CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

MAY 2011
ISSUE 250

MEASUREMENT & SENSORS



0 5>
7 25274 75349 9
$7.50 U.S. ($8.50 Canada)
www.circuitcellar.com

Announcing a complete hardware and software solution from NetBurner

The NetBurner MOD5234

ETHERNET CORE MODULE with eTPU



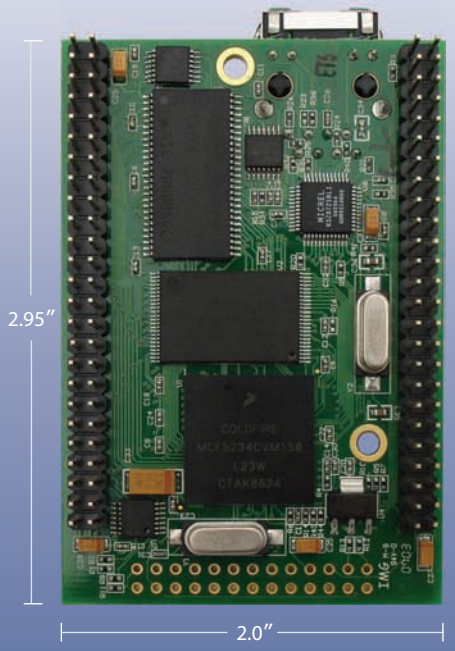
Only $99
Qty. 100

NetBurner MOD5234 Ethernet Core Module Features

INDUSTRIAL TEMPERATURE RANGE
| –40°C to +85°C

PERFORMANCE AND MEMORY
| 32-Bit CPU | Freescale ColdFire MCF5234 147 Mhz
| 2MB Flash Memory | 8MB SDRAM

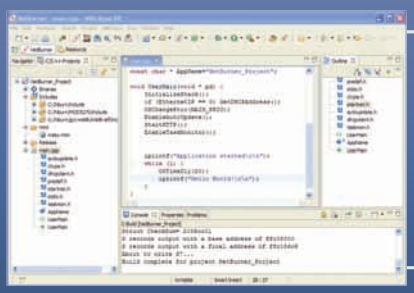
DEVICE CONNECTIVITY
| 10/100 Ethernet | 3 UARTs | 16-channel eTPU | I²C | SPI | CAN
| 47 Digital I/O | 16-bit Data Bus | SD/MMC Flash Card Support



Customize with NetBurner's Royalty-free Software Suite

DEVELOPMENT SOFTWARE
| NB Eclipse IDE | Graphical Debugger | Deployment Tools | Examples

COMMUNICATION SOFTWARE
| TCP/IP Stack | HTTP Web Server | FTP | E-Mail | PPP | Flash File System



All hardware and software is included with the

NetBurner MOD5234 Development Kit for only $299!

The Development Kit features NetBurner's Eclipse, an enterprise level professional IDE that provides editing, downloading and debugging in one environment.

Order the MOD5234 Development Kit: Product No. NNDK-MOD5234-KIT

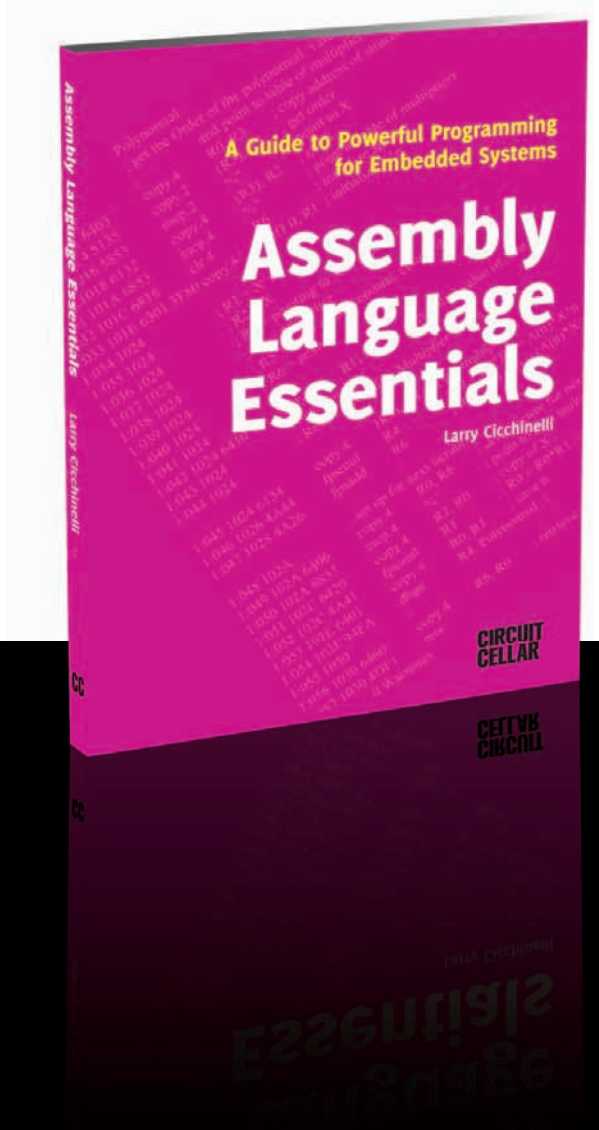
NetBurner
Networking in One Day!

Product No. | MOD5234-100IR
Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

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!

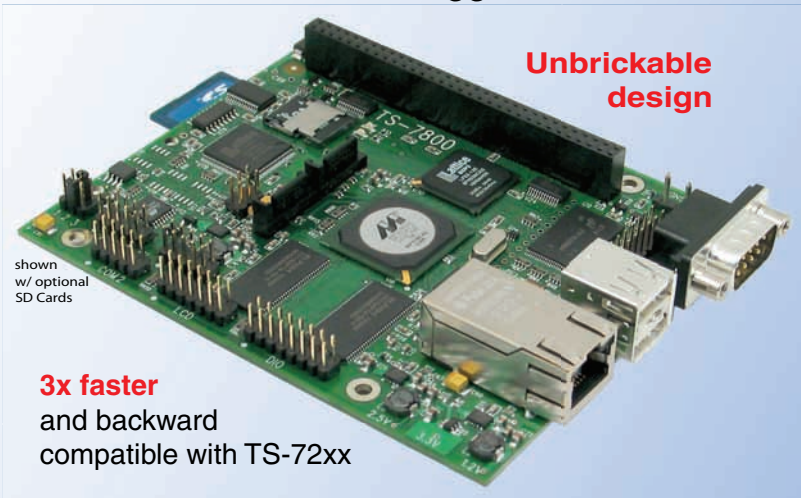
$47.50

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

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

Embedded Systems

High-End Performance with Embedded Ruggedness



shown w/ optional SD Cards

Unbrickable design

3x faster and backward compatible with TS-72xx

TS-7800
500 MHz ARM9

- Low power - 4W@5V — $229 qty 100
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash — $269 qty 1
- 12K LUT customizable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in 0.7 seconds
- Linux 2.6 and Debian by default

TS-TPC-7390
7" Color Touch Panel Computer

- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 200 MHz ARM9
- 64MB SDRAM (128MB opt)
- 512MB Flash w/ Debian Linux
- 800 x 480 video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in less than 2 seconds
- Unbrickable, boots from SD or NAND
- Runs X Windows GUI applications
- Runs Eclipse IDE out-of-the-box



$449 qty 1

More Touch Panel Computers on our website

- 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

TS-WIFIBOX-2

A Complete Solution for 802.11g WiFi Applications

NEW!

$185 qty 1

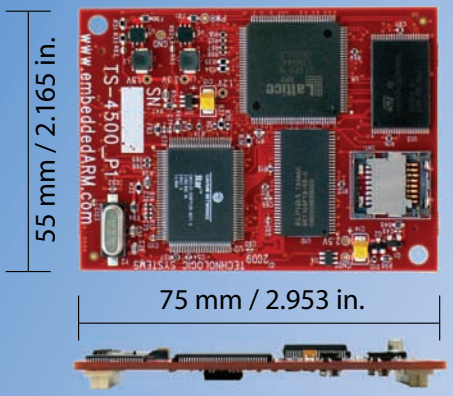


- Low power (3.2 watts), fanless
- Power via 5-12 VDC, USB, PoE (opt.)
- 64MB DDR-RAM
- 256MB ultra-reliable XNAND drive
- Micro-SD Card slot
- RS-232, RS-485, CAN, RTC
- Header with SPI and 11 DIO
- 480Mbit/s USB, Ethernet, PoE option
- Boots Linux 2.6.24 in < 3 seconds
- Un-brickable, boots from SD or flash
- Customizable FPGA - 5K LUT
- Optional DIN mountable enclosure

Ideal for gateway or firewall, protocol converter, web server, WiFi audio, and unattended remote applications

NEW!

series starts at $92 qty 100



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 projects include:

- TS-4200: Atmel ARM9 with super low power
- TS-4300: Cavium ARM11 with dual 600 MHz and FPU
- TS-4500: Cavium ARM9 at very low cost
- TS-4700: Marvell PXA168 with video and 1.2 GHz CPU
- TS-4800: Freescale iMX515 with video and 800 MHz CPU
- Several COTS baseboards for evaluation & development

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

Technologic Systems

We use our stuff.

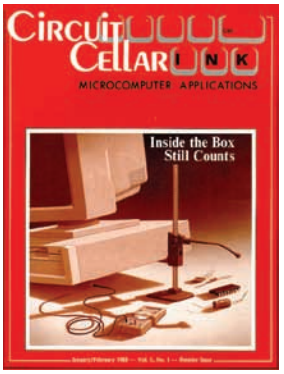
Visit our TS-7800 powered website at

www.embeddedARM.com

TASK MANAGER

Issue 250 and Beyond

This month, as we celebrate *Circuit Cellar*'s 250th issue, we have good cause to consider what we have accomplished and what we plan to do in the future. In January 1988, *Circuit Cellar* shipped its first issue. As you see below, it featured a Robert Tinney cover with the depiction of a PC being lifted with a car jack. (Look inside the box!) On the issue's first page, Steve Ciarcia set forth the magazine's mission:



> *To further the education for and communication with those individuals intelligent enough to recognize that sometimes it is very important to know exactly what's in the box and how it works. Without a continuous effort at understanding and improving present achievements, we cannot progress to higher levels of achievement.* Circuit Cellar Ink *is a publication designed to increase that awareness. We must not ignore the fact that it is a combination of people and machines that create intelligent personal systems.*

Today, *Circuit Cellar* staff and contributors remain dedicated to the same goals of looking inside all relevant embedded technologies and encouraging communication between electronics engineering professionals. To do this in an ever-changing media landscape, we've evolved into something more than a publisher of a print magazine about hardware projects. We facilitate peer reviews of MCU-based design projects of all sorts, we expose engineers to new technologies (e.g., design challenges and sample programs), and we deliver need-to-know information about a variety issues, from programming to robust design to electronics theory.

Moving forward, as we continue this tradition and seek out new opportunities, and we need you—our readers—to get involved. I encourage you to submit articles, participate in design challenges, and interact with the editorial department's staffers (http://twitter.com/editor_cc).

Now let's focus on the present. This month, we feature articles on several essential engineering topics: functional upset (p. 14), spectrum analysis (p. 16), encoder implementation (p. 26), design planning and testing (p. 36, 56, 64), microprogramming (p. 46), and error checking and the CRC (p. 68).

Hold on to this issue. Like Issue 1 shown above, it's bound to be a collectible. Want an even better idea? You can purchase additional copies at www.cc-webshop.com!

cj@circuitcellar.com

CIRCUIT CELLAR®

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
C. J. Abate

ASSOCIATE EDITOR
Nan Price

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Ken Davidson
David Tweed

STAFF ENGINEER
John Gorsky

PUBLISHER
Hugo Van haecke

ASSOCIATE PUBLISHER
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ADMINISTRATIVE COORDINATOR
Valerie Luster

ART DIRECTOR
KC Prescott

MARKETING ASSISTANT
Daniel Netkin-Colllins

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

Circuit Cellar's mission is to collect, select, and disseminate need-to-know information around the world in the fields of embedded hardware, embedded software, and computer applications. Circuit Cellar uses an assortment of print and electronic content-delivery platforms to reach a diverse international readership of professionals, academics, and electronics specialists who work with embedded, MCU-related technologies on a regular basis. Our aim is to help each reader become a well-rounded, multidisciplinary practitioner who can confidently bring innovative, cutting-edge engineering ideas to bear on any number of relevant tasks, problems, and technologies.

ADVERTISING
800.454.3741 • 978.281.7708 • www.circuitcellar.com/advertise

ADVERTISING REPRESENTATIVE
Peter Wostrel
Strategic Media Marketing, Inc.
1187 Washington St., Gloucester, MA 01930 USA
800.454.3741 • 978.281.7708
peter@smmarketing.us • www.smmarketing.us
Fax: 978.281.7706

ADVERTISING COORDINATOR
Kim Hopkins
E-mail: khopkins@circuitcellar.com

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.cc-access.com, E-mail: subscribe@circuitcellar.com
Subscribe: 800.269.6301, www.cc-access.com, Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650
Address Changes/Problems: E-mail: subscribe@circuitcellar.com

GENERAL INFORMATION
860.875.2199, Fax: 860.871.0411, E-mail: info@circuitcellar.com
Editorial Office: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: editor@circuitcellar.com
New Products: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

AUTHORIZED REPRINTS INFORMATION
860.875.2199, E-mail: reprints@circuitcellar.com

AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (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 $45, Canada/Mexico $60, all other countries $63. Two-year (24 issues) subscription rate USA and possessions $80, Canada/Mexico $110, all other countries $116.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

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.

Entire contents copyright © 2011 by Circuit Cellar, Incorporated. 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.



Find It Here. First.

The Newest Products for Your Newest Designs™

Authorized distributor for the most advanced semiconductors and electronic components.

Get What's Next. Right now at mouser.com.

MOUSER ELECTRONICS®

a tti company

Mouser and Mouser Electronics are registered trademarks of Mouser Electronics, Inc. Other products, logos, and company names mentioned herein, may be trademarks of their respective owners.

INSIDE ISSUE 250

May 2011 • Measurement & Sensors

16 **Stereo Spectrum Analyzer**
Richard Pierce

26 **Control Shaft Encoders**
Keith Brown

36 **Reprogrammable UAV Autopilot System (Part 2)**
Testing and Results
Mariano Lizarraga, Renwick Curry, and Gabriel Elkaim

46 **Getting Started with Microprogramming (Part 2)**
A Look at the Meta-Assembler
Thomas Mitchell

56 **Testing Platform**
Construct a Custom Scope
Jaime Garnica

Build a Stereo Spectrum Analyzer



p. 16

How to Choose an Encoder





p. 26

UAV Autopilot Development



p. 36



The dsPIC-Based Scopey II



p. 56

14 **THE CONSUMMATE ENGINEER**
Functional Upset
George Novacek

64 **LESSONS FROM THE TRENCHES**
Design Development (Part 1)
Planning and Tools
George Martin

68 **FROM THE BENCH**
Error Checking
Jeff Bachiochi

TASK MANAGER 4
Issue 250 and Beyond
C. J. Abate

NEW PRODUCT NEWS 8

TEST YOUR EQ 13

CROSSWORD 74

INDEX OF ADVERTISERS 79
June Preview

PRIORITY INTERRUPT 80
Happy 250th
Steve Ciarcia

Faster CPU and More Memory?
The CUWIN5000 Now Shipping!

CUWIN



Need more sophisticated features in a touch screen?
Enter the CUWIN? A Revolutionary Touch Controller.

Programmable in C++ or any .Net Language (C#, VB.Net, etc...) for the creation of highly sophisticated applications involving data acquisition, machine control, sensor monitoring, and even web reporting, just to name a few.

Cheaper than a PC, and more features than an HMI touch screen.
The CUWIN? A Revolutionary Touch Controller
Already in use in thousands of applications throughout the world.



▶ CUWIN5200 ▶ CUWIN5500 ▶ CUWIN6300

- Samsung 2450 (533MHz) ARM Processor
- 128MB SDRAM / 128MB NAND Flash
- Windows® Embedded CE 6.0
- .Net Compact Framework 3.5
- Software Development: C++ or any .Net Language (C#, VB.Net, etc...)
- Ethernet 10/100Base-TX
- RS-232 x 3, RS-485 x 1
- USB 2.0 x 1, USB 1.1 x 1
- Audio (Line Out) / RTC
- Keyboard and Mouse (optional)
- CE, FCC, and KCC certified

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

COST-EFFECTIVE, HIGH-PERFORMANCE 1-T 8051 MICROCONTROLLER

The **STC12C5A60S2** is a single-chip microcontroller based on a high-performance 1-T architecture 80C51 CPU. Featuring an enhanced kernel, the STC12C5A60S2 executes instructions in one to six clock cycles (about six to eight times the rate of a conventional 8051 device), and has a fully compatible instruction set with industry-standard 80C51 series microcontrollers. In-system programming and in-application programming enable you to perform upgrades through a programmer or through the firmware. The STC12C5A60S2 retains all the features of the standard 80C51. In addition, the STC12C5A60S2 has two extra I/O ports, a 10-source, four-priority-level interrupt structure, a 10-bit ADC, two to three UARTs, a SPI (Slave/Master), an on-chip oscillator, a two-channel PCA/PWM/DAC, and a one-time-enabled WDT.

The STC12C5A60S2 offers high-noise immunity and has very low power consumption. It operates at 3.5 to 5.5 V, while the low-voltage version operates at 2.1 to 3.6 V. The device has a maximum working frequency of 35 MHz. It offers 60 KB of flash memory, 1,280 bytes of data RAM, and 1,024 bytes of EEPROM. Available packages include DIP-40, PLCC-44, LQFP-44, LQFP-48, and QFN-40.

The STC12C5A60S2 costs **$1.50** in 1,000-piece quantities. STC Microcontroller offers a free USB ISP tool for new customers.



STC Microcontroller, Ltd.
www.stc-51.com

MULTIFUNCTION ANALOG AND DIGITAL COMBINATION MODULE

Combining analog and digital I/O, **Seal/O-570** modules are designed for applications such as process control, data acquisition, broadcast communications, and remote environmental monitoring.

The Seal/O-570 provides eight single-ended 16-bit analog inputs, eight optically isolated digital inputs, and eight Form C relay outputs. The modules' A/D inputs are independently software selectable for ±5 VDC or ±15 VDC ranges. They feature high-input impedance and easily connect to a variety of sensors. The Seal/O-570's eight isolated inputs are 5- to 30-VDC-rated and provide 3,500-VDC external isolation. Its eight Form C outputs are configurable as normally open or normally closed, and they can switch DC loads up to 24 W.

Ordering options enable connection to a host device via wireless (802.11 b/g), Ethernet (Modbus/TCP), RS-485 (Modbus/RTU), USB, or RS-232. The modules include field-removable terminal blocks to help facilitate fast, flexible field wiring. Seal/O modules operate from 9 to 30 VDC. Power can be input via a terminal block or DC jack. Table-mount and DIN-rail-mounting options are available, and installation is accomplished using Sealevel's software configuration tools.


Ethernet Interface Shown

The standard operating temperature range for the modules is –20° to 70°C. An extended temperature range (–40° to 85°C) is optional.

Pricing for the Seal/O-570 starts at **$399**.

Sealevel Systems, Inc.
www.sealevel.com

WIRELESS pH/TEMPERATURE TRANSMITTER

The **UWPH-2-NEMA** wireless pH temperature transmitter features a high-performance, microprocessor-based wireless radio transmitter that is built into a NEMA enclosure. It transmits pH, temperature, and signal strength in real time and works with UWTC series receivers for a complete wireless system. The UWPH-2-NEMA is ideal for outdoor or high-humidity applications including water treatment, chemical, food processing, waste treatment, and disinfecting.

The transmitter includes free software that converts a PC into a multichannel chart recorder or data logger.

Prices start at **$265**.



Omega Engineering, Inc.
www.omega.com

NEW PRODUCT NEWS

Edited by John Gorsky

SECOND-GENERATION, HIGH-VOLTAGE BATTERY STACK MONITOR

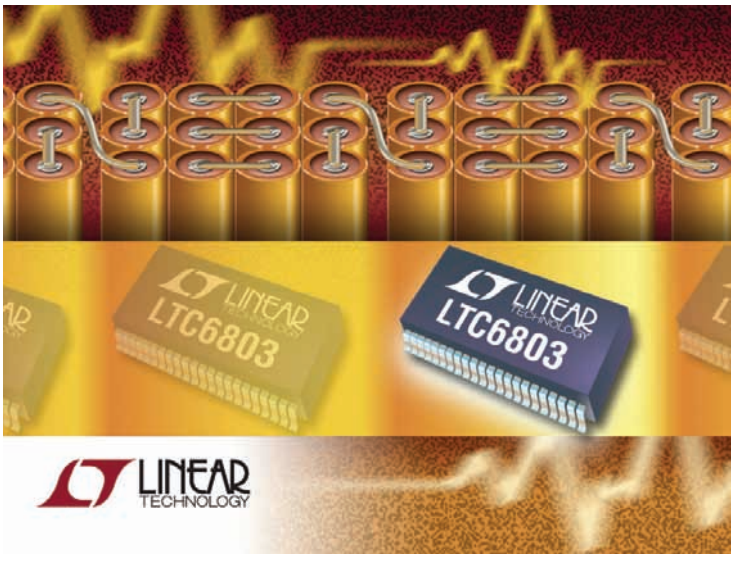
The **LTC6803** is a high-voltage battery monitor for hybrid/electric vehicles, electric vehicles, and other high-voltage, high-performance battery systems. The battery-measuring IC includes a 12-bit ADC, a precision-voltage reference, a high-voltage input multiplexer, and a serial interface. Each LTC6803 can measure up to 12 individual battery cells in a series. LTC6803s can be stacked in a series without optocouplers or isolators for precise voltage monitoring of every cell in long strings of series-connected batteries.

The maximum total measurement error of the LTC6803 is less than 0.25%, from –40° to 125°C. The battery monitor has a cell measurement range from –300 mV to 5 V, which enables it to monitor a wide range of battery chemistries and supercapacitors. Each cell is monitored for undervoltage and overvoltage conditions; an associated MOSFET discharges overcharged cells. The monitor features an onboard 5-V regulator, a temperature sensor, GPIO lines, and thermistor inputs for added functionality.

For long-term battery pack storage, the current consumed by the integrated BMS can potentially unbalance the cells. The LTC6803 addresses this concern through a standby mode that draws less than 12 µA. When powering from this input, the current draw on the pack is reduced to less than 1 µA.

The LTC6803 monitor is priced at **$9.95** each in 1,000-piece quantities.

Linear Technology Corp.
www.linear.com



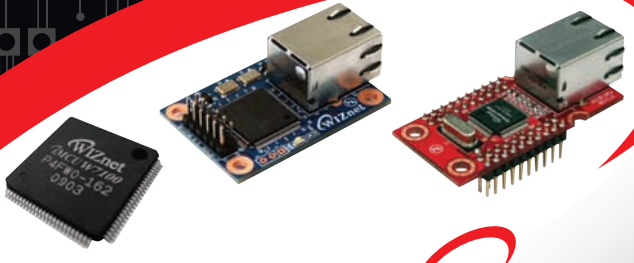
NPN

Making Your Devices
Smart & Simple!

Internet Connectivity Wizard

Hardware TCP/IP Offload Technology
- Offloading Protocol Processing from a System MCU
- Stable Hardwired TCP/IP Logic
- Silicon-proven IP for SoC and ASSP
- High Performance & Low Cost Platform
- No Booting Time
- Simple System Composition
- Best-fits OS-less Device

Applications
- Smart Grid and Smart Meter
- Medical Device Management
- Industrial Control
- Robot Control
- Security System



Get more information at www.wiznettechnology.com
3003 North First Street, San Jose, CA 408-232-5415
sales@wiznettechnology.com
WIZnet TECHNOLOGY

Imagine the possibilities...



