ongoing article series, which began in *Circuit Cellar* 263, 2012
18. **EXBIBYTE**—1,152,921,504,606,846,976 bytes
19. **RELATIVEHUMIDITY**—Amount of water vapor in the atmosphere expressed as a percentage of the total amount the air can hold at the current temperature [two words]

Down

2. **ACCELEROMETER**—The design in Mark Pedley's article, "eCompass: Build and Calibrate a Tilt-Compensating Electronic Compass" (*Circuit Cellar* 265, 2012), was built using one of these
3. **SANDER**—New Zealand-based *Circuit Cellar* contributor and recent interviewee who is fascinated with advanced robot technologies
5. **PHASELOCKEDLOOP**—This control system generates an output frequency, which can be either higher or lower than the input, based on a reference input clock [three words]
6. **NEUROMORPHIC**—*Circuit Cellar*'s October's interviewee, Helen Li, believes this type of computing will solve the contradiction between the limited functions of computing systems and the ever-increasing variety of applications
7. **ELECTROSTATIC**—This type of cell consists of a thin plastic film sandwiched between two metal stators
10. **BOOSTCONVERTER**—Its output voltage is greater than its input voltage [two words]
11. **ARMSTRONG**—American engineer (1890–1954) who invented the regenerative circuit, the super-regenerative circuit, the superheterodyne receiver, and modern frequency modulation (FM) radio transmission
12. **INCLINOMETER**—Used to measure tilt
13. **SHALLENBERGER**—American engineer (1860–1898) who invented an induction meter to measure alternating current
14. **MEMRISTOR**—The functional equivalent of a synapse



PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

An Internet Education

Every time I visit the doctor and complain about some symptom he says, "Stop looking on the Internet or you'll turn into a hypochondriac!" Type a few keywords into Google, and you'll instantly be presented with news articles, personal blogs, commercial vendors, scholarly papers, and maybe a discussion forum or two that are all more-or-less relevant to that topic. The problem is that Google gives you no clue about the information's credibility. Trying to make sense of it is like trying to sip from a fire hose.

In this age of instant gratification, most people read just the first few links on the first page Google presents and then move on to other things. If you already know something about the topic, you at least have a fighting chance of weeding out the sites containing incorrect information or promoting their own agendas. But, if you're completely new to the topic, it's nearly impossible to discern what is accurate.

Medicine aside, learning any new subject takes effort, and you need to get your information from a trusted source. Back in the day (warning: old curmudgeon talking here), that usually meant going to the library to read printed books and journals. These sources could generally be trusted because printed books cost money and any publisher valuing his credibility usually had a peer review process to ensure the information's accuracy. Over time, people generally learned which publishers were most credible and went to them first.

The Internet has pretty much negated that model and it's a bit like the Wild West. Because it costs virtually nothing to publish anything you want on the web (e.g., a personal blog), credibility review isn't a prerequisite to stating false theories any more than stating true ones. To combat this, some websites go to great pains to build credibility, again, primarily through the power of crowd-sourcing and peer review. Wikipedia is probably the best-known example. A link to a Wikipedia article about any given topic usually appears in the first page of Google results. It's usually the first link I click.

The Internet also has made some pretty amazing technology available "to the masses" at reasonable prices. Besides the variety of consumer goods, now you can easily buy microelectromechanical systems (MEMS) sensors, wireless communications modules, single-board computers for every capability level, displays, and so forth. And there is a plethora of websites (e.g., Arduino, etc.) that explain how to put these building blocks together to do interesting things.

All this is background for my main point, which is about the huge variety of people who visit these sites and what they're really getting out of their experiences. In several cases, I came across non-engineers exhibiting a serious "cognitive disconnect" between their perceived technical accuracy and their project-related expectations. Example: someone with no programming or electronics experience wants to do something sophisticated like connect a strain gauge to a Raspberry Pi board. Without the proper foundation, it's virtually impossible to know where to start when answering a question like that. You'd pretty much have to write out one complete education in electronics and another in programming before you could fully understand the answer to the question.

I realize that everyone has to start somewhere, and I'm glad there are people who have the ambition to tackle projects outside their comfort zones. But it seems the combination of inexpensive building blocks, instructional websites, and the expectations of instant gratification is encouraging more and more people to jump into the deep end of the pool without making any attempt to learn the basics.