Features:
- 100kHz Scope Input
- 20kHz AWG/Function Generator
- Built In Sensors/Outputs
  - Sound Waveform
  - Sound Level
  - Temperature
  - Light
  - RGB LED
- Analog Inputs
  - pH
  - Redox/ORP
  - Resistance
  - External Sensors
- 4) Configurable Digital IO's
- Oscilloscope and Data Logging Software

Starting at: $163.35

Designed for students, hobbyist to engineers alike. Makes an ideal gift for your children to get them interested in electronics. Numerous experimental kits also available.

For more information contact us at:
1-800-591-2796
http://www.picotech.com/pco450

pico® Technology

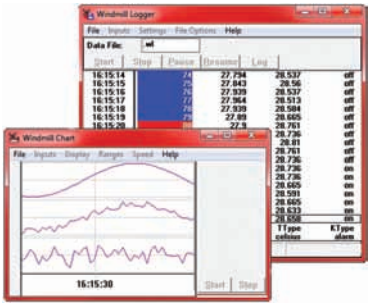
SERIAL DATA ACQUISITION SOFTWARE FOR WINDOWS 7

The **Windmill 7** data acquisition and control software suite for Windows 7 (64- and 32-bit), Vista, and XP can be used to read and control up to 10 devices over RS-232, RS-422, RS-485, and Modbus. Windmill software includes ready-to-run applications that can be used to chart and log data, control analog and digital outputs, monitor the COM port, and send data directly to Excel.

The software's COM port monitoring program, ComDebug, enables you to see all sent and received bytes including non-printing characters, such as carriage returns. This can be critical when managing binary data. You can control the state of the PC's serial port output lines and see the state of the input lines. ComDebug sends 16-bit two's complement integers, unsigned integers, single bits of data, and ASCII values to instruments.

Windmill software works with instruments communicating with ASCII or binary messages. It can collect readings from the majority of serial instruments that can be plugged into the computer's COM port, including PICs, power meters, digital indicators, multimeters, data loggers, GPS receivers, gyro compasses, motion sensors, and laboratory scales. You can mix and match equipment from many different manufacturers in many combinations.

The software suite costs **$80**. The price includes lifetime technical support and a money-back guarantee.



Windmill Software Ltd.
www.windmill.co.uk

WIDE VOLTAGE SERIAL AND PARALLEL F-RAM

Devices in the **W-Family of F-RAM memory** are available in serial I²C, SPI, and parallel communication interfaces, with an operating voltage range of 2.7 to 5.5 V. Performance enhancements include a 25% to 50% reduction in active current requirements and serial devices with faster power-up initialization.

The FM24W256 and FM25W256 W-Family memories are available in 256-Kb serial I²C and SPI, respectfully. The 64-Kb FM16W08 and 256-Kb FM18W08 devices feature a parallel communication interface.

W-Family products are built on an advanced, ferroelectric process. They offer NoDelay writes, 100-trillion high-endurance read/write cycles, and 38-year data retention.

The memories are available in industry-standard "green" packaging that includes an eight-pin SOIC for serial devices and a 28-pin SOIC for parallel devices. The W-Family is ideal for nonvolatile memory applications that require frequent or rapid writes or low-power operation. W-Family serial SPI devices offer full-bus speed writes and can operate over the industrial temperature range of –40°C to 85°C.

The FM24W256 and FM25W256 memories cost **$2.35** in 10,000-piece quantities. The parallel devices cost **$1.96** for the FM16W08 and **$3.48** for the FM18W08.



Ramtron International Corp.
www.ramtron.com

NPN

CAPACITIVE SENSOR SIGNAL CONDITIONER

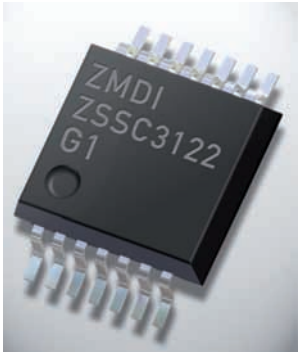
The **ZSSC3122 cLite device** is a capacitive sensor signal conditioner (SSC) featuring a wide-dynamic range. The chip delivers 14-bit resolution and 0.25% full-scale output accuracy. It features a built-in Sleep mode that lowers current consumption to less than 1 µA for temperatures up to 85°C.

The ZSSC3122 can interface with capacitive sensors from two to 10 pF with sensitivity as low as 125 atto-Farads (aF) per digital bit. It accommodates single and differential input sensor configurations. All calibration is digital and completed in one pass, which eliminates the cost of laser trimming and speeds the production of fully calibrated sensor modules.

The SSC connects to microcontrollers but also can be used in standalone designs for transducer and switch applications. Its mixed-signal design offers 14-bit capacitive-to-digital conversion and full 14-bit compensation of sensor offset. Sensitivity and temperature are measured through an internal digital-signal processor running a correction algorithm. Calibration coefficients are stored in an onboard, nonvolatile EEPROM. The system interface offers I²C, SPI, PDM, or alarm outputs.

The standard supply voltage is 1.8 to 5.5 V with accuracy up to 0.25% over a –40° to 125°C range. Programming and single-pass calibration are available through a standard PC environment using the ZSSC3122KIT development tools. The package includes TSSOP 14, a development board, a USB cable, and calibration software.

Unit prices start at **$4.54** in volumes of 1,000.



Zentrum Mikroelektronik Dresden AG (ZMDI)
www.zmdi.com

FOUR- AND 10-FINGER MULTITOUCH CAPACITIVE TOUCH-SCREEN ICs

The **MAX11855** and **MAX11871** are two new products in the TacTouch line of touch-screen controller ICs for resistive and capacitive touch technologies. The controllers offer four- and 10-point touch detection with sensitivity and noise immunity performance. They are capable of touch detection from a ballpoint/stylus and gloved hands.

The MAX11871 is a mutual capacitance touch-screen controller that can detect and track up to 10-finger simultaneous touch. All processing is included on-chip to output up to 10 X/Y-touch location coordinates, with a Z-pressure metric associated to each touch point, via a standard I²C interface to a host microcontroller. The controller's analog front-end is built from the ground up for capacitive touch and provides almost 60 dB signal-to-noise ratio performance, equivalent to a 1,000:1 ratio between touch and no-touch.

The MAX11871 also includes architecture to reject noise (by more than 40 dB) from external sources. This enables the part to be used in high LCD noise environments such as next-generation on-cell and in-cell touch screens.

For applications requiring only four-point touch detection, the MAX11855/MAX11856 implements a four-finger simultaneous mutual capacitive multi-touch solution that eliminates any ghosting effect.

Contact Maxim or a distributor for pricing.



Maxim Integrated Products
www.maxim-ic.com

NPN

HUMANDATA
FPGA Boards from Japan
• Basic and simple features
• Single power supply operation
• Same size/connector layout - ACM/XCM series
• Free download technical documents before purchasing
• High quality and highly reliable board from Japan
• Stocked items are ready to ship immediately
Feel free to contact us to customize speed grade or any features.
0-Series
2-Series
3-Series
1-Sereis
ALTERA FPGA boards
ACM-023 CycloneIV E F484 FPGA board
Cyclone IV E MRAM
EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Credit card size (86x 54 mm)
RoHS compliant
EDA-004 Cyclone III USB-FPGA board
Cyclone III USB Config. USB Comm. MRAM
EP3C55F780C8N
FPGA configuration via USB interface
Credit card size (86 x 54 mm)
RoHS compliant
XILINX FPGA boards
XCM-018 Spartan-6 FGG484 FPGA board
Spartan-6 MRAM DDR2
XC6SLX45-2FGG484C
XC6SLX75-2FGG484C
XC6SLX100-2FGG484C
XC6SLX150-2FGG484C
Credit card size (86 x 54 mm)
RoHS compliant
EDX-006 Virtex-5 USB-FPGA board
Virtex-5 USB Config. USB Comm. MRAM
XC5VLX30-1FFG676C
FPGA configuration via USB interface
Credit card size (86 x 54 mm)
RoHS compliant
XCM-107 Virtex-5 LXT FPGA board
Virtex-5 Rocket IO
XC5VLX30T-1FFG665C
XC5VLX50T-1FFG665C
Compact size (43 x 54 mm)
RoHS compliant
Over 100 varieties of FPGA/CPLD boards are available.
See all our products at:
www.hdl.co.jp/CC/
HuMANDATA LTD.
E-mail: s2@hdl.co.jp
Fax: 81-72-620-2003

HUGE discounts
on select RIGOL equipment
- for a limited time only!

DG1022 - 20MHz Function/Arbitrary Waveform Generator
Was $695 Now only $499

DS1052D - 50MHz Mixed-signal Scope 2ch + 16 Digital Channels
Was $1,095 Now only $899

DS1102D - 100MHz Mixed-signal Scope - 2ch +16 Digital Channels
Was $1,495 Now only $1,099

DG1022 + DS1052E
Basic Signal Analysis Package

20MHz Function/AWG + 50 MHz 2-Ch Digital Oscilloscope
Was $1,094 Now only $799

DS1102CA + DG2041A
Standalone Signal Analysis Package

40MHz Function/AWG + 100 MHz 2-Channel Digital Oscilloscope
Was $2,290 Now only $1,799

Hurry! This sale is only for a limited time!
Call us at 888-772-3544
or go to www.saelig.com today!

Love Saelig's Unique Electronics?

unique geek

Get a free "unique geek" t-shirt with your next $50 or more purchase!
(See website for details)

Saelig.com unique electronics
Follow us:

PRECISION HIGH-TEMPERATURE INSTRUMENTATION AMPLIFIER

The **AD8229** is a 1.0-nV/√Hz high-temperature, low-noise instrumentation amplifier that performs at up to 210°C in extreme environments, such as down-hole drilling, jet engines, and space exploration. The AD8229 instrumentation amplifier is specifically geared toward engineers who design electronics for extremely hot, harsh, and hostile environments. Because it is built to withstand these environments, the amplifier helps improve maintenance and increase operation lifestyle.

The amplifier is designed to perform in high-temperature applications, including common-mode rejection ratio (CMRR), voltage offset, and input noise. Its CMRR performance is 100 dB above the temperature range of 55° to 210°C. The AD8229's input voltage offset varies by only 100-nV/°C above the entire temperature range. Input noise is 1.0-nV/√Hz, so it is well-suited for sensors that read faint signals, such as those used in down-hole drilling, nuclear magnetic resonance spectroscopy, and gamma-ray detection.

The AD8229 amplifier is priced at **$150** each in 1,000-piece quantities.



Analog Devices, Inc.
www.analog.com

SEVEN-CHANNEL HIGH-SPEED MODBUS DAQ MODULES

The **D6000 series** of seven-channel analog input smart data acquisition modules communicate through the Modbus RTU protocol with RS-485 output. The series enables sensors to be interfaced to any computer- or processor-based equipment containing a serial port. The modules measure voltage (±25 mV, ±50 mV, ±100 mV, ±1 V, ±5 V, ±10 V), current (±20 mA), and thermocouples (J, K, T, E, R, S, B, C). A universal module measures all of the above.

Up to 247 modules or 1,729 channels can be strung on a pair wired to a serial port on a host computer. The measurement resolution is 16 bits. Accuracy is ±0.05% of FS max. The conversion rate is 8/s to 25/s. The input to output isolation is 500 V_{RMS}. The input burnout protection is 250 VAC. Transient suppression on all RS-485 lines is provided. The modules also feature a customer-programmable digital filter. All scaling, linearization and calibration, and communications values are stored in EEPROM, which eliminates the need for pots or switches. The sensors' communication features include channel address, data rate (9,600 to 115 K) parity, and checksum.

The series also features a two-channel analog output module with 12-bit DAC resolution, programmable voltage, or current outputs at 250 conversions per second. Modules with 15-bit digital input and digital output are also available.

D6000 series modules are packaged in a 6" × 4" × 1.5" ABS case with plug-in screw terminal connectors. Each module requires 10 to 30 VDC unregulated supply. Rated performance is specified from –25° to 70°C.

The D6000 series costs **$245** to **$375**.



DGH Corp.
www.dghcorp.com

NPN

Test Your EQ

Edited by David Tweed

Problem 1—What does the following C function compute?

```
int foo (int n)
{
  int k = 0;
  int t;
  while (t = n & -n) {
    ++k;
    n &= ~t;
  }
  return k;
}
```

Problem 2—What advantage does this function have over a more "conventional" implementation?

Problem 3—A 15-megapixel consumer-grade digital camera is being used for aerial photography. The lens being used on it gives it a 22° horizontal (cross-track) field of view. When flying at 10,000' above the ground, what is the approximate per-pixel resolution?

Problem 4—While not an engineering question *per se*, here's a problem where "thinking outside the box" can help lead to a solution.

It's fairly straightforward to create the number 55 by using the digit 4 five times:

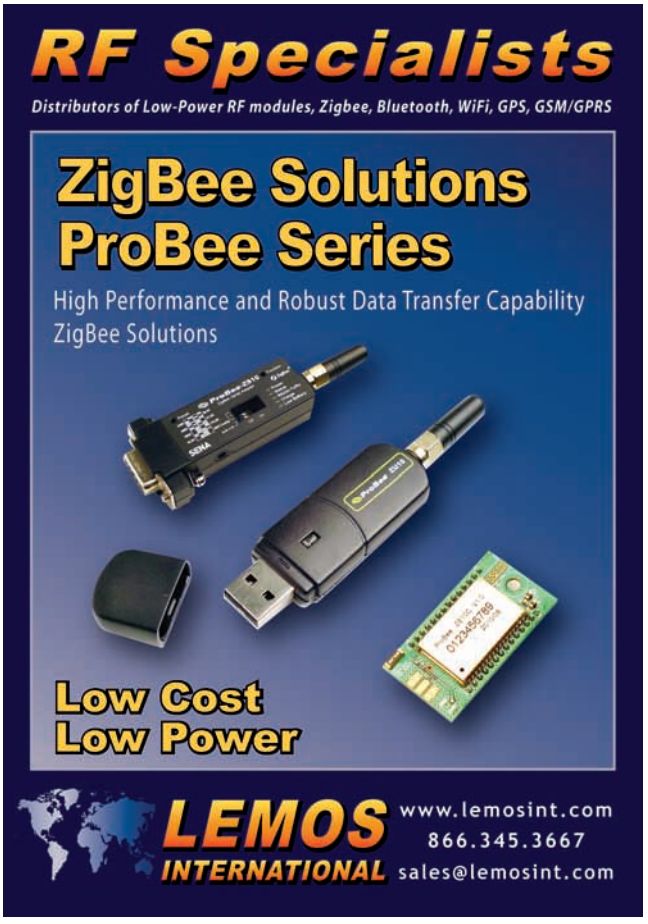
$$44+\frac{44}{4}=55$$

Can you create the number 55 using the digit 4 just four or fewer times? You may use any mathematical operation, but no other digits.

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





by George Novacek (Canada)



Functional Upset

Functional upset occurs when an embedded controller stops working as designed. To minimize the impact of such a problem, you should follow professional wiring practices and create system topologies in a thoughtful, systematic fashion.

In my January 2011 article, I covered the topic of how to protect electronic circuits from electromagnetic pulses (EMPs), and particularly those encountered as indirect lightning effects (*Circuit Cellar* 246). I mentioned two areas of interest: the physical robustness of circuits, known as damage tolerance, and the effects of lightning on the system performance resulting in functional upset. In this article I'll revisit the topic of functional upset.

EMBEDDED ISSUES

What is functional upset and what can we do about it? Every embedded controller has at least one input and one output, with the output being a function of the input in some predefined way. When, due to interference, the embedded controller stops working as designed, functional upset has occurred.

Consider Figure 1a, a topology of many control systems. Electromagnetic fields and shared return currents induce unwanted signals into the loops as shown. The transient protection inside the controller limits the input signal excursions to prevent damage to the electronics. But the interference, albeit clamped to a safe level, remains superimposed on the input signal, which is thus distorted, even completely obliterated. Functional upset is the result.

Many problems can be minimized, even avoided, by thoughtful system topology. Fixes after the fact are costly and often not very effective. It is important that each signal and its return use their own twisted and preferably shielded pair as shown in Figure 1b.

Interference will continue to generate differential signals in any asymmetric input in Figure 1b, but eliminating returns through the chassis and common paths will significantly reduce its effects. I have seen voltage differences between two points on a chassis, presumably on the same potential, as much as 200 V, from DC to microwaves. Figure 1b is an example of effective topology to minimize the effects of magnetic fields and ground noise.

INTERFERENCE SUPPRESSION

While the embedded controller designer can protect his circuits from being damaged, ensuring the signal integrity with good wiring practice and protecting of the peripheral devices are the system designer's responsibility. They both may be the same person, but system design must always come first.

Harnesses using twisted pairs laid close to the chassis go a long way toward robust design. Having each twisted pair shielded is even better, and generally necessary for electromagnetic compatibility (EMC). For effective electromagnetic interference (EMI) suppression, the shielding grounded at least at both ends is recommended. This is not such a good idea for high-intensity fields such as those generated by lightning effects. The shield grounded at both ends forms a one-turn loop with potentially huge currents circulating through it. In the end, an engineering tradeoff must be made. It is not unusual to see double-braided harnesses with the outer shield grounded at the controller end only and the inner braid grounded at both ends.

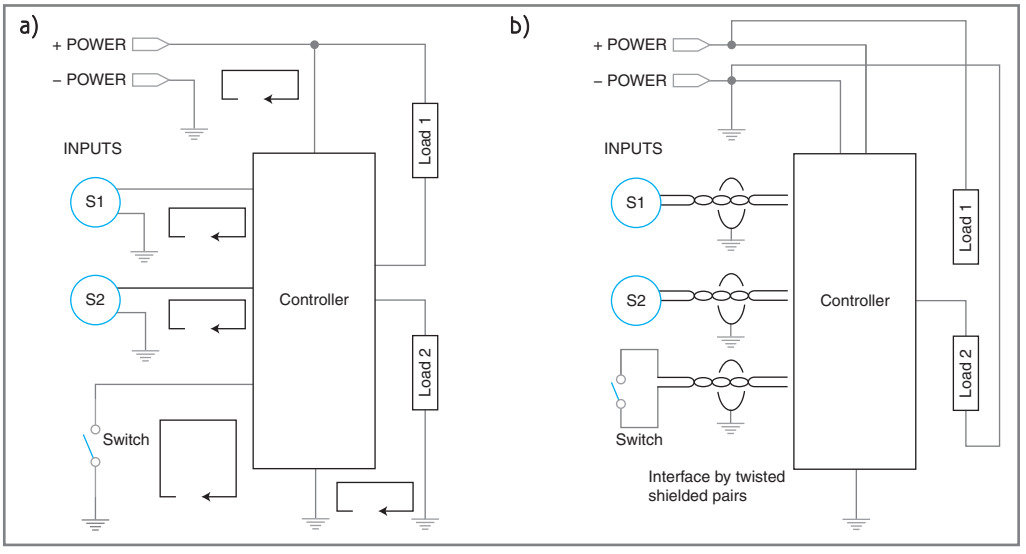


Figure 1—Two typical system wire layouts

A system designer must analyze the working environment and design the harnesses accordingly. Transfer impedance is a characteristic commonly used to calculate the shielding effectiveness and the resulting impact on the interference levels the embedded controller will experience. Once you've defined the interference levels to be seen inside the controller, you can consider strategies to deal with the system upset.

A simple and common approach, if accepted by the system safety analysis, is to respond to the upset by turning off the loads. You need to confirm that system upset has in fact occurred—that's the job for the built-in test (BIT). Input signals are tested for plausibility, firmware execution is monitored, and so on. For example, upon detection of an invalid condition, an aircraft nose wheel steering controller would disconnect hydraulic pressure from the actuators, causing the nose wheel to free castor—a behavior similar to the front wheels of a shopping cart. Directional control is then maintained by differential braking or differential thrusting of the engines. When the interference has ceased, the controller either detects clear conditions and reasserts the control automatically or it is manually reset.

Another tactic is to hold the last good data until more good data become available again. It's just like driving into a fog patch: you hold steady until you emerge from it. This scheme can be used with a full-authority, digital-engine controller (FADEC). You don't want engines in flight to suddenly stop. Duration and subsequent recovery of the system from the hold mode is monitored, and delayed recovery results in another previously determined action are designed to satisfy the system's safety.

The third method is ideal, but often the most complicated and costly to achieve. The system is designed to ignore hiccups, like lightning strikes, and keep working as if nothing happened. In a really tough environment, your only option may be to use metal conduits for wiring. The fundamental basis for making electronics impervious to interference is the inherent immunity of symmetrical I/Os. The interference induced into the twisted pairs results in a common mode signal. Unfortunately, any imbalance in the circuits will degrade the immunity and result in the interference appearing as a differential signal, too. Therefore, the careful design of the system and the circuits is critical.

A transformer interface is simple and effective, with good isolation and common-mode input range. It is often used for communications, such as Ethernet or the military MIL-STD-1553B. The main disadvantage of a transformer interface is limited bandwidth and the cost when relatively low-volume protocols are concerned. Some transformers can cost hundreds of dollars each.

Fiber optics provide great isolation, inherent immunity from electromagnetic interference, and huge bandwidth. On aircraft where weight is always a concern, fiber optics, weighing less than a copper wire bundle, are finding their way into places such as entertainment consoles.

Discrete as well as DC and low-frequency signals can be successfully interfaced with optocouplers. They provide isolation and common-mode range in kilovolts, but some authorities consider optocouplers to have poor reliability and long-term stability. In my many years in engineering, I have never encountered that problem. Similarly, linear optocouplers had been discouraged from being used in critical applications, but that view also seems to be changing.

For precision and characteristics not achievable by other methods, instrumentation and isolation amplifiers are the answer. It is important to characterize the conditions that the amplifiers will be exposed to. Only then can we make sure that the worst-case scenarios will be handled without distortion or even destruction. Calculations and simulations will, at best, put us in the ballpark, but the test is the ultimate proof. We can build differential stages to accept hundreds of volts of common-mode signal; but personally, I would consider methods such as the transformer or optocoupling first in order to avoid the risks associated with galvanic connections.

MORE ON ROBUST DESIGN

Robust design is a goal toward which we all must aspire. We have merely scratched the surface of robust system design issues. Suppressing and managing interference is one piece of the puzzle. I'll cover more techniques in future articles.

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.*

by Richard Pierce (USA)

Stereo Spectrum Analyzer

There are plenty of spectrum analyzers on the market. But you'll likely need two of them to see both channels of a stereo signal. One solution is to build an analyzer to suit your needs. This design features two integrated channels of display, each with a 16 × 16 LED bar graph.

Audio spectrum analyzers are widely available. Even though there's nothing wrong with the available spectrum analyzers, I created a stereo, 2,048-bit FFT design that features a 16 × 16 × 2 multicolor LED, automatic brightness, and a multiple-display mode. My design gives you two integrated channels of display, each with its own 16 × 16 LED bar graph: perfect for stereo.

I built a couple of different-sized versions of this spectrum analyzer, one using small and one using medium-sized LED arrays. Both of these versions use the same core control CPU and audio signal processing. They just use a different LED display component (see Photo 1).

The overall design is conceptually simple (see Figure 1). An input section adjusts the stereo audio signal level to an appropriate value for a microprocessor's A/D converter. The microprocessor performs an FFT analysis to convert the audio signal time domain to the frequency domain and then displays the results on a pair of 16 × 16 LED arrays. The display shows the signal level for both stereo channels from about 25 Hz to 20 kHz in 16 bands each.

ANALOG INPUT

The analog input is designed to work with a "line-level" stereo signal, which can vary from 0.5 to 2 V_{PP}, depending on the standard. The circuit is designed to work with any signal in this range (see Figure 2).

The input section is a simple variable gain amplifier that AC couples the line input to the analog inputs of a Microchip Technology dsPIC processor. The gain setting is controlled by a dual digital potentiometer, which is connected to a couple of buttons (gain Up, gain Down) through the dsPIC. Resistors R11/R22 and R12/R21 attenuate the input signal and center it at a level that's half of the supply voltage. The dsPIC's analog inputs require this center level in order to use the A/D converter's signed integer conversion mode. After attenuating the input, R5/R33 in combination with a Microchip Technology MCP4261 dual digital potentiometer determine the gain setting of the audio quality

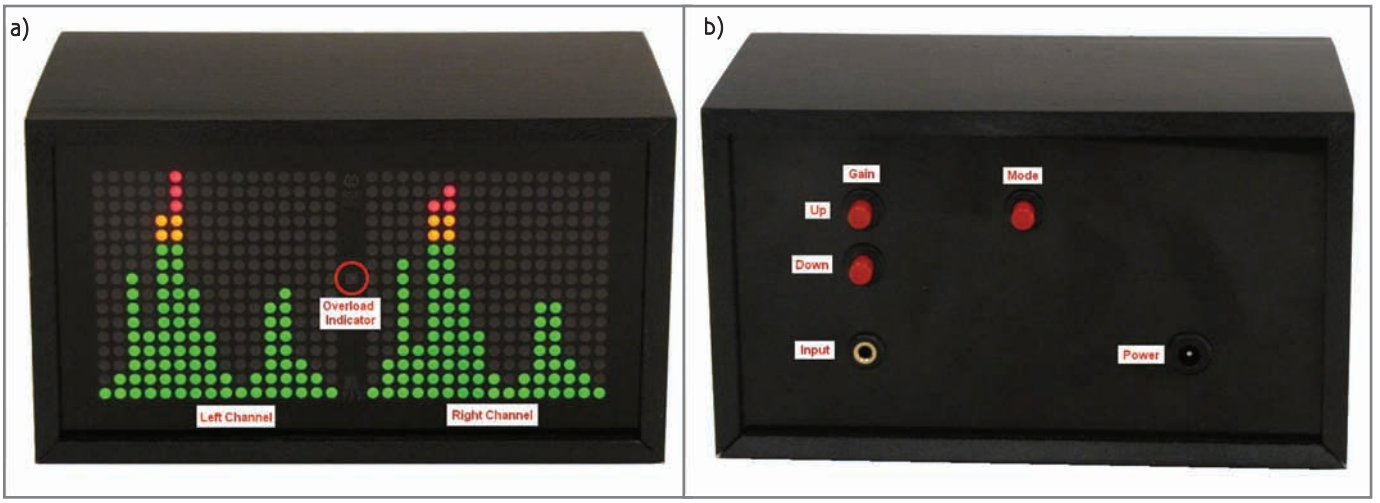


Photo 1a—This is the front of the spectrum analyzer. b—This is the back panel.

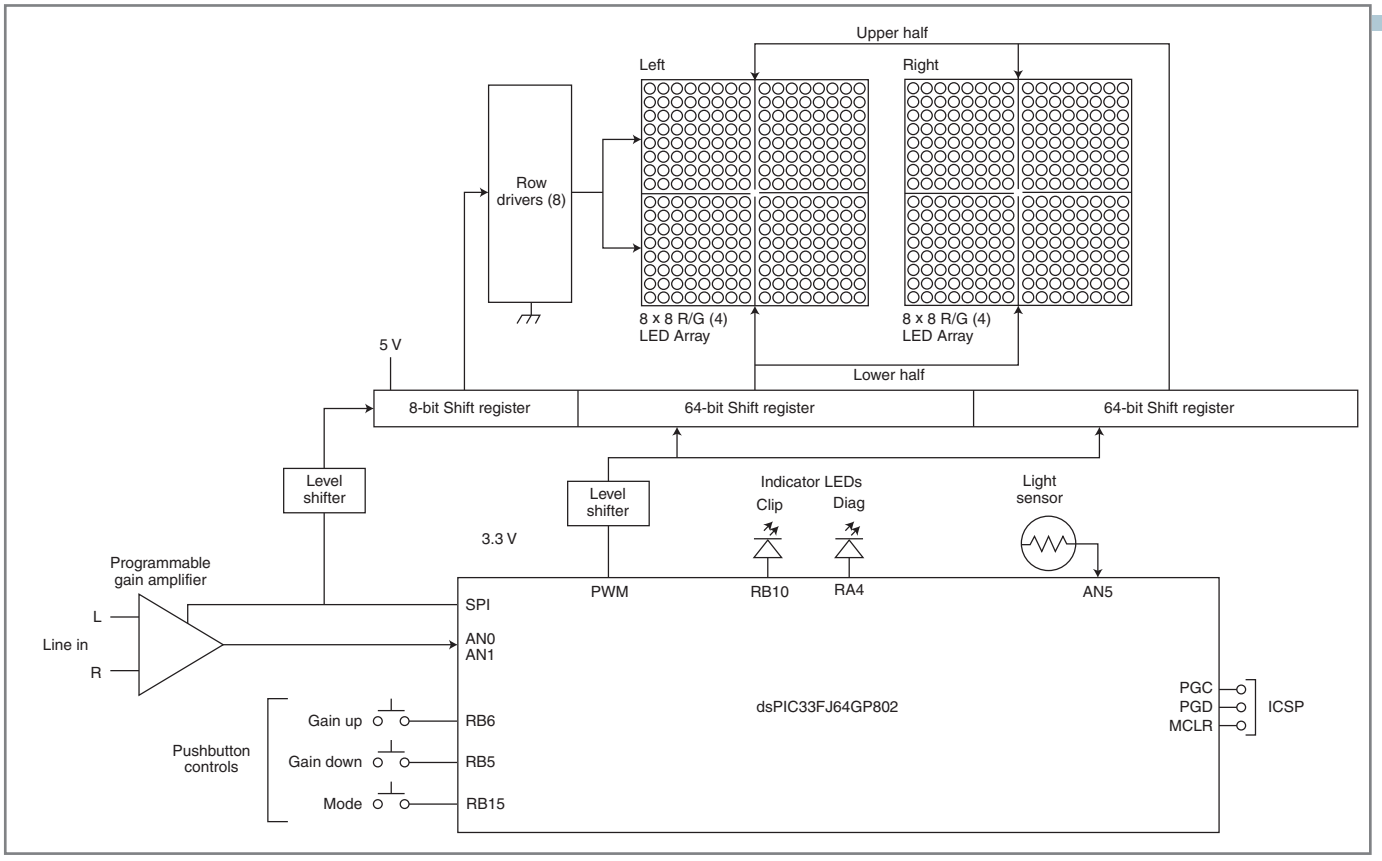


Figure 1—The spectrum analyzer is designed around a dsPIC33FJ64GP802 processor.

MCP6244 op-amp. This op-amp is pretty hard to kill, but diodes D1 and D2 are there anyway to provide additional protection from large transient or inadvertent input signals. The same gain setting is used for both of the stereo channels, and the dual digital potentiometer resistance is matched to within 0.2% (typical), so no channel gain balancing adjustment is needed.

Because there is no low-pass filter in the input section, input signals with frequencies higher than about 22 kHz will confuse this spectrum analyzer. On the other hand, not having a filter means that the input signal is not distorted at all, so accurate results are easy to achieve.

ANALYZER DISPLAYS

The primary spectrum analyzer displays are 8 × 8 red/green LED arrays (see Figure 3). I found out the hard way that the LED array brightness varies a lot with LEDs from different manufacturers, so there are some conditional code switches to select the type of display you have and use an appropriate brightness control range. When you buy LEDs (of any type), you should always buy them in complete batches to ensure a uniform brightness.

Four 8 × 8 LED arrays are used for each channel (a total of eight arrays) so that each channel display is 16 × 16. Each LED pixel can be red, green, or orange (both red and green). To drive the displays, a multiplexing scheme must be used. Rather than doing it the obvious way and driving 32 columns of 16 rows, the displays are organized as 64 columns of eight rows. Since each LED requires 2 bits (red and green), the result is a total of 128 columns that must be driven. A 136-bit shift register is used (128 column bits plus eight row bits) to enable the controller to talk to the display using one serial peripheral interface (SPI) port. Because of the large number of display chips

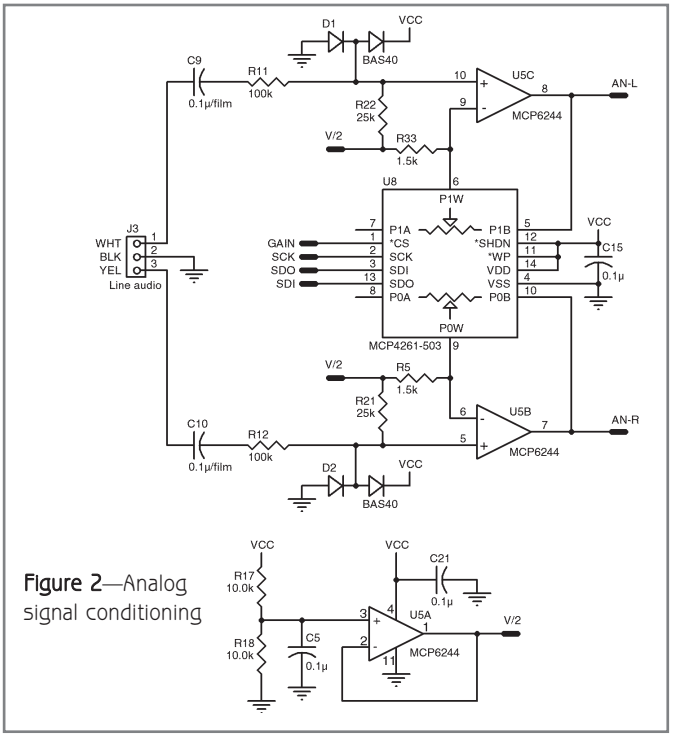


Figure 2—Analog signal conditioning

and long PCB traces, a relatively low SPI clock rate of 2.5 MHz is used here. Every millisecond one row (136 bits) is clocked out to the display, resulting in a net display refresh rate of 8 ms (125 Hz). This is a high enough refresh rate that there is no visible flicker in the display.

The 74VHC595 shift register chips can provide enough current to drive the LED columns directly because only one LED in each column can possibly be on at a time. But, since each row can have up to 64 LEDs on at a time and two rows are being driven simultaneously, MOSFET row drivers are used (see Figure 4). Turning on all the LEDs at once would overload the 1-A power supply and exceed the MOSFET current ratings, so you don't want to do that (see Figure 5). The maximum current that's actually needed stays reasonably low due to the display organization and by not providing any display modes that use the orange color very much (since an orange pixel requires twice the current of a red or green pixel). The column shift registers are also connected to the dsPIC's PWM output, which enables the display's overall brightness to be controlled. A photocell (actually an integrated light sensor and amplifier) is used to detect the ambient light level, and the display brightness is automatically adjusted accordingly.

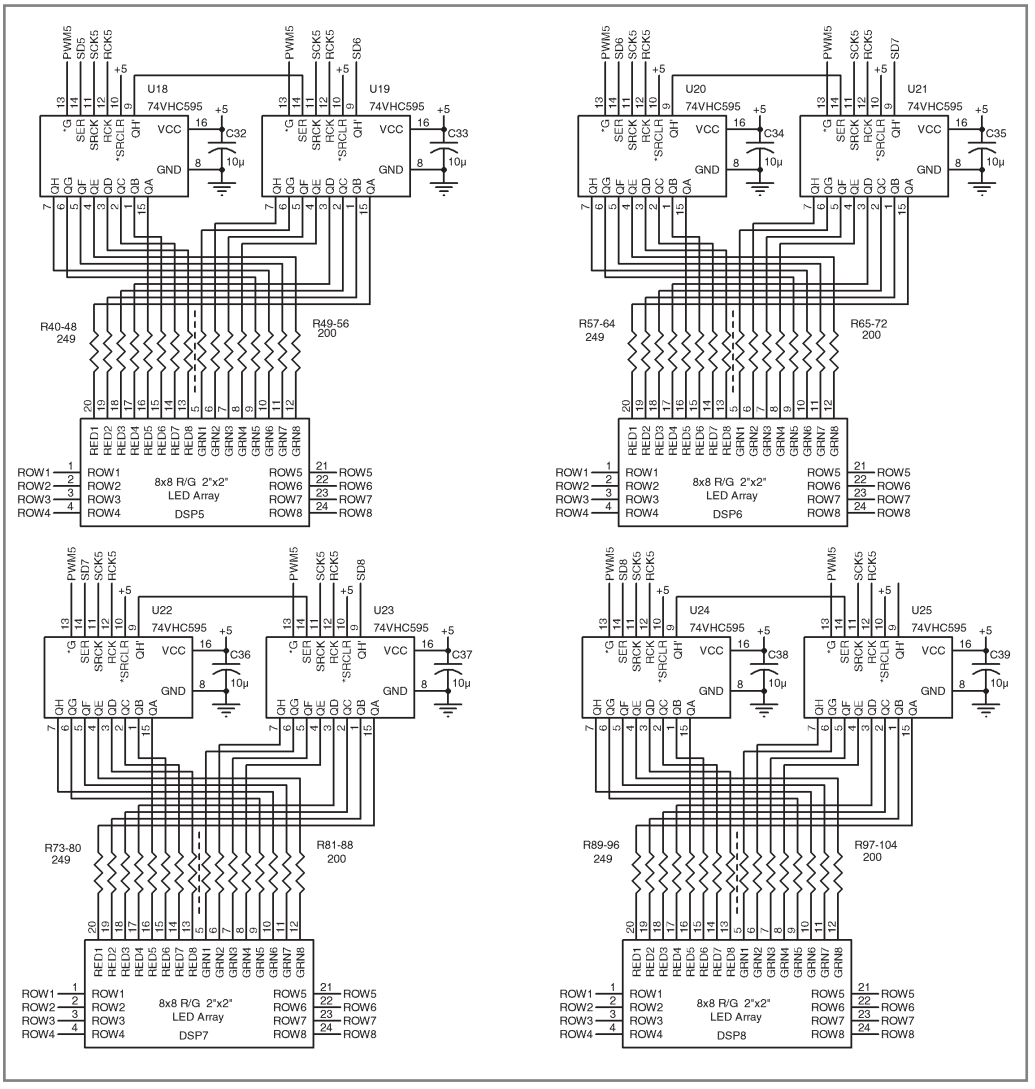


Figure 3—The spectrum analyzer's right-side display

The display is operated at 5 V, which enables it to run directly from the 5-V main bus and makes the 74VHC595 shift register chips able to provide more LED current than using 3.3 V would. However, the dsPIC requires a 3.3-V power supply, so some level of conversion is needed to talk to the display chips. This is done by the 74HCT365 hex buffer chip, which operates at 5 V but has TTL-level inputs that are compatible with the 3.3-V CMOS signals from the dsPIC.

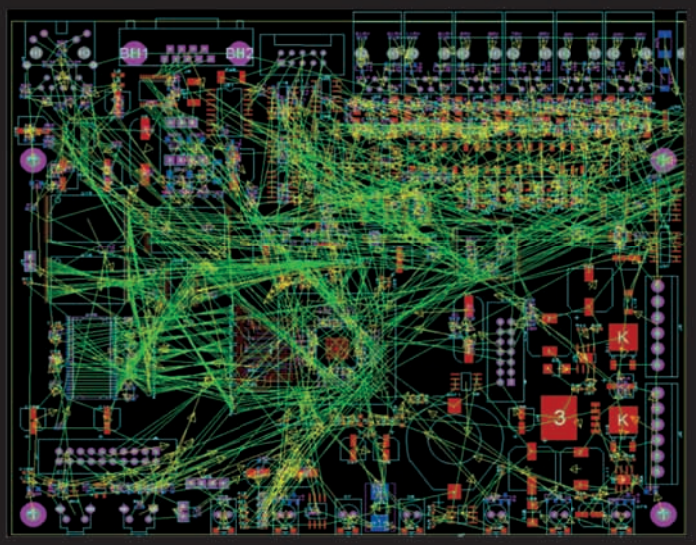
A spare op-amp is used to generate a signal that keeps the display turned off during power-up. This prevents the possibility of large currents being drawn by random data in the driver chips that might turn on a lot of LEDs at the same time. Once the power-up is complete and the display chips have been initialized, the display is enabled.

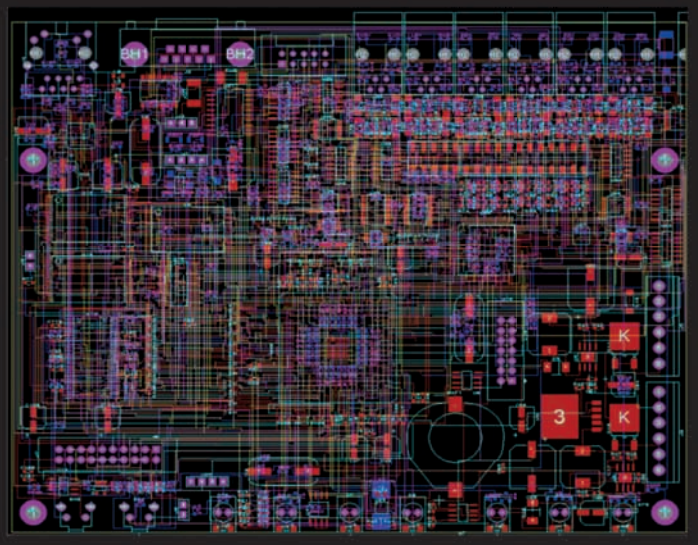
One more input button to the dsPIC selects the display mode. A dozen display modes are available. The Mode button is used to select the desired one. The display orientation can be horizontal (the classic style for spectrum analyzers) or vertical, which is really nice for a stereo display (see Photo 2). In each orientation, there are several choices for the colors and styles that will be used. The style choices are bar/dot, fast/slow response, and normal/peak decay. The color choices are red, green, orange, and VU style (which is green with orange/red at the top).

An additional LED indicates when input overload is occurring. In the context of this spectrum analyzer, overload means that the FFT analysis algorithm is encountering saturation (mathematically overflowing) and the signal is being clipped. The FFT algorithm will continue to operate even when saturation is occurring, but too much clipping will eventually confuse the algorithm and cause inaccuracy in the display. Overload can be resolved by adjusting the gain setting downward or, if necessary, by adding a simple attenuator to the input signal to bring it down to the maximum level of 2 V_{pp}.

The "bug" LED is a diagnostic indicator. This really is

Our new autorouter will turn this... into this...





in less than 5 minutes!

All levels of the Proteus Design Suite now include a world class fully integrated shape based autorouter at no additional cost.

The Proteus Design Suite also incorporates:

- Professional schematic capture
- Publication quality schematics
- Fully configurable bill of materials
- Large component libraries for both simulation and PCB layout
- Mixed mode SPICE circuit simulation
- Co-simulation of PIC, AVR, 8051 and ARM7 microcontroller firmware
- Automatic component placement and gateswap optimization
- Highly configurable design rules
- Interactive design rule checking
- Polygonal and split power planes
- RS274X, Excellon and ODB++ database export
- 3D Viewer with 3DS, IDF and DXF export



Prices start from just $249* - visit our website for full details or to download a free demo.

labcenter Electronics www.labcenter.com

20 Years

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

*exc. tax & delivery

the last LED. No kidding this time. In addition to the various program logic error traps that are sprinkled throughout the code, the dsPIC has an error-trapping mechanism for conditions such as oscillator failure, memory access errors, and stack overflow. If any errors occur, the spectrum analyzer application tries to shut down everything (enter a fail-safe condition) and then blinks an error code on the diagnostic LED. This code is just a number of blinks, from one to 11, that repeat forever.

CONTROL

A Microchip Technology dsPIC33FJ64GP802 processor is at the core of this spectrum analyzer design. This dsPIC has enough memory and processing power to perform the spectrum analysis on two audio channels and update the display at a rate of about 12 updates/second. The display update rate is determined by the sampling rate and sample buffer size required for the FFT, not by a processor power limitation.

The dsPIC will operate at up to 40 MIPS, but because it's got a 16-bit architecture and has instructions that are optimized for FFT types of calculations, it gets everything done for this application while operating at only 10 MIPS. A crystal-based oscillator is used so that the frequency analysis results are highly accurate. The dsPIC is programmed via ICSP, and the serial port (which is not used by the spectrum analyzer) is brought out for use as a potential diagnostic port.

TWIDDLES & BUTTERFLIES

This is the fun part, as there is a lot going on in the software. I programmed primarily in C language (using a Microchip Technology C30 compiler) with some sections done in Assembler for performance reasons.

The dsPIC33FJ64GP802 has a built-in DMA capability that can be used with the A/D converter. This is utilized by the spectrum analyzer application to read analog data from one of the stereo input channels while simultaneously performing the FFT data analysis on the other stereo channel. Ideally, it would read and analyze analog data from both channels continuously, but the dsPIC doesn't have quite enough RAM to do that. So, this spectrum analyzer is actually looking at the analog signal from each channel only half of the time. A timer kicks off each A/D read at a 50-kHz sampling rate. The DMA module is triggered when each A/D cycle completes and it automatically transfers the A/D result to the DMA buffer RAM and increments the DMA address. This happens completely in the background, so 100% of the dsPIC processing power is still available to do other things (e.g., an FFT analysis).

The dsPIC has a total of 16 KB

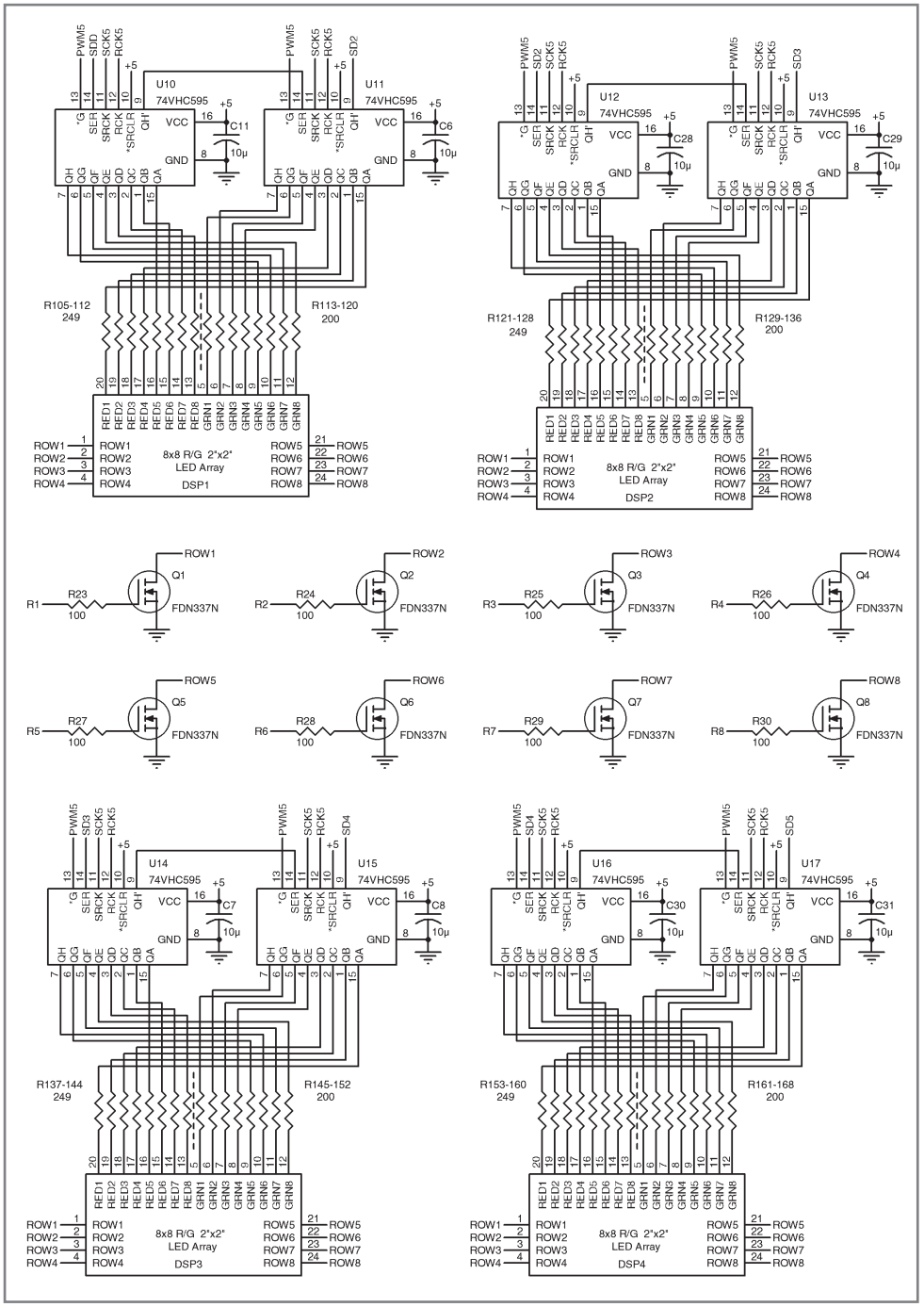


Figure 4—Schematic diagram of the spectrum analyzer left-side display and row drivers

of RAM. Of this total, 2 KB is dedicated to DMA use, 4 KB is needed to store one sample period, an 8-KB buffer is needed to perform the 2,048-bit FFT, 2 KB is used for storing some FFT constants, and some RAM is needed for program variables and stack. So, a little fancy footwork is needed to make everything fit into RAM. First, the DMA starts filling its 2-KB buffer with data from one of the stereo channels. When the buffer is half full, the first half is transferred to a temporary buffer (at the same time as the DMA continues to fill the second half of the buffer). When the second half of the buffer is full, the ADC is switched to the other stereo channel, a new buffer acquisition cycle is started (on the first half of the DMA buffer), and the second half of the DMA buffer is transferred to the FFT buffer. The temporary buffer is then combined with the FFT buffer, so the temporary buffer is now free, and there is 4 KB of data in the FFT buffer ready to be analyzed. The FFT analysis is now able to run while the DMA collects the next set of data into the DMA and temporary buffers. The FFT analysis process will use another 4 KB of RAM for a total of 8 KB when it's done.