These people may or may not be able to solve their immediate problems, but any "education" they receive along the way could be narrowly focused, heavily skewed, and full of huge gaps. It's more likely they'll end up frustrated and abandon the project. As engineers, I'm not sure what we can do about this situation, except to be aware that it exists. In our day-to-day interactions, we need to recognize such people when we come across them, and where possible, we must be willing to appropriately mentor them so they have a positive experience and want to learn more. Encouraging them to read *Circuit Cellar* isn't a bad idea either.

steve.ciarcia@circuitcellar.com



www.ftdichip.com

FTDI Chip

ENHANCED USB PERFORMANCE

Streamlined USB Bridge Solutions

X-CHIP

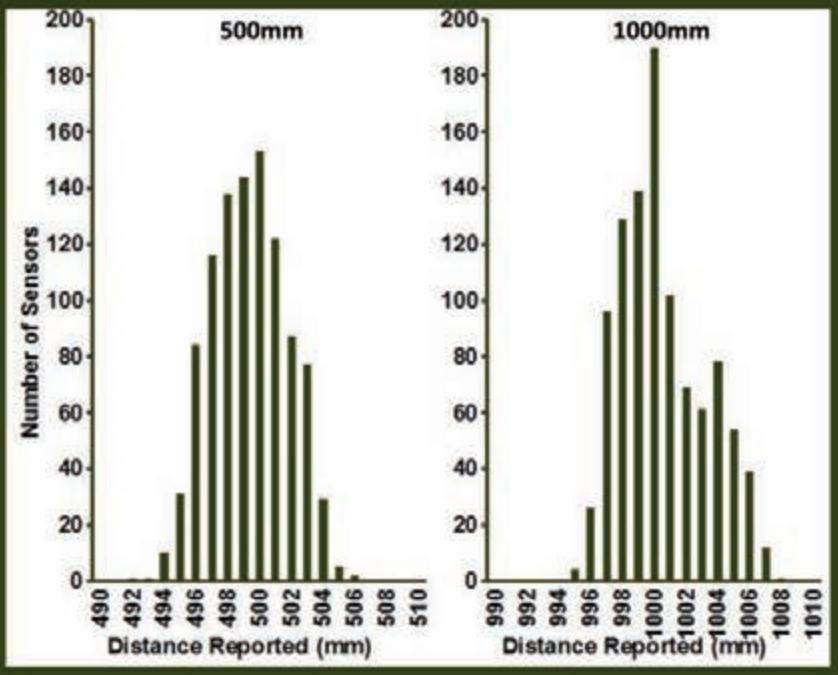
EXtensive Interfaces
UART, FIFO, SPI, I²C, FT1248

EXtended Features
Battery charger detection
Low active power (8 mA, typical)
Internal MTP memory
Expandable clocking; clock generation and system clock out

EXceptional Drivers
Windows, MacOS, Android, and Linux



MaxBotix Offers
High Accuracy Distance Sensors



MaxBotix sensors use sound to measure the distance to nearby objects with outputs that are easy to use and integrate. Our newest ultrasonic range finding sensors offer a 1mm resolution. Try one today for only $34.95 indoor, or $109.95 outdoor.

The new High-Resolution sensors feature: on-board temperature compensation, target size compensation, up to a 10Hz reading rate, a calibrated beam pattern, and low power requirements.

About the Graphs: These graphs show the total variation across a complete sample of HRLV-MaxSonar-EZ sensors, voltages, and target distances. For a given setup, each sensor's range output had a standard deviation of less than 1mm. 250 sensors were tested at 2.7V, 3.3V, 3.9V and 5V with target distances of 500mm and 1000mm. Each graph above shows 1000 range readings.



$29.95 MSRP

People Sensing made EZ
LV-ProxSonar-EZ line

This new proximity sensor offers people detection in a small, compact and affordable package. We offer detection zones, from 1 foot to 7 feet, to meet almost any proximity application. Perfect for HIPAA compliance and multiple side by side kiosks, as these sensors also play well together, allowing multi-sensor operation. Check out our datasheets online for additional features and products.

MaxBotix® High Performance Ultrasonic Rangefinders

www.maxbotix.com

Come see us at MDM conference in Minneapolis, MN October 30th thru November 1st. Booth #451

The names MaxBotix®, MaxSonar®, EZ™, and ProxSonar® are trademarks of MaxBotix Inc.