The FFT analysis code is based on the algorithm described by Steven W. Smith in *The Scientist and Engineer's Guide to Digital Signal Processing* (Chapter 12, "The Fast Fourier Transform"). I also used some stock Microchip Technology FFT library subroutines and an integer square root algorithm that I found online. These were adapted for the dsPIC and optimized somewhat to reduce the RAM requirement. The sampling rate is 50 kHz, which allows for a maximum effective input frequency of about 22 kHz. A 2,048-bit FFT is used, which gives a frequency resolution of about 25 Hz. Refer to past issues of *Circuit Cellar* for more information about FFTs.

Once an FFT analysis is complete, the results for display are calculated. The FFT output is 2,048 "bins" of equally spaced frequencies, where each bin gives the amplitude of the signal at a particular frequency. But the desired display is not linear, so different sized groups of bins are totalized and averaged to determine each display column value (which is a representation of the total signal magnitude in the group frequency range). The bins to use for the frequency groups are set up in a table so they can be easily changed (e.g., if you want to have a linear frequency spectrum display instead). For audio signals, most of the activity is in the lower end of the frequency range, so more bins are combined into each group at the higher frequencies. The number of bins in one group varies from one bin at 24 Hz to 529 bins at 16 kHz. This spectrum analyzer display is set up to show frequencies on an exponential scale (see Table 1).

After all of the 16 column values are determined, and depending on the display mode, the data is further modified to implement the fast/slow response, peak decay, and the orientation of the data that's finally displayed. Each output column value is fed through a "display charset" that is established based on the current mode setting. The display charset is a translation table that tells which LED(s) should be turned on for a particular column display value. The column display value can be 0 to 16, and each column is 32 LEDs (16 red and 16 green), so there are 17 entries of 32 bits in each charset. If the display orientation is vertical, the display data is fed through more code that translates it from horizontal to vertical. Once that's done, the display data is moved off to another buffer where the display refresh timer interrupt simply reads it and transfers it out via the SPI to the display chips.

Between the audio data buffer acquisition cycles, the ADC is switched over to read the photocell and then the display brightness (PWM value) is set accordingly. There are minimum and maximum limits applied to the PWM value so that the display is not too bright and it's never completely off.

Periodically (every 1/50 of a second), the push buttons are polled and debounced. Button inputs are queued up so that they are handled at an



Photo 2—Some of the spectrum analyzer's display modes

appropriate time. This dsPIC doesn't have a built-in EEPROM, so any permanent settings would have to be stored in flash memory and you'd have to worry about flash memory write endurance limits. However, the MCP4261 digital potentiometer has a small amount of EEPROM, so the settings (gain and mode) are stored there. Any time a setting is changed, the settings are automatically rewritten to the EEPROM after five seconds. The EEPROM is good for about 1 million writes, so endurance shouldn't be a problem.

Column	Center frequency	Frequency span
1	24 Hz	37–61 Hz
2	49 Hz	37–61 Hz
3	73 Hz	61–85 Hz
4	98 Hz	85–110 Hz
5	122 Hz	110–134 Hz
6	159 Hz	134–183 Hz
7	220 Hz	183–256 Hz
8	317 Hz	256–378 Hz
9	488 Hz	378–598 Hz
10	781 Hz	598–964 Hz
11	1,270 Hz	964–1575 Hz
12	2,185 Hz	1,575–2,795 Hz
13	3,772 Hz	2,795–4,749 Hz
14	6,213 Hz	4,749–7,678 Hz
15	9,875 Hz	7,678–12,073 Hz
16	16,036 Hz	12,073–20,000 Hz (actual limit is approximately 22 kHz)

Table 1—This is a frequency display table. Input frequencies are shown in each display column.

All the various software tasks are synchronized and interlocked so the SPI, memory buffers, and display content integrity are maintained. For example, the display is the SPI's priority user, but the digital potentiometer uses it too, so communication with the digital potentiometer is limited to times when the display is guaranteed not to be using the SPI.

CODING NOTES

The FFT "twiddle" constants, Hamming window constants, display charsets, and display rotation code were all created using script-generator programs. The display charset logic is pretty obvious and could be tweaked by hand, if desired. The generator code for the twiddle and window constants is included in the source files. The display rotation code, however, is magic. The process of rotating the display was so confusing that I ended up creating the rotation code empirically.

You may notice that there are dual (.h and .inc) files for the "p33FJ64GP802" and "sa" include files. These are for use by C and Assembler source files, respectively. These files must be kept in sync for the application to compile and run correctly. For example, if you want to change the CPU speed, it must be changed in both the sa.h and sa.inc files, and it must always be the same value in both files.

For performance reasons, some data tables are loaded from flash memory to RAM at the beginning of the program. If you run low on RAM, these tables can be used directly from flash memory since there is plenty of spare processing time available. Accessing data from flash memory is only a little bit slower than accessing it from RAM. And even at only 10 MIPS, the processor is idle about 50% of the time.

The code is configurable to run at 10, 20, or 40 MIPS. So, if you add features and need more processing power, all you have to do is change the speed constants in the .h and .inc files. The power supply has enough spare capacity to handle all processor speeds up to 40 MIPS (the maximum speed for this processor).

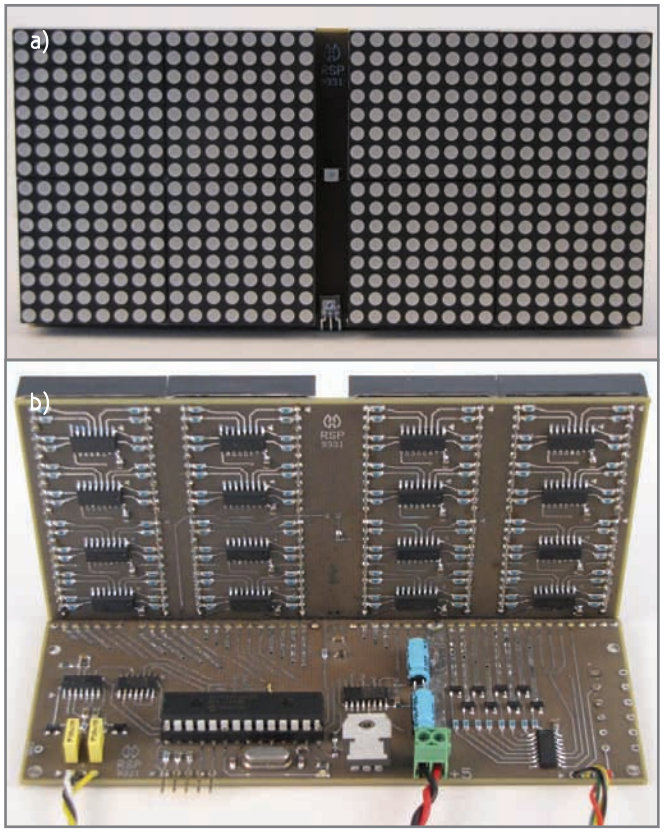


Photo 3—The front (a) and back (b) of the spectrum analyzer's PCB

CONSTRUCTION

The spectrum analyzer's electronics are built in two sections: a base PCB and a display PCB (see Photo 3). The base PCB contains all of the control logic, analog signal processing, and the 3.3-V power supply. The display PCB contains the column drivers and the LED arrays. It mounts vertically, attached to the horizontal base PCB using a right-angle header.

I chose a four-layer type of PCB primarily for the display PCB requirements. Using four layers pretty much guaranteed that there would not be problems with noise, even with the high currents the display can draw. Using four layers for the base PCB probably wasn't necessary, but it didn't hurt and it made the layout process a lot easier. A four-layer board costs only a few dollars

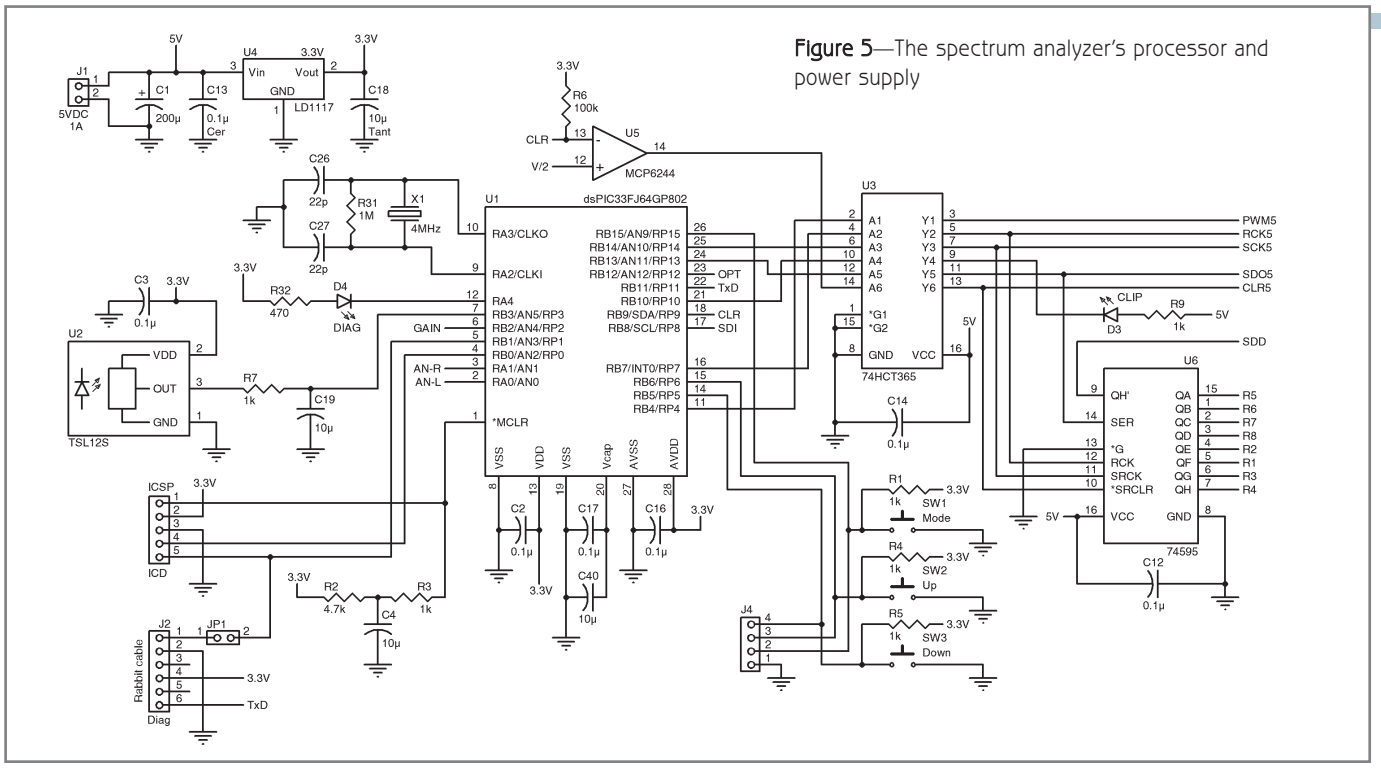


Figure 5—The spectrum analyzer's processor and power supply

more and it sure beats trying to track down and fix noise problems.

The photocell, attached to the back of the display PCB, is wrapped around the bottom of the PCB so it receives light from the front of the unit and it has some play for adjustment. It should be located such that it

Great New Products

New ET-Easy Stamp
- Includes ATMega168 with Installed Arduino Bootloader
- Direct USB Program Download
- Up to 22 I/O Points, 10-Bit ADC Included
- Compact, Easy to Use and Program

ONLY $24.90

ONLY $14.90

Ultrasonic Range Finder
- Ideal for use on Robots and Water Tanks
- Measures from 3cm to 3m
- High Accuracy
- Ready to Run, No Set-Up Required

10A Solar Charger for Lighting
- High Efficiency PWM Charging
- Charges Batteries During Daylight and Switches Lights on at Night
- Suitable for 12V and 24V Systems
- LED Status Indication for Charging, Low Battery etc

ONLY $36.90

www.futurlec.com

Open Frame Panel PC
PPC-E7+
- Fanless ARM9 400MHz CPU
- 4 Serial Ports & SPI
- Open Frame Design
- 10/100 BaseT Ethernet
- SSC-I2S Audio Interface
- 2 High Speed USB 2.0 Ports
- SD/MMC Flash Card Interface
- Battery Backed Real Time Clock
- Up to 1 GB Flash & 256 MB RAM
- Linux with Eclipse IDE or WinCE 6.0
- JTAG for Debugging with Real-Time Trace
- WVGA (800 x 480) Resolution with 2D Accelerated Video
- Wide Input Voltage from +12 Vdc to +26 Vdc

Windows CE
2.6 KERNEL

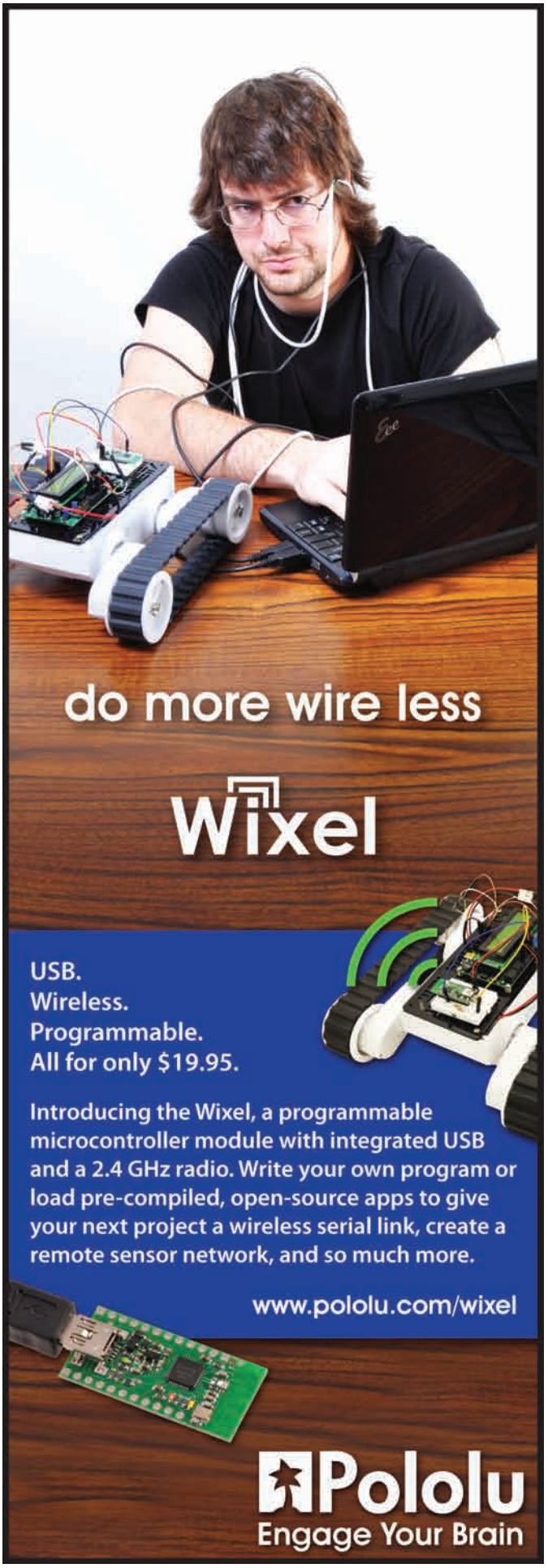
Setting up a Panel PC can be a puzzling experience. However, the PPC-E7+ Compact Panel PC comes ready to run with the Operating System installed on Flash Disk. Apply power and watch either the Linux X Windows or the Windows CE User Interface appear on the vivid color LCD. Interact with the PPC-E7+ using the responsive integrated touch-screen. Everything works out of the box, allowing you to concentrate on your application, rather than building and configuring device drivers. Just Write-It and Run-It.

For more info visit: www.emacinc.com/panel_pc/ppc_e7.htm

Since 1985 OVER 25 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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



receives as little light from the LED arrays as possible. If the automatic brightness doesn't seem to work well, try aiming the photocell up or down to get the desired effect.

I painted the edges of the LED displays with some flat black paint. (Why do they use white plastic for these anyway?) This improves the way it looks through the front panel and reduces the amount of reflected light on the photocell. I then put the entire assembly in a custom wood box with an acrylic front panel and a plastic back plate. The base PCB slides into a couple of wood cleats, the front and back panels slide into slots, and everying goes together sort of like a Chinese puzzle. The only fasteners are the four screws that attach the bottom plate of the box. The control push buttons, audio input connector, and power connector are all mounted on the back panel. A standard 5-V, 1-A power cube completes the unit.

OPERATION

Operating the spectrum analyzer couldn't be easier. There's nothing to calibrate. Just plug in the power and line audio connectors and press the mode and gain Up/Down buttons until you get a desirable display. The signal input is protected and has a high impedance, so it's hard to kill and it won't distort the signal it's monitoring.

What can I say? This is a lot more entertaining than a simple VU meter.

Richard Pierce (rsp@fearlessnight.com) is a medical product design engineer who studied Computer Engineering at the University of Illinois. He also designs home entertainment/automation gadgets.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pubCircuit_Cellar/2011/250.

RESOURCE

S. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing,* California Technical Publishing, 1999.

SOURCES

LED Arrays
Fearless Night | www.fearlessnight.com

dsPIC33FJ64GP802 Processor, C30 Compiler, and MCP4261 Digital potentiometer
Microchip Technology, Inc. | www.microchip.com

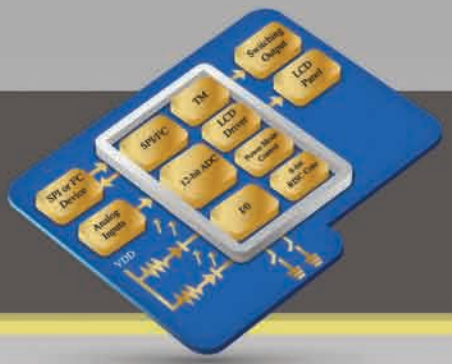
LED Arrays
SparkFun Electronics | www.sparkfun.com

Audio analyzers
Velleman, Inc. | www.vellemanusa.com

HOLTEK
www.holtek.com

All in one

Ultra Low Power Consumption - Highly Functionally Integrated Flash MCU Solutions



Holtek's new ultra low power consumption Flash MCU device series, the HT67Fxx series A/D with LCD type, and the HT69Fxx series LCD type, in utilising Holtek's unique TinyPowerTM technology, greatly reduce power requirements to meet the demands of today's environmentally conscious products. Additionally, Holtek's HT66Fxx A/D type series and its HT68Fxx I/O series further broaden the Flash range to provide an increased selection of flexible device solutions. The complete device range contains 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.

HT68Fxx	Internal Multi-function Timer Modules. IRC 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.
HT69Fxx	Uses Holtek's TinyPower ultra low power technology. Internal R-type/C-type LCD driver.
HT67Fxx	Includes all the HT69Fxx functions with an additional Internal High Speed 12-bit ADC with internal reference voltage source. Dual SPI or I²C interfaces. Internal Data EEPROM function etc.

STD Flash MCU HT66/67/68/69Fxx Series

- Wide Operating Voltage Range: 2.2V~5.5V
- Ultra Low Power
- 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 STD Flash 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 Keith Brown (Canada)

Control Shaft Encoders

Searching for an ideal control shaft encoder can be a time-consuming task. This article details some workarounds to help determine which encoder is appropriate for an application. An interesting debounce and decode circuit is also presented.

This article is about control encoders that are built from mechanical switches whose contacts alternate between conductive and non-conductive areas arranged in one or more circles. These devices are intended to be used as controls mounted on a panel with a knob on their shaft. They are also meant to be turned by human fingers. The devices are a subset of what are called "shaft" or "rotary" encoders, but these have generally fewer than 100 resolvable steps per rotation. Anything better requires an optical disk or another method with finer resolution (even some resistive potentiometers can be better than this under appropriate circumstances). Unless you have vintage wheels, your car radio almost certainly uses one of these types of low cost encoders for its volume control. The scroll wheel in your mouse is likely in this category as well—although the actual mouse position sensors usually use an optical disk in a mouse with a ball, or are purely optical in one without a ball.

My goal is to let you in on what I learned when I was doing my own search for an ideal control shaft encoder for a couple of my projects. I found that I was buying units that did not do the job. My frustrations included not knowing beforehand how the control would feel (Just what do the torque specs mean?), not knowing what resolution was needed for a finger-rotated control shaft, and trying to gauge just how much debouncing was required.

As part of this article, I'll also present a small debounce and decode circuit. I used a Microchip Technology PIC10F206 in an SOT-23 six-pin package (see Figure 1). All four I/O pins are used. Two inputs from the decoder and a clock pulse along with an up/down flag are used to drive an external counter. I wrote the code in Assembler and it served as a testbed. It can be adapted to any bigger PIC with little trouble since the PIC10 family's instructions are a subset of larger PICs.

Whether they are described as "Gray code," "Gray scale," "incremental encoders," "2-bit quadrature," or even "phase difference signal" types, they all share this basic principle: at a constant rotation speed, they output two nominally square waveforms with what is intended to be a 90° overlap. It seems universal that the names of these signals are A and B, leaving the inspiring label "C" for the common. The relative polarity of the two waveforms might differ, but that does not matter. Just swap A and B if the opposite phase relation is desired. Also, it is not always clear which level, high or low, corresponds to a switch being closed or open: Sometimes the datasheets don't show whether it's assumed that the common is wired to the system's positive voltage with pull-down resistors or wired to the ground with pull-ups. But again, it does not really matter since just swapping the signals will accomplish the direction reversal. So, either check out a real unit with a meter, or design your system to enable the easy swapping of the detected direction, either in hardware or software.

There are two "weasel words" in the last paragraph. Did you catch them? Yes, the waveforms are *nominally* square and the overlaps are *intended* to be 90°. Therein lies the real world. I'll get into the real-world aspects later, but first the theory.

CODE THEORY

I think "2-bit quadrature" or "phase difference signal" are the most descriptive terms. Calling them Gray code devices is not wrong, but not really useful either. The 2-bit Gray code is effectively a degenerative case of general Gray codes. It is not necessary to know anything about Gray

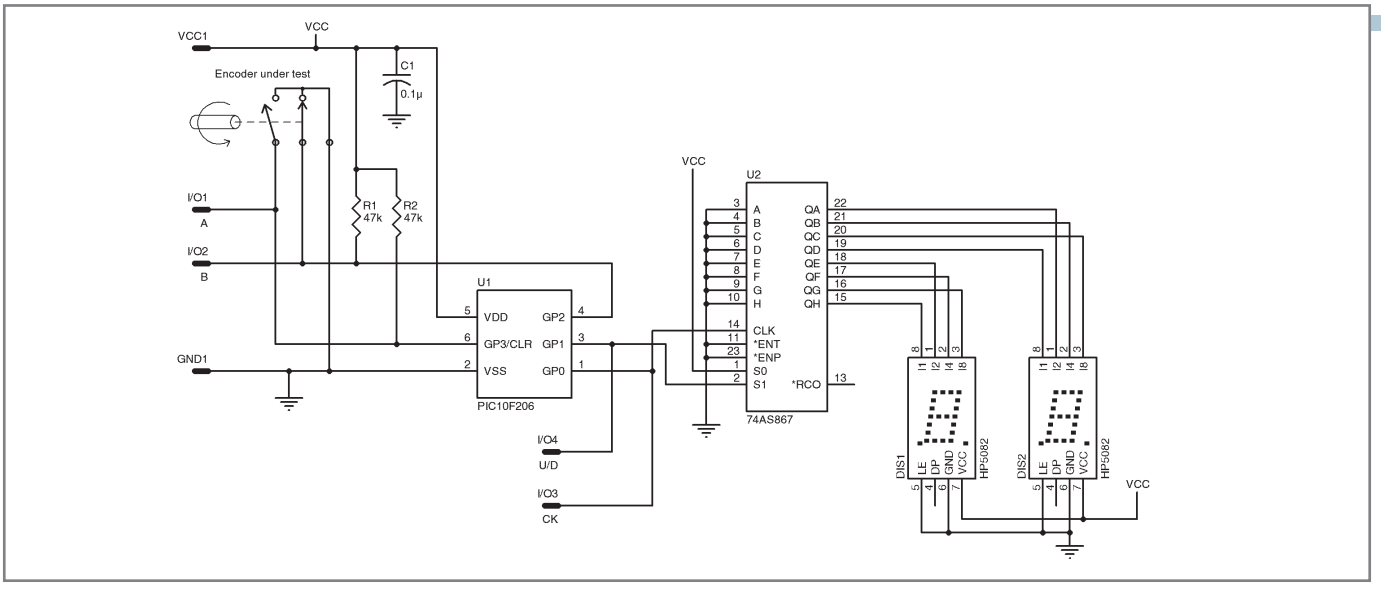


Figure 1—A simple testbed circuit. The firmware in the PIC10F206 needs to be changed depending on the type of encoder being tested. One thing to watch out for is that the processor cannot be programmed if either input A or B is held to the ground by an attached encoder. (The programming header is not shown.) If the encoder can be set to a position where both contacts are open, then that setup can be used. Otherwise, the encoder will have to be removed for programming.

codes here. All you need to know is that the overlapping codes that result from the waveforms in Figure 2 change only 1 bit at a time and that all four binary states of the 2 bits are represented. Moving from left to right in the figure, you see 00, 01, 11, and then 10. An important characteristic of this sequence is that if signal B leads signal A, the encoder is going in a certain direction. But if the lead/lag relationship is reversed, the encoder is rotating in the opposite direction. And, when you look at things in detail, for each and every case where a signal changes level, it can be determined which direction the shaft is rotating. This can be done by looking at 4 bits, two that are the state of the signals before the edge changed and two that are the state after the change. As an example, using Figure 1 again, if you have an upward edge on B when A is low (AB transition 00 to 01), the direction is the same (as if time flows from left to right in the figure). But if the upward edge on B happens when A is high, the opposite direction is indicated. I'll show the code I used to implement this later.

DETENT DETENTE

The biggest surprise I had when trying these units was how many different situations there are regarding the position of the detents with respect to the codes. Let's define the electrical resolution as the angle determined by 360 × 4 pulses per revolution (PPR), and the mechanical resolution is 360 divided by the number of detents. I had originally thought that if a unit had detents, the detents would occur at every code position so that the mechanical and electrical resolutions would be the same. It turned out that I found four different detent schemes with three electrical-to-mechanical resolution ratios (see Figure 3). There are four lines above the code waveforms and each indicates with dots where detents can occur. The first line from the top shows just one detent per cycle. I show it when both A and B are high (contacts open), since that is what I found in all the units that had this resolution ratio. However, there is also the possibility of three other cases. The second and third lines show two cases with two detents per cycle, one with detents at A==B and the other with detents at A!=B.

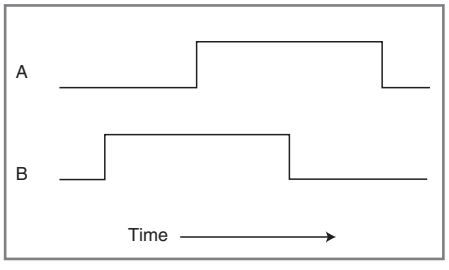


Figure 2—A typical timing relationship of the A and B phases. Unlike most such diagrams, time can flow either way.

Now we have another "real-world" aspect to consider—the positioning of the detents. At first, it didn't make sense that these were made with fewer detents than code positions. Why waste perfectly good codes? Besides, I found that it actually makes the decoding software more difficult. And, for a given mechanical resolution, the electrical resolution is two or four times finer, making the size of the contacts smaller by the same ratio. It turns out there is an advantage.

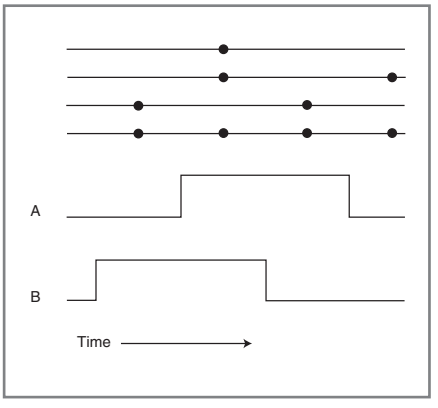


Figure 3—The detents can be built into the units with several relationships to the position codes. One, two, or four detents per cycle are all available.

Index	Manufacturer	Part number	Number of detents	PPR	Nominal size	Codes/detent	Notes
[1]	Grayhill	25LB10-Q	36	9	1 inch	1	Acoustically noisy!
[2]	Panasonic	EVE-GA1F1724B	24	24	12 mm	4	Warning: similar units may contain barlings!
[3]	CUI	ACZ16NBR1E-15KQA1-24C	24	24	16 mm	4	
[4]	Bourns	3315C-001-006L	0	6	9 mm	0	All plastic housing
[5]	CTS	288T232R161A1	0	4	16 mm	0	
[6]	CTS	288T232R161A2	16	4	16 mm	1	Same as above, but with detents
[7]	CTS	290VAA5F201A1	0	20	9 mm	0	PPR not verified
[8]	Mountain Switch	101-5437-EV	24	12	10 mm	2 (HH,LL)	Cheapest (less than $1)
[9]	ALPS	EC11B15202AA	30	15	11 mm	2 (HH,LL)	Appears similar to units 10 and 12
[10]	Piher	CI-11C0-V1Y22-HF4CF	30	15	11 mm	2 (HH,LL)	
[11]	Taiwan-Alpha	RE130F-40-20F-12P	12	12	12 mm	4	Appears similar to units 9 and 12
[12]	BI	EN11-VNB1AQ15	20	20	11 mm	4	Cheap, appears similar to units 9 and 10
[13]	BI	EN12-VN20AF20	0	24	12 mm	0	Cheap
[14]	BI	EN16-V22AF15	24	24	16 mm	4	Cheap, form, fit, and function same as unit 3

Table 1—The encoders' manufacturers, part numbers, and resolutions. The same numbers apply to Photo 1.

More on this later.

The torque needed to turn the shaft is another aspect of detents. Actually, the torque itself is not the entire story. The knob used to drive the device's shaft acts as a force multiplier. (Think of it as a differential force, with the thumb applying force in one direction and the forefinger in the other.)

According to Eastman Kodak's book, *Ergonomic Design for People at Work* (Volume 1, 1983), the maximum torque that can be applied to a knob is in the order of 0.43 N-m (0.5″ knob) to about 2 N-m for a 2″ diameter knob. After 2″, the maximum torque falls off. It turns out that these numbers are all much larger than the largest torque spec I found in encoder datasheets. But there is still a significant range to the finger feel in these devices.

THE LINEUP

After encountering problems with three encoders from past projects, I went on a buying spree (the units cost anywhere from $0.50 to around $6) and bought 11 more. The first seven units were bought from distributor "D" and the last seven were obtained from distributor "M." Photo 1 shows the devices and Table 1 lists the manufacturers, model numbers, resolutions (both PPR and number of detents), nominal sizes, and the code/detent relationships of all 14 units. In the rest of this article, I will refer to the units by their numbers, which you can see in the photo and the first column of the table.

All but two of the encoders have PCB through-hole terminals. The exceptions are units 5 and 6, which are in the same-lugged package. Of the PCB mount types, six are designed so that the encoder's shaft is parallel to the PCB and six orient the shaft at right angles to the PCB. One can also obtain SMD versions of some of the smaller types.

As you can see in Photo 1, the devices differ greatly in size. Unit 4 is the smallest and unit 1 is the largest. There is no direct correlation between size and resolution, or between size and number of detents.

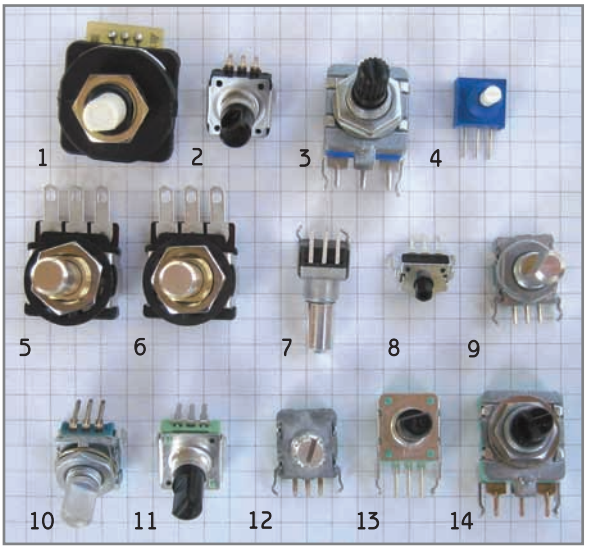


Photo 1—All 14 of the units examined. Many were taken apart. Some were used with the PIC10F206 testbed. Note that the background grid is nominally either 0.2″ or half a centimeter, whichever you prefer. The same grid is used in all of the other photos.

DATASHEET DESPAIR

The datasheet state of affairs is generally despicable. Several of these would rather say how high a voltage the contacts can handle (most designers won't care that the unit that is going to be used to switch a logic level can handle 120 VAC) rather than inform the designer how the code pattern relates to the detents. I also found nonsensical units, cases where specifications were quoted without a clear indication of test conditions, and specifications stated in terms of entities that were not defined elsewhere in the datasheet. A situation I found at least once was signal timing diagrams that showed the detent positions as being in line with a signal edge. This was either a serious design flaw or a serious misunderstanding of how the units were designed on the part of the datasheet authors.

To describe the electrical resolution, datasheets can use any of the following: pulses per revolution; PPR; Number of detent [sic]; and, in one case, pulses per revolution per track. Contact bounce is most often called just that, but it seems that chatter and sliding noise refer to the

LPC1200

Cortex-M0 – A simple choice



- Up to 128 KB Flash with 512 Byte page erase and 8 KB SRAM
- Industry's first ROM-based divide library for Cortex™-M0, offering fast and deterministic execution for 32-bit division
- Unique configurable peripherals suited to energy efficient applications
- Supported by LPCXpresso tool chain for easy development and migration

www.nxp.com/cortex-m0

same problem: when a contact is sliding along a conductive section and it momentarily opens. Sometimes there is a specification for mechanical vibration.

There was an unusual number of odd or just wrong entries in the datasheets. I think electronics engineers are used to a fairly consistent standard with our IC and discrete electronic component datasheets, but at this specific intersection of mechanical and electrical design, that consistency does not exist. I'll just list some of the most unusual things I found. In the datasheet for unit 1, it clearly states that the insulation resistance between terminals is 1,000 mΩ. I suspect that instead of 1 Ω, 1,000 MΩ was meant, but 1 GΩ seems unusually high. In the datasheet for unit 2, I was confused by a word I had not seen before: some of the encoders covered by the document were built with "barlings." The term was used consistently throughout, and it took Google "define: barling" (no definitions found) to convince me that this had to be a term for bearing! Another translation problem was found in unit 10's document where the term "phrase difference" is used instead of "phase difference."

The majority of the datasheets have contact bounce specifications in the range of 2 to 5 ms. Four of the datasheets do not have this very important specification (those that cover units 1, 3, 5, 6, and 8)! Often it is not clear at what shaft speed the debounce spec applies: a speed will be listed, but these are seemingly associated with either the lifetime specifications, or are a maximum speed, and not necessarily tied to the contact bounce spec. When one is using fingers to turn a shaft, a wide range of speeds can be obtained, and it would seem obvious that the slower you turn the shaft on this kind of switch, the longer the bounce time. So, if the bounce times are specified at either the maximum or lifetime test speeds, those specs will be next to useless! This might mean that to get absolutely reliable results from these units some sort of dynamic debounce logic or circuit would have to be used. The only datasheets that supply a clear test condition for both the bounce time and the sliding noise are the last three, those from BI Technologies. In all of these three datasheets, the sliding noise is specified at 60 RPM and the bounce time is at 15 RPM.

A wide range of units is used to specify torque. I found all of the following: oz-in, mN-m, N-cm, gfcm, and gf. (The last one has no length component and therefore is *not* a torque. This was on the datasheet for unit 8.)

One thing I never found any reference to in the datasheets is the acoustic noise level of the clicks when moving between detents. The first encoder I tried, unit 1 (Grayhill), was annoying in the application that I intended it for—lighting control in a quiet room—because it was just too loud!

DECODER DECONSTRUCTION

David Jones, host of the EEVblog, often says, "Don't turn it on, take it apart." I followed his advice and started doing just that.

The first unit did not need much deconstruction. Unit 1, the largest one, is made from a fairly thin PCB that's heat-staked to the plastic body. Photo 2 is a view through the board from the back of the unit after a small label was removed. You can clearly see the encoder's contact pattern and that there are nine PPR per track. The rotor must consist simply of three contacts connected together on the same radial line, with the contacts wiping near the centers of each track, connecting each outer one as a function of the angle to the center common one. It's definitely a straightforward implementation. I was concerned that these radial connections to the pins would cause extra pulses at one particular location, but after thinking about it some more, I realized there wasn't a problem. (I'll leave this as a proverbial "exercise for the student.") The contacts for units 5 and 6 can be seen through an opening, so I was able to conclude that they are of the same type of design.

Another observation: If you look at the angles covered by the contact area and the insulator between them, you will see that the contact area's angle is the smaller of the two. In the first photo, where the two tracks are at quite different radii, a further difference is seen: the insulator's angles seem to be about the same for both tracks, but the conductive areas are significantly different. Why? The sliding contacts themselves must necessarily cover an angle larger than zero, and this angular size needs to be subtracted from the conductive portions to approach the ideal performance.

I opened up unit 3. This proved to be quite a different beast (see Photo 3). It uses a very different contact/track arrangement. As you can see in Photo 3a, the rotor is made up of three contacts in the same circle and equally

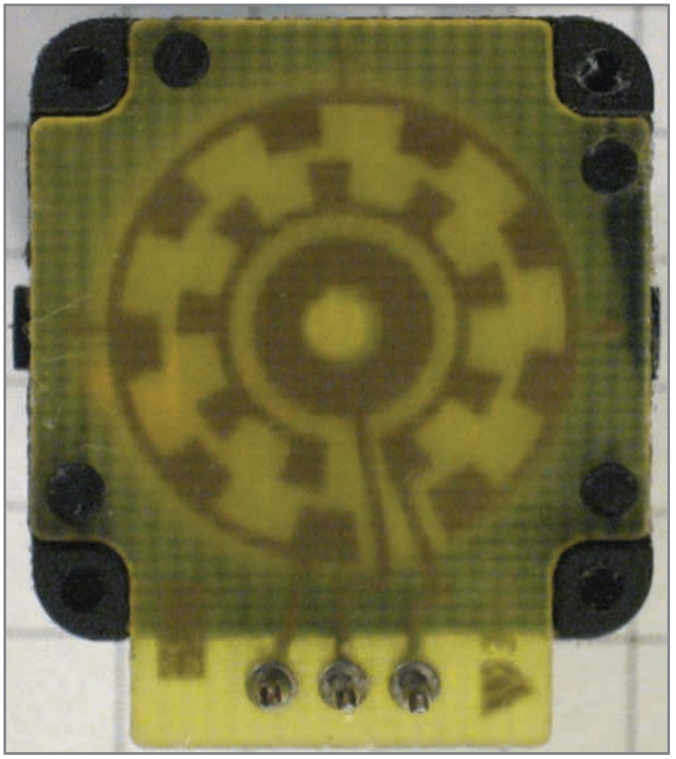


Photo 2—The back of encoder 1. Note the nine contact areas in each of the tracks and the common track on the inside.

DISTANCE LEARNING COURSE

Programming Embedded PIC Microcontrollers

using Assembly, C and Flowcode

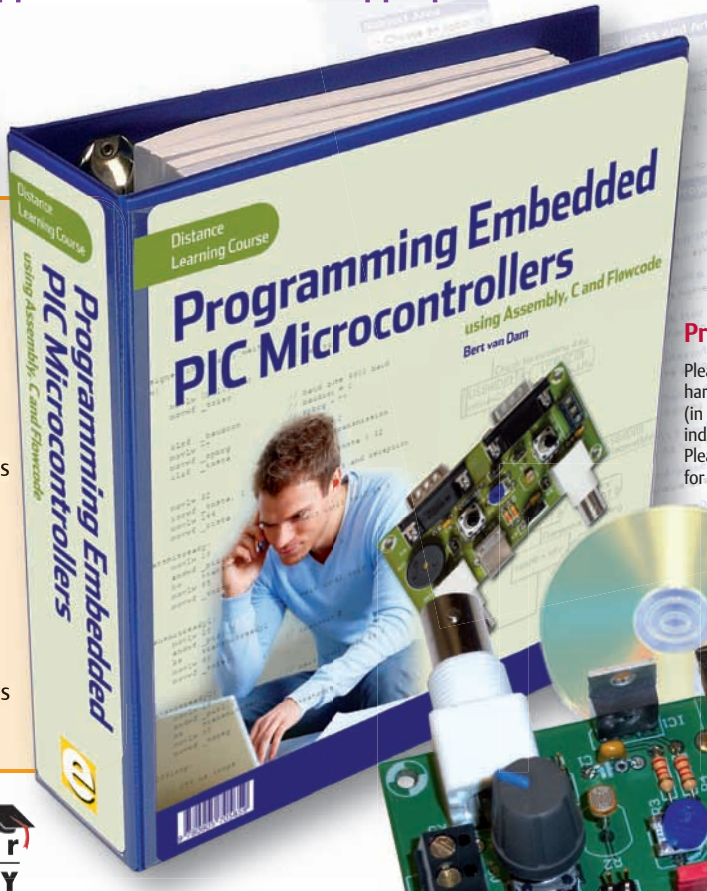
PRICE SLASHED!
$70 DISCOUNT

In this course you will learn how to program an embedded microcontroller. We will start with the absolute basics and we will go into a lot of detail. You cannot learn about software without understanding the hardware so we will also take a close look at the components and schematics. At the end of the course you will be able to design your own embedded applications and write the appropriate software for it.

including elektor electronics worldwide Certificate

Contents:

- Background
- Digital Ports
- Serial Communication (RS232)
- Analog Signals
- Pulse Width Modulation
- Timers/Counters/Interrupts
- Memory
- LCD Display
- I²C Communication
- SPI Communication
- USB Communication
- Configuration (Fuses)
- Answers to the assignments
- Appendix





Your course package:

- Courseware Ring Binder (747 pages)
- CD-ROM including software and example files
- Application Board
- Support at Elektor Forum
- Elektor Certificate

Price: ~~$645.00~~ $575.00

Please note: to be able to follow this course, E-blocks hardware is required which you may already have (in part). All relevant products are available individually but also as a set at a discounted price. Please check www.elektor.com/distancelearning for further information.

elektor ACADEMY
the school of electronics

Further information and ordering at
www.elektor.com/distancelearning

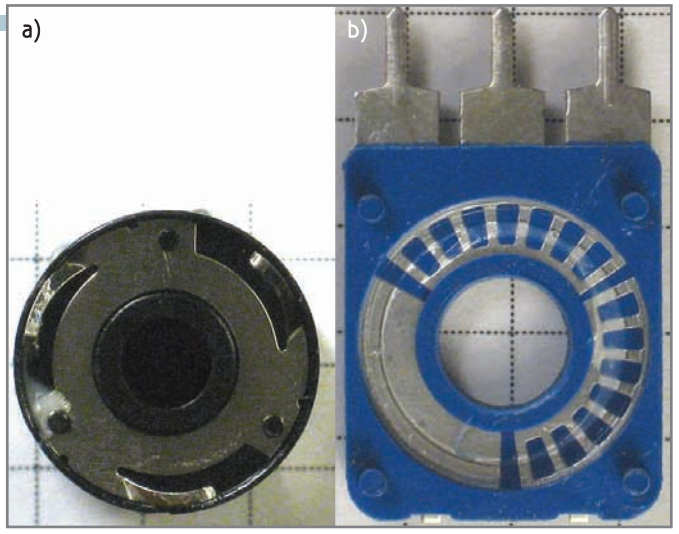


Photo 3—The inside of encoder 3. I thought I was looking at something unique when I opened this unit; however, this turned out to be the most common style of track/contact arrangement.

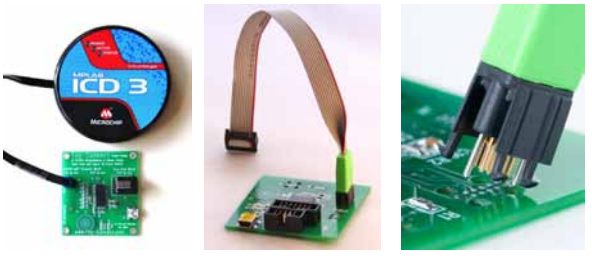
spaced. The stator, on the other hand, is shown to be made up of three parts, each occupying a third of the circle (see Photo 3b). It's a clever transformation since it requires the same number of sliding contacts (three) and the same number of continuous metal sections for those contacts to slide on (three again), but there is no squeezing of tracks into smaller radii, which is required with the straightforward scheme.

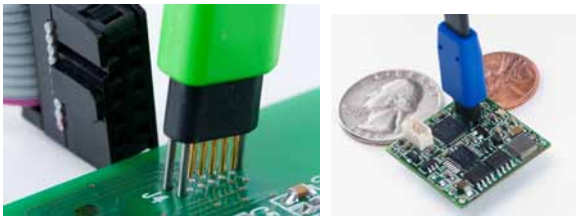
The contact arrangement found in unit 3 turned out to be the most common scheme when I opened more and more units. Units 2, 8, 11, and 13 were all found to use the same scheme. It's likely units 9, 10, 12, and 14 also use the same scheme. That only leaves unit 4, which is totally sealed, and unit 7 in the undecided column. You can certainly understand the advantage of using one radius instead of three. Some of these units are so small that it is hard to imagine that two tracks (three counting the common) could fit in the radius available.

There is another type of design that appears in various sources I consulted. It is a scheme where one track is used but two sliding contacts are spaced with an effective offset of a quarter cycle. Of all 14 units, I found only one that uses this scheme: unit 7 (see Photo 4). It has contacts on opposite sides of the device's body but at the same radius. The contacts were in wells filled with silicone grease, so it was hard to immediately see that this was the scheme in use. Looking at the dimensions, the contacts are about 5.8 mm apart on a line through the center, so the required circumferential offset would be only a 0.23 mm ($5.8\pi/80$ mm since this is a 20 PPR unit) difference in the lengths of the two contact arms, which seems rather hard to control. What seems to be the case is that the contact arms are the same length but both land on the code disk just a bit above the center line. I added a line through the center of the unit to show this (see Photo 4). Ideally, the offset

Easy Embedded Linux
OMNI-EP


• 200 Mhz ARM9 CPU
• 10/100 Mb Ethernet
• 32 MB RAM
• 16 MB Flash
• 16 Digital I/O Lines
• 2 Ports of USB 2.0
• SPI Bus
• AC97 Amplified Audio
• Battery Backed Clock
• 2 Serial Ports
• Low Power Consumption
• RoHS Compliant
Our newest ARM9 Linux controller, the OmniEP doesn't cost an arm and a leg. It delivers removable storage, amplified audio, ethernet and serial RS232 communication ports in a rugged and attractive enclosure. Models without enclosure and LCD available.
The OmniEP comes preloaded with Linux to jumpstart your development process, with LCD and pushbutton drivers supplied. Large capacity USB drives can be easily mounted in the USB port.
Call 530-297-6073 Email sales@jkmicro.com
www.jkmicro.com
JK microsystems, Inc.
International Orders Welcome
MADE IN U.S.A

Tag-Connect


JTAG Connector Plugs Directly into PCB!!
No Header! No Brainer!

Our patented range of Plug-of-Nails™ spring-pin cables plug directly into a tiny footprint of pads and locating holes in your PCB, eliminating the need for a mating header. Save Cost & Space on Every PCB!!
Solutions for: PIC . dsPIC . ARM . MSP430 . Atmel . Generic JTAG . Altera Xilinx . BDM . C2000 . SPY-BI-WIRE . SPI / IIC . Altium Mini-HDMI . & More
www.PlugOfNails.com
Tag-Connector footprints as small as 0.02 sq. inch (0.13 sq cm)

would be half of the calculated value, so something like 0.115 mm!

CONTACT BOUNCE

Now, let me justify the need for contact debounce. In an ideal situation, it would not be needed since, in theory, only one contact can change state at a time. But again, the real world intrudes. There are two situations that can cause both contacts to be changing at or near the same shaft angle. The first is when the bounce on one contact has not finished by the time the other contact has reached a transition. The second is when sliding noise (when a contact that is supposed to be closed opens briefly) occurs near either of the other contact's transitions. Both of these conditions will be exacerbated by aging and will depend on shaft speed. In an optical encoder, there should be no need for debouncing, but that is not the case with these sliding contact types.

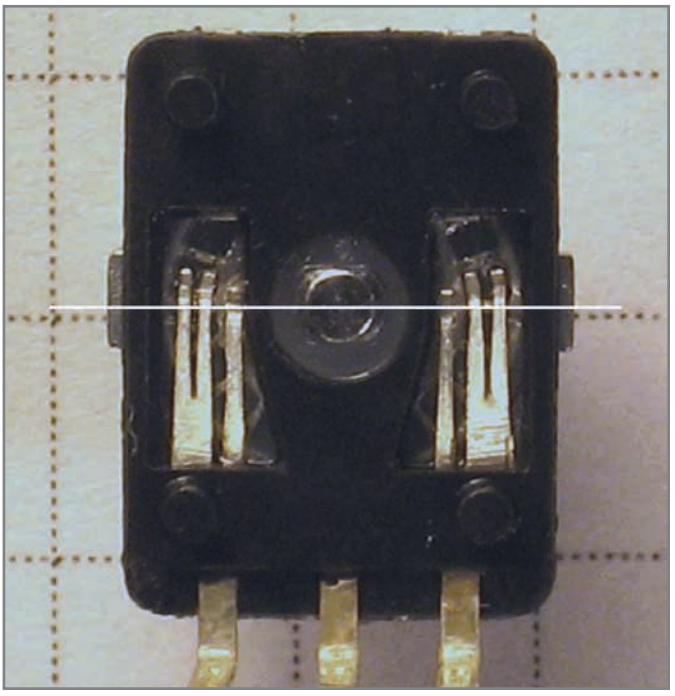


Photo 4—Inside encoder 7. This was unique in the set as it turned out to be the only one with one physical track and an offset of a quarter cycle between the A and B contacts.

Useful information about contact bounce is available at the Ganssle Group's website (www.ganssle.com/debouncing.htm). Bear in mind that most of the 16 switches dealt with there would have had some kind of spring-driven snap action that cannot be implemented in a position encoder. This can be seen by the fact that switch F, a slide switch, was found to be "very sensitive to the rate of activation." That is what these encoders are: circular track slide switches. Both the R-C filtering approach and the use of the MC14490 hex debouncer that Jack Ganssle covers are also suggested in several of the encoder datasheets.

My approach to debouncing was to take samples in a tight loop as quickly as possible but to require that N samples be the same before a change is declared. By changing the value of N, longer or shorter debounce times can be implemented (see Listing 1). The maximum number of instructions in the loop is 10 when the value read from the contacts matches the stored value, so it runs in 10 ms in the processor I used. A value for N (the constant `NEEDED` in the code) such as 32, which gives a debounce period of 320 ms, seemed to work fine. But wait a minute! Isn't that about 10 times faster than the specs say? Welcome back to the real world. I had no means of measuring the shaft speed, but it was likely that since no knob was being used, the test speeds were on the low side. Like most design decisions, it is all about tradeoffs. Here, the tradeoffs are whether to ignore transitions that might happen at higher shaft speeds, but to be able to sense every single detent at low speeds, or whether to allow for the degrading effects of age, and it is a given that



Listing 1—The simple debouncing routine. The constant NEEDED determines how many times through the loop the processor will run when the values do not change so that a new stable condition can be declared.

```
loop_start
       movf     GPIO,w              ;read in the port
       andlw    0x0C                ;just want two bits from it!
       movwf    entry               ;store these two bits into a variable
       subwf    old_sample,w        ;is the new value the same as the previous value?
       btfsc    STATUS,Z            ;if subwf result was zero then old and new are (still) the same
       goto     are_same
       ;since samples are different, need to re-start the debounce counter:
       movlw    NEEDED              ;initialize counter,
       movwf    needed_count        ;needed_count <- NEEDED
       ;now put the new reading into old_reading
       movf     entry,w             ;get back newest reading
       movwf    old_sample          ;now the new is the old
       goto     loop_start
are_same
       ;the old and new match so decrement the counter if not at zero already:
       movf     needed_count,f      ;syntax is basically a test for zero
       btfsc    STATUS,Z            ;if needed_count is zero, go to at_end
       goto     at_end
       ;otherwise just decrement the counter:
       decf     needed_count,f
       goto     loop_start
at_end                              ;at end of search for NEEDED values the same
```

the contacts will get worse with time.

LOOK UP, LOOK WAY UP

It only takes a simple 16-entry look-up table to affect the decoding of the combination of the two previous bits and the two new bits into a number, which tells the processor what to do. Also, half of the entries are zero because there are four combinations that are deemed incorrect (where both bits have changed) and four other combinations that represent no change at all. The pseudocode in Listing 2 is a distillation of the Assembler code on the *Circuit Cellar* FTP site. All variable names are the same as in the Assembler code. The listing shows how the two least-significant bits (LSBs) of the look-up result are used to decide whether to increment or decrement the software counter `internal_count` or to do nothing. If the counter is modified, the next 2 bits are used to decide which encoder output had an edge change so that the two flags `A_edge` and `B_edge` can be set accordingly. If an edge happens on both A and B and the counter is such that its two LSBs are 0, then an output pulse is issued. The last sentence reflects the key difference between a one-code per-detent and a four-code-per-detent encoder: if either of the tracks is seen to toggle without activity in the other, no change is reported. This is the advantage I alluded to earlier: fault tolerance. In a one-code-per-detent encoder, the external counter would be told to increment and decrement repeatedly under that condition. In a two-code-per-detent encoder, it turns out that at least one change on both phases is also required. A two-code-per-detent case would seem to be the ideal, but as Table 1 shows, these don't seem to be as popular as the four-code-per-detent case.

To change the code to handle a two-code-per-detent encoder, only one change would be needed: require the LSB of the counter to be 0 to produce an output pulse instead of requiring the "10" condition. To use with a one-code-per-detent encoder, neither the internal counter

Listing 2—Some pseudocode to show the logic of the look-up table. The variable names are taken from the actual code so that you can read the Assembler code easily.

```
combo = (entry & 0x06) | ((combo >>2) & 0x03;
//i.e., combo is 2 new bits from encoder + the 2 old bits
CK_OUT = Low; //clear a possibly set clock output. Clear it here for a longer pulse width.
looked_up = table_start[combo];
if (looked_up !=0){
  if ((looked_up & 0x03) == 1) internal_count--;
  if ((looked_up & 0x03) == 2) internal_count++;
  if ((looked_up & 0x04) ) flag(B_edge) = true;
  else flag(A_edge) = true;
  if (flag(A_edge) & flag(B_edge) & ((internal_count&0x03)==0)){
    CK_OUT = high;
    flag(B_edge) = flag(A_edge) = false;
  }
}
```

nor the two edge flags are needed.

ENCODERS IN HAND

There are many important aspects to consider when using one of these encoders in a design. The datasheets are generally so poor that I can only conclude that in all cases you should get a sample or two before designing one of these in. For one-off, quick-and-dirty applications, I suggest having an assortment on hand. There are so many variations in mounting style, resolution, and size, I can't really suggest a list. But I hope Photo 1 and Table 1 will help. The other conclusion I offer is that device manufacturers generally need to improve how they prepare their datasheets.

Keith Brown, P.Eng. lives on Vancouver Island, BC. Operating as PLD Designs, which consults in the selection of PLDs and designs circuit boards for customers (with and without PLDs), he has published two earlier articles in Circuit Cellar. *Go to www.island.net/~kdbrown for details. You may contact Keith by e-mail at kdbrown@island.net.*

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/250.

RESOURCES

EEVBlog, www.eevblog.com

Eastman Kodak Company, *Ergonomic Design for People at Work, Volume 1,* Wiley, John & Sons, Inc., 1983.

The Ganssle Group, "A Guide to Debouncing," 2004, www.ganssle.com/debouncing.htm

SOURCES

EC11B15202AA Encoder
ALPS Electronic Co. | www.alps.com

EN11-VNB1AQ15, EN12-VN20AF20, and EN16-V22AF15 Encoders
BI Technologies | www.bitechnologies.com

3315C-001-006L Encoder
Bourns, Inc. | www.bourns.com

288T232R161A1, 288T232R161A2, and 290VAA5F201A1 Encoders
CTS Corp. | www.ctscorp.com

ACZ16NBR1E-15KQA1-24C Encoder
CUI, Inc. | www.cui.com

25LB10-Q Encoder
Grayhill, Inc. | www.grayhill.com

PIC10F206 Microcontroller
Microchip Technology, Inc. | www.microchip.com

MC14490 Hex contact bounce eliminator
ON Semiconductor | www.onsemi.com

EVE-GA1F1724B Encoder
Panasonic Corp. | www.panasonic.com

CI-11C0-V1Y22-HF4CF Encoder
Piher International Corp. | www.piher-nacesa.com

RE130F-40-20F-12P Encoder
Taiwan Alpha Electronic Co. | www.taiwanalpha.com



by Mariano Lizarraga, Renwick Curry, and Gabriel Elkaim (USA)

Reprogrammable UAV Autopilot System (Part 2)

Testing and Results

In the first part of this series, you were introduced to the SLUGS reprogrammable UAV autopilot system. This article details the system's software and flight test results.

In the first part of this two-part series, we introduced the hardware used in the Santa Cruz Low-cost UAV GNC System (SLUGS), an open-source unmanned aerial vehicle (UAV) autopilot primarily focused on supporting guidance, navigation, and control (GNC) research. In this article, we'll focus on the software architecture and algorithms. We'll also present results from temperature compensation, sensor calibration, and flight tests.

SOFTWARE ARCHITECTURE

The SLUGS has two Microchip Technology dsPIC33 digital signal controllers (DSCs) that serve as its main processing units, running at 40 MIPS. These are interconnected via a SPI at 10 MHz and are called the "sensor" DSC and "control" DSC, respectively. The sensor DSC is in charge of reading the sensor data and converting it into a high-quality estimate of the aircraft position and attitude (3-D orientation). The control DSC interacts with the user through the ground station, generating signals to the UAV's control surfaces to stabilize the aircraft and fly it along the user-designated mission (at its highest level, traversing waypoints at a given altitude and airspeed).

One of the key advancements in the SLUGS autopilot is that the DSCs are programmed directly from MATLAB Simulink using a block and signal flow metaphor for visual programming. This offers several advantages. Most researchers in the GNC field are already familiar with MATLAB and use it to prototype and simulate new algorithms. Complicated algorithms and changes are easy to implement, rapidly increasing the prototype-simulate-test cycle. Lastly, the software can be easily retargeted for better processors without having to rework the low-level, back-end code. While this environment sacrifices some efficiency, any performance issues are made insignificant by the speed with which changes can be made and tested.

The sensor DSC's low-level software architecture (see Figure 1) is a combination of blocks provided by Lubin Kerhuel's dsPIC Simulink Embedded Target (ET) (shown in

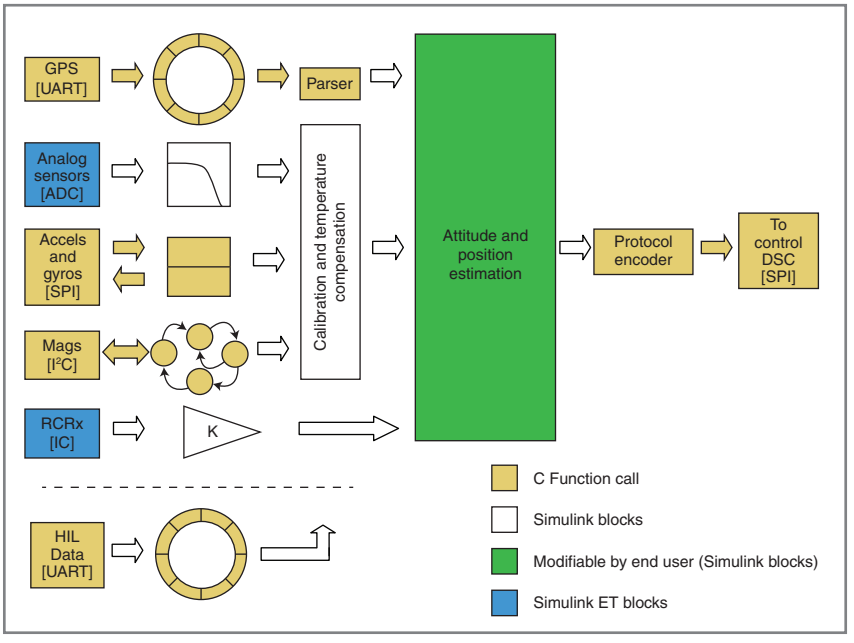


Figure 1—Low-level software architecture for the sensor DSC

blue), blocks implemented in C (shown in yellow), and basic Simulink blocks (shown in white).

Although the ET offered ready-to-use blocks for most of the DSC's peripherals, it was decided early in the design process that many of these would be rewritten in C to provide greater flexibility and full control of the implementation. For instance, to use less processor time while sending data through the serial port, it was necessary to use the direct memory access (DMA) feature of the DSC. The ET's UART block did not offer this capability, and it was necessary to rewrite it in C and provide it as a C function call block in Simulink. This is one of the key advantages of the ET: C functions can be merged into the overall generated code with great ease.

The GPS serial stream is read via a UART port and placed into a circular queue data structure. An NMEA 0183 protocol parser reads data from the circular queue and decodes the protocol sentences producing position, velocities, and course over ground (COG) information. The analog sensors (dynamic and static pressures, battery voltage, and temperature) are read directly from the hardware via the DSC's ADC and passed through a digital low-pass filter bank to attenuate sensor noise. The accelerometers and rate gyros are queried via a SPI, and the digital stream is decoded according to the device specifications. The magnetometers are configured and queried through I²C, implemented as an interrupt-driven state machine to decode the data.

The analog sensor, accelerometer, rate gyro, and magnetometer data are passed through a calibration and temperature compensation block to produce temperature-compensated data in meaningful engineering units. The algorithms and test procedures are detailed later. The data being received by the input capture port from the radio control receiver (RCRx) are received and scaled from the PWM duty cycle to angular values in radians corresponding to control surface deflection commands. The data are assembled into a data structure that is passed to other parts of the code for further calculations.

In hardware-in-the-loop (HIL) simulation (see Part 1) and in real flight, the attitude and position estimation algorithm (shown in green) processes the data and sends the results to the control DSC via a communications protocol encoder (which separates the data into sentences and manages the SPI inter-processor communications bus).

The control DSC's low-level software architecture (shown in Figure 2) is currently less resource intensive than that of the sensor DSC. The packets sent from the sensor DSC are received and placed in a circular queue. The inter-processor communications (IPC) protocol decoder reads from the queue and assembles the data for the navigation and control algorithm (shown in green).

In a similar manner, data received from the radio modem—containing ground control station (GCS) commands—are stored in a separate circular queue. The GCS commands protocol decoder reads the queue, interprets those commands, and passes them on to the navigation and control algorithm. The data being received by the RCRx are scaled from the PWM duty cycle to angular values and passed to the guidance and control algorithm to be used as initial conditions for the manual-to-autonomous transition. This ensures that no sudden transients are experienced by the aircraft when switching between manual and autonomous mode. In other words, an instant after telling the autopilot to fly the airplane, the control surfaces should be in exactly the same place that they were before the switch.

The guidance and control algorithm generates the required control surface commands, which are scaled from angular values to the correct PWM duty cycle and sent to each of the control surface's servos. These commands are also passed on (along with the data sent from the sensor DSC) to a protocol encoder, which splits them into different sentences, schedules them for transmission, and sends them to the radio modem for ground telemetry data. These data are used to drive the GCS display, recorded for post-flight analysis, and used for both control loop tuning and fault diagnosis in real time.

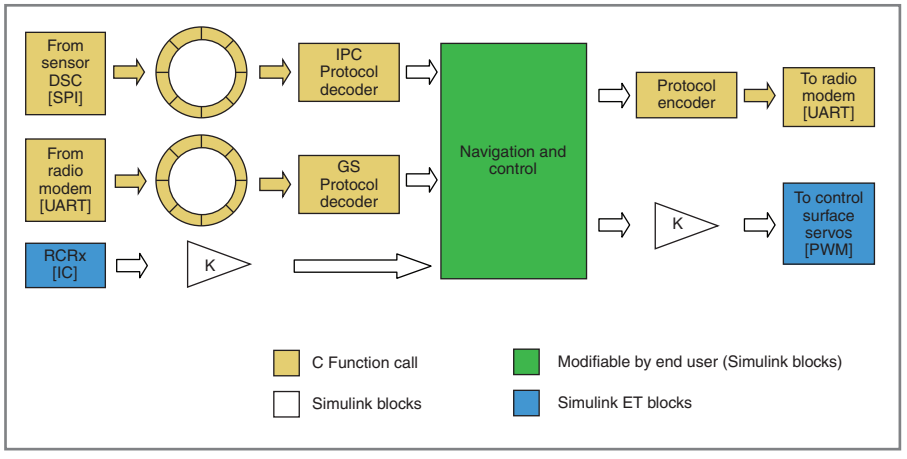


Figure 2—Low-level software architecture for the control DSC

COMMUNICATIONS PROTOCOL

The complete autopilot system generates 168 different meaningful variables. Attitude, position, sensor biases, temperature, absolute and differential pressure, control surface commands, controller gains, and many more are generated on every cycle. These values are not only used by the autopilot, but also sent to the GCS and logged to facilitate debugging and post-flight analysis. During our initial development phase, we produced a very simple, yet effective, binary communications protocol, where each message contained a 4-byte header and a 3-byte trailer. Nevertheless, we soon realized that adding a new sentence to the protocol was tedious and error-prone. After a detailed evaluation of possible solutions, we decided to adopt the

open-source communications protocol MAVLink developed by Lorenz Meier at ETH Zurich. This protocol is a lightweight, message-marshalling library for autonomous vehicles that allows the serialization of data structures (C-structs in particular) for transmission. But the feature that tipped the scales toward selecting MAVLink as the communications protocol of choice was the capability to automatically generate all of the required C code to serialize, pack, send, receive, and decode these data structures. One only needs to describe the fields of the packets at a high level using a simple XML syntax, and the tools available with the protocol will generate all of the required C code. Furthermore, the project itself has great documentation on how to integrate MAVLink into an existing project (provided the project is coded in C).

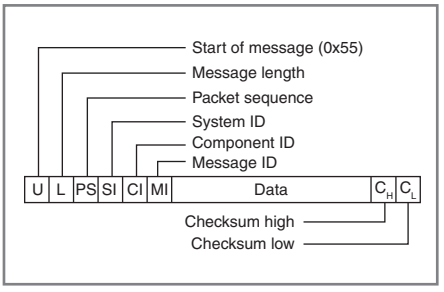


Figure 3—Communications protocol message structure

Figure 3 shows how the protocol's packets are structured. There is a 1-byte packet start sign (the letter U). Payload data specifies the size of the actual data being transmitted. A packet sequence is a counter that gets incremented as packets are sent out. The system and component identifiers uniquely identify the source of the message. A message identifier indicates what type of message it is. After the payload data, a 2-byte checksum is appended to the message to verify its validity upon reception.

Due to limitations in the RF link bandwidth, some of these messages are sent at a high data rate (100 Hz), while others that do not change very often are sent at a lower data rate (10 Hz). The protocol implements a small set of "spare" messages for debugging that can be assigned to any variable or calculation accessible within the autopilot software, both for floating-point and integer values. These are automatically logged and displayed on the GCS, and have proven very useful when testing out new algorithms.

TEMPERATURE & SENSOR COMPENSATION

The sensors used in the SLUGS are relatively inexpensive MEMS devices. As a result, they tend to have larger noise, nonlinearities, and temperature drifts than traditional (and more expensive) inertial sensors. The largest error seen in these low-cost sensors is a null-point shift as a function of temperature. Temperature compensation is used on the SLUGS to attenuate this null shift via an onboard, low-power linear Microchip Technology MCP9701 active thermistor. This is inexpensive and requires very little board real estate.

To determine which sensors within the suite required temperature compensation, we conducted the following experiment. The board was sealed inside a Ziploc bag and placed inside the lab's freezer. After an hour, the board was taken out, powered, started data recording, and placed on top of an electric grill to generate a slow temperature sweep (see Photo 1).

In order to find the temperature dependence for a given sensor, using the captured data, we performed a scatter plot of temperature versus sensor readings. From that plot, a simple linear coefficient (delta sensor counts/delta temperature counts) was extracted using a least squares fit. The sensor reading is then adjusted using this coefficient. The new sensor reading is the raw plus the coefficient multiplied by the difference in temperature from a nominal point.

The advantage is that this dramatically reduces the temperature drift of the sensors; the disadvantage is that the sensor reading is now a function of both the sensor and the temperature sensor, possibly increasing the noise in the calibrated sensor output.

Lastly, a more careful temperature curve could be used instead of the simple linear correction, although this was deemed effective for our needs. Figure 4 shows the importance of temperature compensation. It shows the temperature run (top left) and compares it with the readings from the static pressure sensor (top center). A clear correlation exists between the two. As expected, the scatter plot (top right) confirms the dependence between the two and shows the linear temperature coefficient. The resulting reading after temperature compensation (bottom plot, red) shows a significant improvement over the uncompensated reading (bottom plot, blue). Although temperature compensation does not completely remove the sensor's temperature dependence, it corrects the majority of the effect.

Bias points on the sensors are relatively easy to calibrate using static data (and several are estimated on-the-fly by the attitude and position estimation algorithms). In order



Photo 1—The SLUGS autopilot on top of an electric grill used to record data for temperature compensation calculations

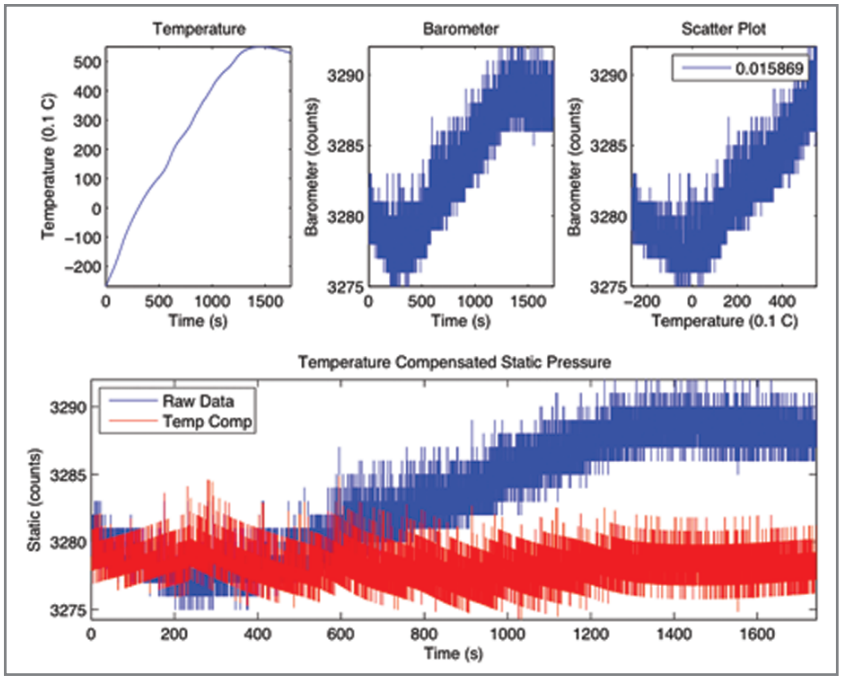


Figure 4—Static pressure sensor (barometer) temperature compensation results

to match scale factors, two different methods were used depending on the sensors. For the gyros, they were matched against a better sensor, a KVH DSP-3000 fiber optic gyro (FOG). A one-axis rate table and a least squares process were used to find the best scale factor match between the MEMS gyros and the FOG.

For the accelerometers and magnetometers, an ellipse method was used to simultaneously find scale factor, null shift, and non-orthogonality errors. The three axis sensors measure the x, y, and z components of a single (nonzero) vector. If the sensors were ideal, and you rotated the body around all axes and plotted the x, y, and z measurements, you would see points on the surface of a sphere. When plotting the real measurement instead of a sphere, you get a rotated ellipse. The center of the ellipse corresponds to the null shift, the semi-major axes to the scale factors, and the rotation to non-orthogonality within the axes. In the case of the magnetometer, these are often referred to as hard- and soft-iron effects. In practice, the aircraft was tumbled out at the field while telemetry data was stored on the ground station in order to get as close to the flying configuration as possible. (The specific algorithms that back out the parameters to correct the accelerometers and magnetometers can be found in the References section of this article.) Photo 2 shows the aircraft being tumbled by one of us prior to a test flight. (This is necessary to do only once, not before every flight.)

Lastly, the UAV was carefully leveled into flying attitude in the lab using a high-accuracy spirit level. The autopilot was turned on and allowed to converge on its attitude. This was used to generate a small rotation offset matrix to correct for the difference between the autopilot mounting and the aircraft body coordinates. This is held as a direction cosine matrix (DCM) in order to correctly account for the rotation. (You cannot simply add angles, and this does not preserve the mathematical relationships for rotation sequences.)

POSITION & ATTITUDE ESTIMATION

The initial design of the attitude filter used a complementary filter with a magnetometer triad for a heading reference and estimates of inertial velocity from the position filter. This design was placed in a Simulink simulation of 6DOF UAV flying a closed course defined by waypoints requiring changes in altitude and track. The original design did not include longitudinal acceleration (dV/dt in body coordinates), but this was found to be necessary to account for changes in speed. Several candidates for computing this term were compared on the basis of tracking errors before choosing a high-pass filter to compute the approximate numerical derivative.

Two attitude filters were designed. One used magnetometers for a heading reference. The other used GPS COG. The COG design was created because of the possibility of the high noise magnetometer readings (which were, in fact, experienced in the field) due to the UAV's electric motor.

The initial design for the position filter was a nine-state extended Kalman filter (EKF) with state variables consisting



Photo 2—Aircraft tumbling while recording data for sensor calibration

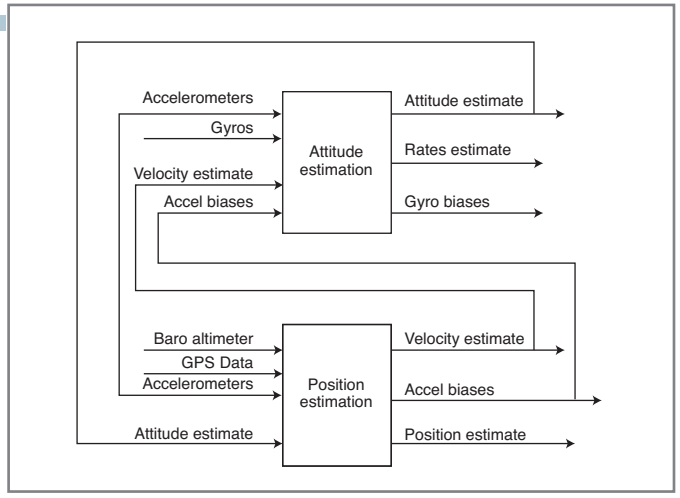


Figure 5—Position and attitude filter architecture

of position and velocity in North-East-Down (NED) coordinates and accelerometer biases. The input measurements were horizontal GPS position and velocity, barometric altitude, accelerometer readings, and the direction cosine matrix from the attitude filter. The filter performed strapdown navigation calculations for the state and covariance matrix at 100 Hz between GPS updates by integrating accelerometer readings after they were transformed to NED coordinates. This filter performed well in the six DOF simulations with noise characteristics taken from the sensor specifications. It was downloaded to the autopilot for further testing.

The objective of the first few flights was to log sensor readings while the UAV was manually controlled to validate attitude and position estimates before using these outputs for closed-loop guidance and control. Magnetometer readings were taken in the field during static ground testing before the first flight, and they were deemed unreliable. Thus, the attitude filter with COG heading reference was downloaded to the autopilot in the field and used for the remainder of the flights.

Sensor readings were logged during manually controlled flight. These data were then replayed after the flight through the Simulink position and attitude models. Unfortunately, the accelerometers' in-flight readings contained a high level of noise and the output of the EKF was unusable. These telemetry data also showed that the GPS readings had little noise (except for altitude drift and very large jumps in altitude readings) and GPS updates were averaging 3 Hz.

Due to the relatively clean GPS data and the fast GPS update rate, we replaced the EKF with complementary filters for horizontal position and velocity. For the vertical channel, barometric altitude was blended with GPS altitude readings to eliminate drift. The outlier changes in altitude readings were trapped with a threshold test before further processing.

The new filter design reduced the computational load on the sensor DSC from near 100% to 40%. Using the SLUGS environment, the elapsed time—from beginning the redesign to the first download to the autopilot—was less than 24 hours, demonstrating the flexibility and usability of the SLUGS for research purposes.

The final design performed extremely well, as shown by the excellent performance of the UAV under autonomous control. In addition, the attitude and position filters did extremely well when compared with more sophisticated models using other flight-test data (see Figure 5).

NAVIGATION & CONTROL

The control system designed for the SLUGS is based on the successive loop closure (or inner/outer loop) approach for aircraft control. It is completely implemented in the control DSC (see Part 1 of this series). The inner loop comprises a set of proportional integral derivative (PID) controllers separated into lateral (pitch and airspeed) and longitudinal (roll and yaw) channels. These are designed to stabilize the UAV and receive mid-level commands of turn rate, altitude, and airspeed. This loop generates the direct commands to the control surfaces: ailerons, rudder, elevator, and throttle. These each contain saturation limits on the servos, and can be individually tuned in flight. The outer loop implements a high-level, waypoint-based navigation, which in turn generates commanded turn rate and altitude for the inner loop. Airspeed is set to a constant and held by the inner loop PID controller. Altitude is controlled with two cascaded PID loops, one to command pitch based on altitude error with a strong saturation limit (set from the GCS to 15°), and a second to command the elevator to achieve the desired pitch output from the first PID stage.

The autopilot itself has three different submodes of operation. The first mode is Pilot Feedthrough, where the pilot commands are received by the RCRx and passed directly through the autopilot untouched.

The second mode is Manual Direct, where the GCS sends mid-level commands directly to the autopilot. Manual direct can be combined with Pilot Feedthrough such that

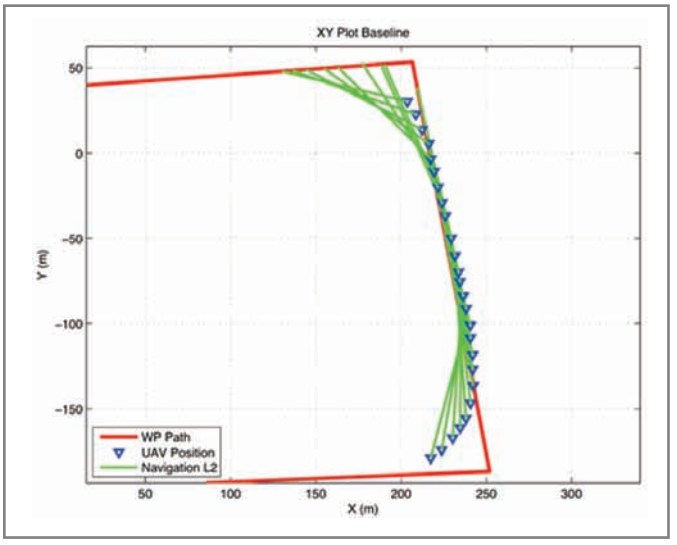


Figure 6—Time progression of the UAV position and the computation of the L2 vector during a flight test

the safety pilot controls some of the control surfaces, but the autopilot others. This is extremely useful when tuning control gains. For instance, throttle is given to the autopilot, but all other controls remain with the safety pilot. The airspeed control loop is tuned for decent behavior while the safety pilot can keep the UAV within the bounds of the test area. Furthermore, the safety pilot can induce errors by climbing or diving the UAV and seeing if the throttle control responds correctly.

The third submode is Autonomous-waypoint Navigation, in which the GCS operator configures the waypoints, and the navigation algorithm becomes responsible for generating the mid-level commands. All modes receive configuration settings from the GCS operator. The inner loop gets the PID gains for each controller. The outer loop receives the waypoints (in latitude, longitude, and height) and additional configurations that are necessary for its operation.

In our project, the UAV is tracking a point that is on the path from the previous waypoint to the next one that is ahead of the UAV. The "look ahead" length, called "L2," is scaled by ground velocity, making the aircraft less sensitive in a tailwind and more sensitive to cross-track errors in a headwind. The angle between the aircraft and the L2 point is used to command the lateral acceleration of the aircraft (conveniently, this corresponds to a command-in-roll angle for the UAV). Tighter control is achieved by shortening the look-ahead distance, but at the risk of oscillations about the path. Figure 6 shows data from an actual flight path of the UAV in the presence of wind. The L2 vector is shown in green, the UAV's position is in blue, and the commanded path is in red.

FLIGHT TEST RESULTS

The UAV employed for flight tests was a modified version of an off-the-shelf Multiplex Mentor radio control (RC) aircraft. Modifying the aircraft's fuselage involved shaving out a significant section of the "cockpit"' (see Photo 3a) to make way for a wireframe pod (see Photo 3b). This pod, made out of balsa wood, securely housed the autopilot and the radio modem (see Photo 3c) in the airframe. The pod has two functions. One, it protects the main avionics in case of a crash. And two, it provides a vibration-isolated environment on which to mount the autopilot. This also makes the core avionics trivial to mount and unmount from the UAV for lab testing and HIL simulation.

Flight tests were performed at UCSC's east field. During the first three weeks of flight tests, the UAV was flown under pilot control. The collected flight data were analyzed offline to verify and validate the inner workings of the autopilot. This initial test period was also used to tune the aforementioned position and attitude estimation filters. Once the position and attitude estimation filters were proven to work, one week of flight tests (more than nine flights) was devoted to tune the PID gains for the inner loop. The tuning process

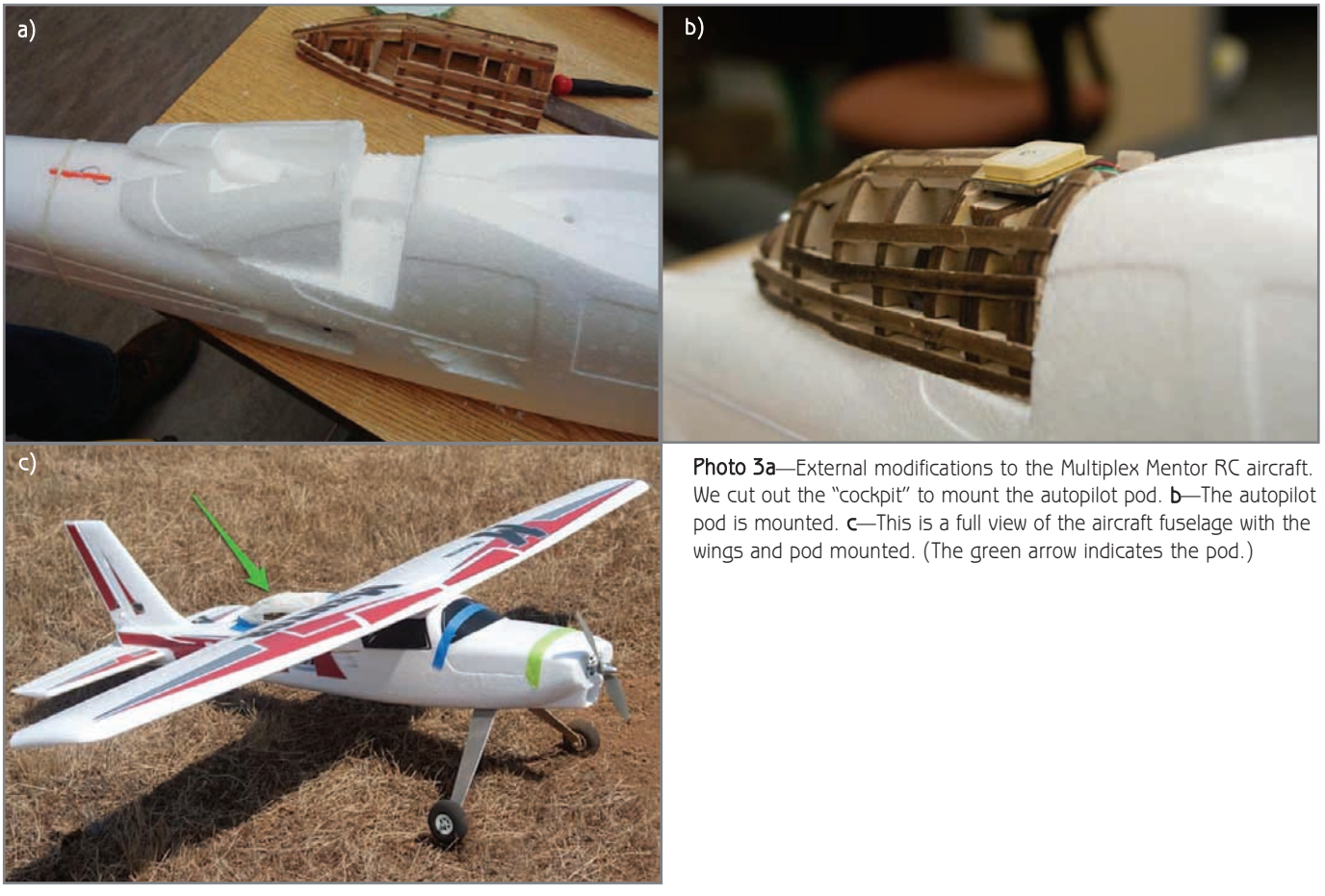


Photo 3a—External modifications to the Multiplex Mentor RC aircraft. We cut out the "cockpit" to mount the autopilot pod. **b**—The autopilot pod is mounted. **c**—This is a full view of the aircraft fuselage with the wings and pod mounted. (The green arrow indicates the pod.)

involved adjusting the gain of each controller one at a time during flight while the safety pilot retained partial control of the aircraft. Any time it was deemed necessary, the safety pilot could recover complete control by flipping a switch on his transmitter that turned the analog MUX back to pass through. Eventually, the safety pilot only had control of the rudder, using it to skid the UAV through turns to keep it within the test area while the inner loop stabilized airspeed, roll, altitude, and turn rate.

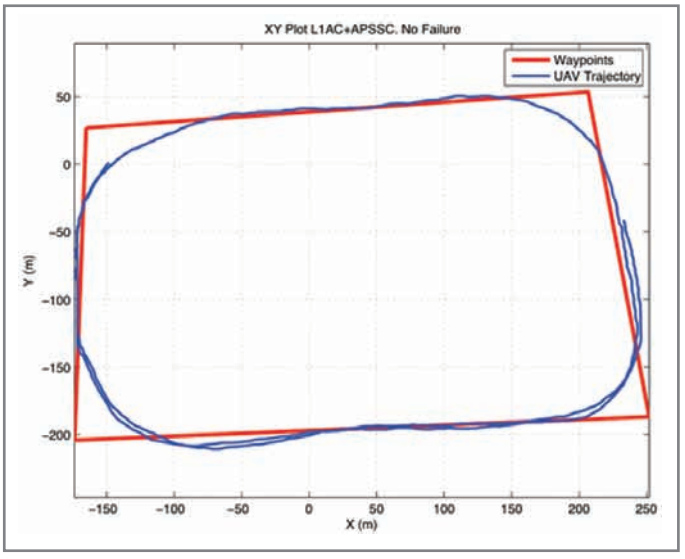


Figure 7—Trajectory plot of the UAV under autonomous control. The ground station was physically located at the origin.

The primary objective of experimental testing was to validate the results correspondence between Simulink and HIL simulations and flight tests. The secondary objective was to validate the complete workflow in the SLUGS: Simulink to HIL simulation to flight test. The waypoint track set for the flight test experiments consisted of four waypoints defining two short and two long legs (roughly defining a rectangle). The flights were always performed within visual range of the safety pilot, thus the short separation between waypoints. For every flight test, the UAV took off under pilot control. The pilot would do several circuits around the path to make sure everything was working correctly, then release the control to the SLUGS autopilot. Figure 7 shows a typical route around the preprogrammed waypoint track under autonomous control.

It is important to stress that all of the SLUGS's control and navigation loops were implemented directly in Simulink. A high-level programming language (such as C, C++, or Java) wasn't used. The Simulink models used for software simulation, HIL simulation, and flight tests are exactly the same. These were first used in Simulink and then compiled and downloaded to the autopilot hardware. The navigation loop was shown to perform robustly, with good tracking results even in the presence of turbulence and wind. The inner and outer loops performed well, and we achieved basic flight and waypoint following.

TAKING FLIGHT

We presented a top-level view of the SLUGS, an open-source UAV autopilot, its ground control station, and its HIL simulator. We presented results for its sensors, temperature compensation, and calibration. Using Simulink as the primary development platform, UAV researchers can easily transition between software simulation, HIL simulation, and flight testing without the need to re-code the algorithms in a traditional programming language.

The SLUGS was flight tested under nominal flying conditions with successful results, but there is room for improvement. We recently adopted the open-source GCS software QGroundControl. This project is in active development and has a thriving community of contributors; but, at its beginnings, it was mostly focused on autonomous quad-copter support. This has changed, and now there is a wide range of tools to support autonomous airplanes. The SLUGS has been slowly adopting all these features.

Also, we're taking steps to adapt the SLUGS's code base so it can work in a wider variety of autonomous vehicles. In particular, work is being done to use the SLUGS in an autonomous boat, a VTOL aircraft, and a ground vehicle. Finally, the SLUGS is currently being adopted in several educational research laboratories in the United States and elsewhere. This will undoubtedly help improve the overall quality of this project as a community of users each adds their own improvements.

Authors' note: We would like to express our deepest gratitude to Greg Horn for working so hard on the board schematics and layout and hand-soldering the components of the first two batches of autopilots. He was also our safety pilot, and more than once he saved us from having to build another UAV. We would also like to thank Vladimir Dobrokhodov for all of his help in making the SLUGS happen and lending his experience as a long-time UAV developer and end user. Finally, we would like to thank Lorenz Meier at ETH Zurich for all of his help in integrating MAVLink into the SLUGS.

Mariano Lizarraga (malife@soe.ucsc.edu) holds a BS in Naval Sciences Engineering from the Mexican Naval Academy, an MSEE and Electrical Engineer's degree from the U.S. Naval Postgraduate School, and a PhD in Computer Engineering from UCSC. He has more than eight years of experience writing software and designing hardware for UAVs. He is currently a Research Fellow in the Computer Engineering Department at the University of California Santa Cruz.

Renwick Curry (rcurry@ucsc.edu) earned an AB in Physics from Middlebury College, and he holds a BS, MS, and PhD in Aeronautics and Astronautics from MIT. He has worked as a faculty member (MIT and Cornell), a spacecraft guidance engineer (Mariner and Apollo projects), and an aviation safety engineer for NASA. Renwick is currently an Adjunct Professor in the Computer Engineering Department at University

of California Santa Cruz, and president of AASI (www.aasi.com), a company specializing in aircraft flight-path optimization.

Gabriel Hugh Elkaim (elkaim@soe.ucsc.edu) holds a BSE in Aerospace Engineering from Princeton University, as well as an MS and PhD in Aeronautics and Astronautics from Stanford University. He is an Associate Professor in the Computer Engineering Department at University of California Santa Cruz, where he works on autonomous vehicle guidance, navigation, and control.

RESOURCES

M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV," Australian National University College of Engineering & Computer Science, 2008.

GitHub Inc., Source code repository, http://github.com/malife/SLUGS

L. Kerhuel, Simulink Embedded Target for PICs, www.kerhuel.eu/wiki/Index.php5, 2010.

R. Mahony, T. Hamel, and J. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control*, Vol. 55, No. 5, 2008.

S. Gleason and D. Gebre-egiabher, *GNSS Applications and Methods (GNSS Technology and Applications)*, Artech House Publishers, 2009.

S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov Stability of Non-Linear Path Following Guidance Method," *Journal of Guidance, Control and Dynamics*, Vol. 30, 2007.

QGroundControl, MAVLink Micro Air Vehicle Communication Protocol, www.qgroundcontrol.org

SLUGS Project, http://slugsuav.soe.ucsc.edu

UC Santa Cruz Autonomous Systems Lab, http://asl.soe.ucsc.edu

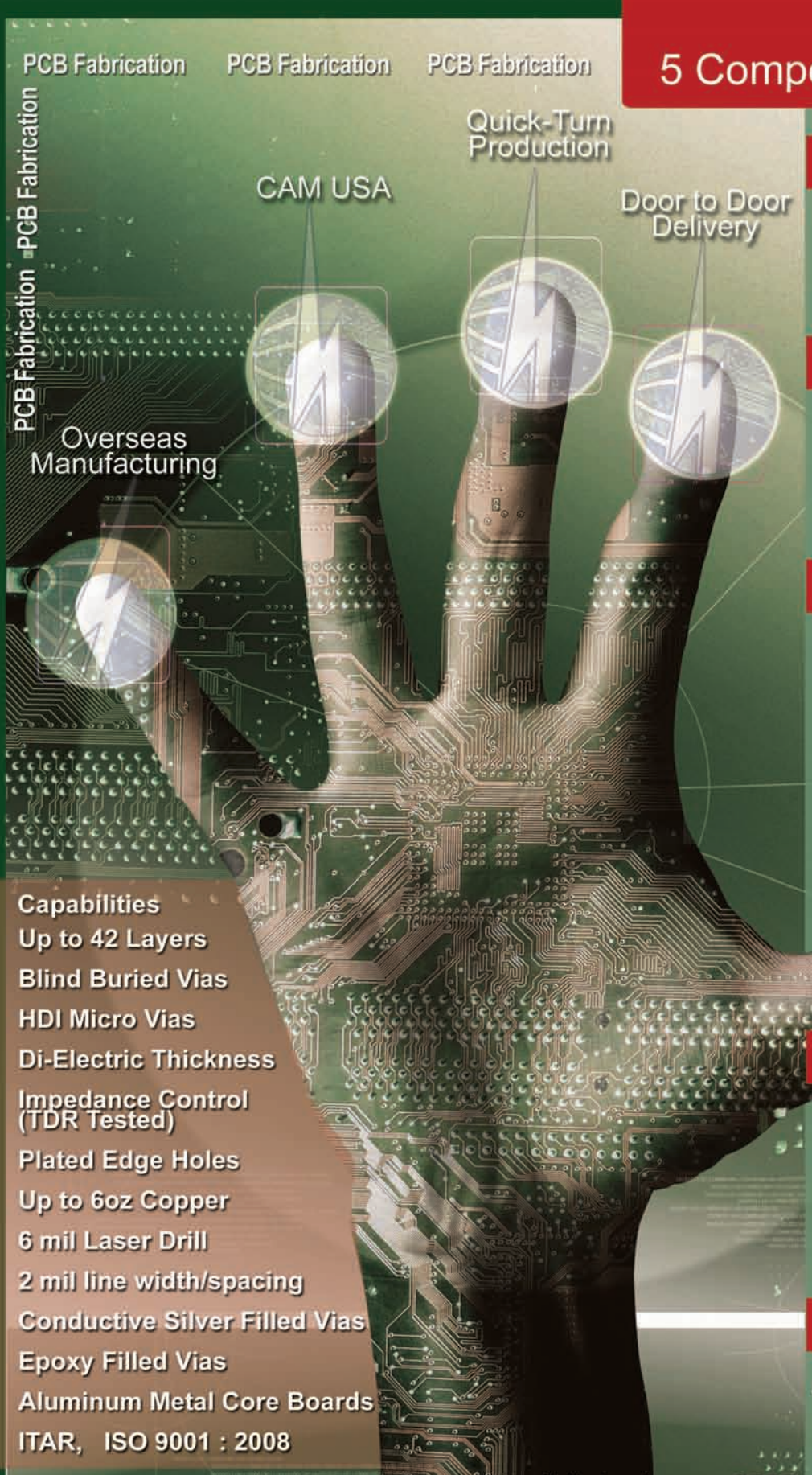
SOURCES

DSP-3000 Fiber optic gyro (FOG)
KVH Industries, Inc. | www.kvh.com

dsPIC33 DSC and MCP9701 Thermistor
Microchip Technology, Inc. | www.microchip.com







5 Competitive Advantages

Overseas Manufacturing .1

Imagineering, Inc. enjoys the reputation of being one of the most experienced & successful offshore PCB suppliers.

CAM USA .2

Our Illinois based DFM office has eight fully staffed CAD / CAM stations. Within hours of receipt of new Gerber files, our highly experienced DFM engineers conduct thorough and precise analysis.

Quick-Turn Production .3

Imagineering offers small volume production in 5-6 days and medium to large volume production in 2-3 weeks.

Significant Price Saving

Shipping Logistics .4

With Imagineering there is no need to deal with multiple suppliers, language barriers, customs headaches, and shipping logistics. We do it all for you ..and deliver door-to-door.

Significant Price Saving .5

Our global buying power combined with the capabilities of our overseas manufacturers translate into tremendous savings to our customers.

Capabilities

- Up to 42 Layers
- Blind Buried Vias
- HDI Micro Vias
- Di-Electric Thickness
- Impedance Control (TDR Tested)
- Plated Edge Holes
- Up to 6oz Copper
- 6 mil Laser Drill
- 2 mil line width/spacing
- Conductive Silver Filled Vias
- Epoxy Filled Vias
- Aluminum Metal Core Boards
- ITAR, ISO 9001 : 2008

Last year we successfully took over 6000 prototype part numbers to production

imagineering inc. 847-806-0003 www.PCBnet.com email: sales@PCBnet.com

25 YEARS IN BUSINESS...AND STILL GOING STRONG

5 Competitive Advantages

PCB Assembly PCB Assembly PCB Assembly

PCB Assembly PCB Assembly

1. Minimizing Procurement Issues

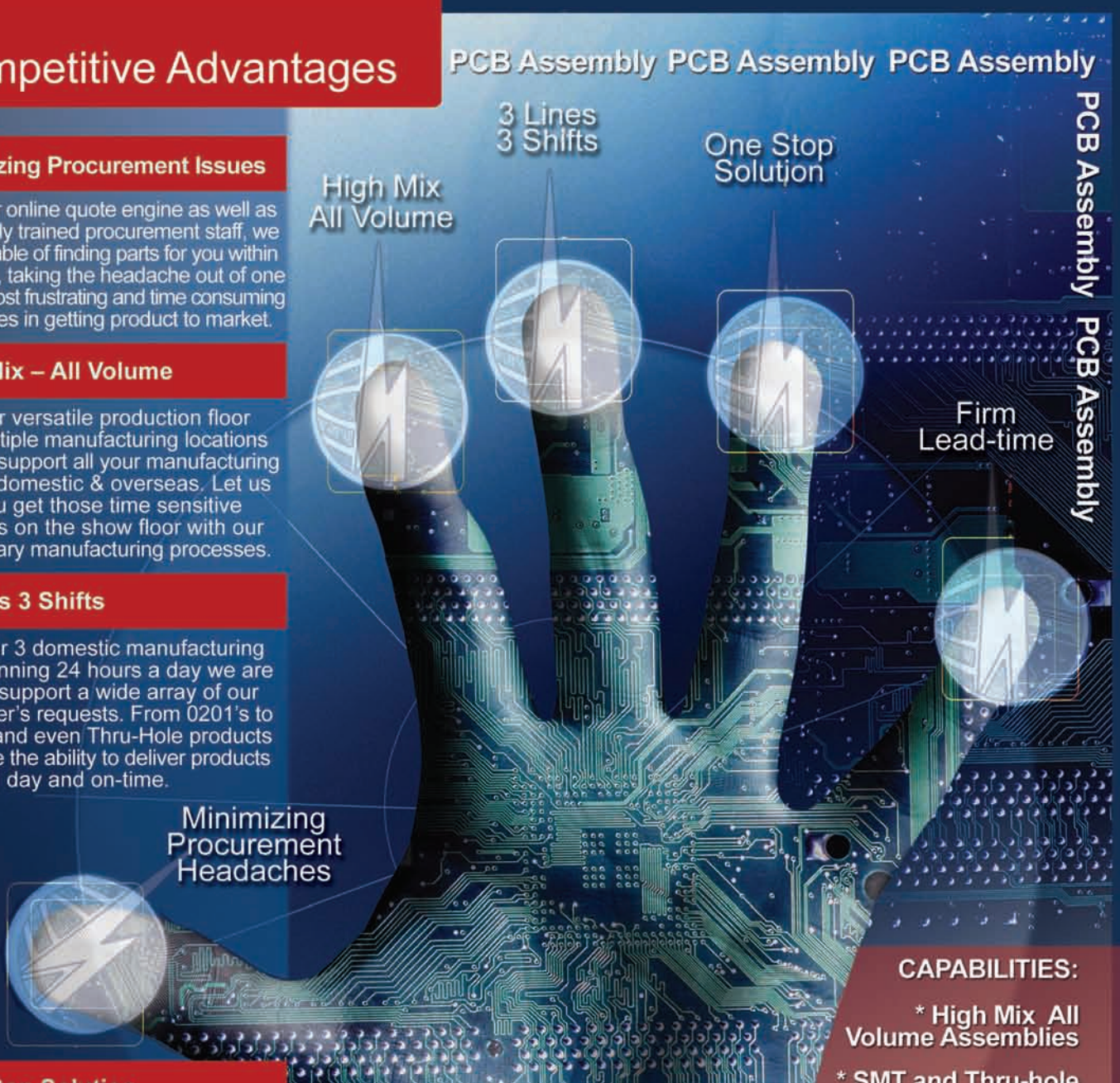
With our online quote engine as well as our highly trained procurement staff, we are capable of finding parts for you within minutes, taking the headache out of one of the most frustrating and time consuming processes in getting product to market.

2. High Mix – All Volume

With our versatile production floor and multiple manufacturing locations we can support all your manufacturing needs, domestic & overseas. Let us help you get those time sensitive products on the show floor with our proprietary manufacturing processes.

3. 3 Lines 3 Shifts

With our 3 domestic manufacturing lines running 24 hours a day we are able to support a wide array of our customer's requests. From 0201's to BGA's and even Thru-Hole products we have the ability to deliver products within 1 day and on-time.



4. One Stop Solution

Why deal with the headache of sourcing from multiple vendors and part suppliers? Let Imagineering handle all that for you. With our excellent reputation of Manufacturing PCBs to our Proprietary Parts Search Engine and Manufacturing Process, we are a TRUE one stop solution for all your Contract Manufacturing Needs.

5. Firm Lead-times

Don't let other CM's throw you around with soft deadlines. When you want something done in 1 day we won't tell you 1-3 days. We will get it done in 1 day, and in some circumstances even sooner!

CAPABILITIES:

* High Mix All Volume Assemblies
* SMT and Thru-hole
* Double Sided SMT Assembly
* Leaded and Lead Free Assemblies
* Passives down to 0201
* VFBGA and BGA
* Fine Pitch to 8 Mils
* BGA Re-work and Re-Ball
* PCBA Rework – Remove and Replace Components

ITAR, ISO 9001 : 2008

Over the past 5 years, 90,000 prototypes have been successfully delivered from overseas to over 5000 customers

imagineering inc. 847-806-0003 www.PCBnet.com email: sales@PCBnet.com

25 YEARS IN BUSINESS...AND STILL GOING STRONG

by Thomas Mitchell (USA)

Getting Started with Microprogramming (Part 2)

A Look at the Meta-Assembler

The first article in this series detailed the architecture, programmer model, instruction set, and general operation of a simple microprogrammed processor. Here you learn about real-world microprogramming and the meta-assembler.

In the first article in this series, I set the groundwork for building a custom microprogrammed processor suitable for an FPGA, and I detailed the "what" of the simple microprogrammed processor (SMP). Now I'll tell you about the "how." Real-world microprogramming is the focus of this article. So, to begin, let's look at the meta-assembler.

WinTIM

Previously, I used WinTIM—which was developed by Eric Van Heest and Mitch Kispet as a class project at the Georgia Institute of Technology—to generate the test vectors to verify my Advanced Micro Devices Am2910 VHDL design against real hardware. I used WinTIM for the development of the SMP because, although it has its limitations, it's free and the source is available.

WinTIM is based on the program TIM, which is Texas Instruments's version of Hi-Level Engineering's HALE meta-assembler. WinTIM is a meta-assembler, and it's a program that can be configured to convert Assembly language mnemonics from different processors into a binary form. A meta-assembler makes no assumptions about the format of the Assembly language because the format must be defined before the source can be processed. In fact, a meta-assembler can generate binary patterns (an executable, if you like) for many purposes, including assembling a source file into processor-executable binaries, microprograms, or look-up tables.

In the SMP development, I used WinTIM to develop the SMP microcode and to develop Assembly language programs for the SMP. The first step is to create a "definition" file. The purpose of the definition file is to define the structure of the microword and symbolic values, which are assembled to generate microwords. This concept can be better explained with an example of a definition file.

MICROPROGRAM DEFINITION FILE

Listing 1 is an example of a constant definition in a WinTIM definition file. The TITLE directive is just what it says, a title for the definition file. WORD is a directive

Listing 1—An example of a constant definition in a WinTIM definition file

```
;*************************************************************;
; Am2910 Instruction:            I3 - I0                      ;
; Logical Microword Bits:       71 - 68                       ;
; Physical Microword Bits:                                    ;
;*************************************************************;
;
; Am2910 Microinstruction Definitions
;
JZ:      EQU    H#0     ; Jump to location zero
CJS:     EQU    H#1     ; Conditional jump to subroutine
JMAP:    EQU    H#2     ; Jump map
CJP:     EQU    H#3     ; Conditional jump to pipeline
PUSH:    EQU    H#4     ; Push/Conditional Load Counter
JSRP:    EQU    H#5     ; Jump to register or pipeline subroutine
CJV:     EQU    H#6     ; Conditional jump vector
JRP:     EQU    H#7     ; Jump to register or pipeline location
RFCT:    EQU    H#8     ; Repeat loop until cnt=0 (loop on stack)
RPCT:    EQU    H#9     ; Repeat loop until cnt=0 (loop on pipeline)
CRTN:    EQU    H#A     ; Conditional return from subroutine
CJPP:    EQU    H#B     ; Conditional jump to pipeline and pop stack
LDCT:    EQU    H#C     ; Load counter and continue
LOOP:    EQU    H#D     ; Test end of loop
CONT:    EQU    H#E     ; Continue
TWB:     EQU    H#F     ; Three way branch
```

Listing 2—The eight sections of the microword defined using the sub-format (SUB) directive. The definition for the SMP format is at the bottom of the listing.

```
;************************************************************;
;       Microword Field Format Definitions                   ;
;                                                            ;
;************************************************************;
AM2910: SUB   4VH#E, 16V%:H#000%, 4VB#0111
; Controls the fields for the Am2910 Microprogram Sequencer

AM2901: SUB   3VB#001, 3VB#011, 3VB#100, 4VH#F, 4VH#F, 1VB#0
; Controls the fields for the Am2901 ALU

REGLD:  SUB   9VB#000000000
; Controls register load enables and interrupt enable set and clear

MUX:      SUB  6VB#00000
; Controls the processor multiplexers

RWEN:     SUB  4VB#0000
; Controls Memory and I/O read and write enables

SHIFT:    SUB  3VB#000
; Controls shift multiplexer

IMAP:     SUB  2VB#00
; Controls which instruction map is used for JMAP

PSTATE:  SUB  6VB#000000
; Indicates processor state

SMP:     DEF  AM2910, AM2901, REGLD, MUX, RWEN, SHIFT, IMAP, PSTATE
```

that sets the microword size—in this case, 72 bits. What follows are comments (they begin with a semicolon ";") about the subsequent definitions. The "jump to zero" (JZ) Am2910 instruction is defined using the EQU (equate) directive as H#0 (i.e., a hex digit with a value of 0). The other 15 Am2910 instructions are similarly defined. In Listing 2, the eight sections of the microword are defined using the sub-format (SUB) directive. For example, the Am2910 section is defined with a sub-format with three fields. The first field (4VH#E) is a 4-bit variable field with a default value of hex E. The second field (16V%:H#000%) is a 16-bit variable field with a default value of hex 0000 (i.e., a 16-bit binary value of all zeros). The % after 16V indicates the values assigned to this field will be right-justified if they are less than 16 bits. The : after the 16V% indicates that values assigned to this field greater than 16 bits will be truncated to 16 bits. The third field (4VB#0111) is a 4-bit variable field with a default value of binary 0111.

So what does all this mean? This statement defines the format of the Am2910 section and sets default values. These values are as follows: The Am2910 instruction is a hex E—that is, a "Continue" (CONT) instruction, with the Am2910 direct input set to hex 0000 (12 bits right-justified and expanded to 16 bits). The test input is "ALWAYS"—that is, the condition code input to the Am2910 is a logical TRUE. Because each field of the sub-format is defined as a variable, they can be set to any value with the same number of bits. However, if the "V" is omitted, the sub-field is defined to be the constant value; or, if the value is omitted, it is a "don't care." This feature can be used to define multiple sub-formats where differing bits may be variable or not, depending on the context. Only one format is used in the SMP microprogram.

The definition for the SMP format is at the bottom of Listing 2. The SMP format consists of the eight previously defined sub-formats. There is no overlap between the different sub-formats, but different formats can be defined from different combinations of sub-formats and fields. The SMP microprogram uses only one format, but I could just as easily have defined different formats with most of the bits fixed and only a few variable fields.

MICROPROGRAM SOURCE FILE

The microprogram source file is akin to a normal Assembly language source file except it's coupled with a definition file that explains which bit patterns correspond to which source code "instructions." Listing 3 shows an excerpt from the SMP WinTIM source file. Like the definition file, there are typically directives that precede the actual source. The TITLE directive states the title of the source file, the LIST directive controls the output of the listing file generated. The "B" parameter of the LIST directive indicates that the formatted object code will follow the listing of the source, rather than be interspersed with the source code. The "W" parameter of the LIST directive indicates that the listing should word-wrap rather than truncate lines longer than the listing width. The FORM directive defines the output of the formatted object code in the listing where a "1" indicates a binary digit, a "3" indicates an octal digit, a "4" indicates a hex digit, and a "b" indicates a blank space. The WIDTH directive sets the width of the listing file. The ORG directive sets the location for the object code output.

The first actual line of source code starts with MAIN:, which is a location label. In this case, it has a value of hex 0 which is the location value previously defined by the ORG directive. What follows is SMP, which indicates that the following parameters will be applied to the SMP format. CONT is the Am2910 instruction, H#0% is the Am2910/Am2901 direct input, and ALWAYS is the test input control. The resulting value will be hex E00007, which is only 24 bits. Where is the rest of the microword? The next seven lines all start with a forward slash, which indicates that this line is a continuation of the previous line. Therefore, the eight lines constitute one microword of the microprogram. The comments at the end of each line are a convenience to keep track of which part of the microword the parameter applies to. The resulting

microword (formatted according to the FORM directive) is:

```
00000 E 0000 07 007 FF 0
  000000000 0000 00 00000 00 00
```

Reading from left to right, hex 00000 is the microprogram location (not part of the microword). Hex E is the Am2910 instruction. Hex 0000 is the Am2910/Am2901 direct input. Binary 0 and octal 7 are the test input polarity and test input select, respectively. Octal 007 is the Am2901 instruction. Hex FF are the Am2901 A and B addresses. Binary 0 is the Am2901 carry-in. Binary 000000000 is the register load enables. Binary 0000 and 00 are the multiplexer controls. Binary 0000 is the read/write enables. Octal 0 is the shift control. Binary 00 is the IMAP control. Octal 00 is the processor state.

So, is that clear? Well, maybe not. Microprogramming takes a little bit of effort to learn. What I find helpful is being consistent in the writing of micro-instructions and good comments. In the SMP microprogram, every micro-instruction uses the same eight-line format with comments to indicate the different sections of the microword. If you use a similarly consistent format, then writing a line of microcode becomes a matter of plugging in values for the relevant fields and leaving the rest at their default values. When writing a microprogram, it is important to have a clear picture of the structure of the microword, the values that can be applied to the different fields, and how these values translate into hardware signals.

Listing 3—An excerpt from the SMP WinTIM source file

```
ORG H#00

;
; initialize Am2901 Q register to zero
;

MAIN:    SMP CONT, H#0%, ALWAYS,            ; Am2910
/            QREG, ADD, DZ,,, NCI,          ; Am2901
/            NOLOAD,                        ; REGLD
/            DEFMUX,                        ; MUX
/            NORW,                          ; RWEN
/            DEFSHIFT,                      ; SHIFT
/            DEFIMAP,                       ; IMAP
/            SINIT                          ; PSTATE

; Instruction fetch
; Loads MAR with PC and increments the PC
;

FETCH:   SMP ,,,                            ; Am2910
/            RAMA, ADD, ZA, PC, PC, CI,     ; Am2901
/            MARLD,                         ; REGLD
/            DEFMUX,                        ; MUX
/            NORW,                          ; RWEN
/            DEFSHIFT,                      ; SHIFT
/            DEFIMAP,                       ; IMAP
/            SFETCH                         ; PSTATE

;
; Asserts MEMRD until MWAIT is true,
; i.e. read memory addressed by MAR
; MDIR is loaded while waiting for MWAIT
;

         SMP CJP, $%, NMWAIT,               ; Am2910
/        ,,,,,,                             ; Am2901
/            MDIRLD,                        ; REGLD
/            DEFMUX,                        ; MUX
/            MEMRD,                         ; RWEN
/            DEFSHIFT,                      ; SHIFT
/            DEFIMAP,                       ; IMAP
/            SFETCH                         ; PSTATE
```

"PROGRAMMING" THE VHDL

Up to this point, the HDL and microprogram development have been independent. Now, we need to use the binary generated by WinTIM to initialize the block RAMs in the control store. This is the easy part, because I wrote a program to process the .mif file output of WinTIM and turned it into a text array, which can be used in the control store VHDL code to initialize the RAM. The following runs the program:

```
mif2array <mif file> <array file> <array
  size>
```

The array size sets the number of values in the array regardless of the number of values in the .mif file. This is necessary because the initialization of an array in VHDL expects all of the values of the array to be initialized whether they are used or not. Values not defined in the .mif file are initialized to zero. If the array size parameter is omitted, the array will use only the .mif file values.

To install the microcode into the control store VHDL, open the array file and copy everything. Then open the Control_Store.vhd file and find the definition of the signal RAM. Paste the copy buffer after the line that says "Insert microprogram here" and save the file. Use the Xilinx ISE program to rebuild the bit file for the target device. Clearly, this is not the fastest way to update microcode, but Xilinx provides a program, "data2mem,"

Listing 4—The .bmm file for the SMP

```
ADDRESS_SPACE control_store RAMB36 [0x00000000 : 0x000001ff]
BUS_BLOCK
U1/inst_Mram_mem1 [0:35];
U1/inst_Mram_mem11 [36:71];
END_BUS_BLOCK;
END_ADDRESS_SPACE;
```

ESC sensors expo & conference

Co-Located Event: June 6-8, 2011,
Donald E Stevens Convention Center, Rosemont, IL

Conference Tracks:

- Designing for Embedded Linux or Android
- Graphics, Displays, and Lighting
- Medical Systems
- Networking and Connectivity
- Open Source Software
- Project Management
- Real-Time System Development
- Robotics and Motion Control
- Safety and Security in a Stuxnet world
- Software Debugging Techniques

Bring this ad and your expo pass is FREE onsite or use promo code EXPOPASS to register online for FREE.

Event Highlights Include:

- FREE Expo Floor party in the ESC Expo Zone
- FREE Sponsored Sessions
- ESC Theater Events
- Live Teardowns
- AND MORE! Register today!

Learn today. Design tomorrow.

ESC

Chicago • June 6 - 8, 2011
www.embedded.com

sensors expo & conference

Pre-Conference: June 6, 2011
Conference & Expo: June 7-8, 2011
Rosemont, IL • Donald E. Stephens Convention Center

25+ YEARS OF EXCELLENCE

www.sensorsmag.com/sensorsexpo

The Only Industry Event in North America Exclusively Focused on Sensors and Sensor Integrated Systems! Find the Solutions to Your Sensors and Sensing Technology Challenges!

Gain the knowledge you need from leading experts and peers in the sensors industry.

The Conference Program includes more Technical Sessions in Tracks covering the topics that are most important to you and your business, including:

- MEMS
- Energy Harvesting
- Wireless Sensor Networking
- Novel Approaches to Measurement & Detection
- Wireless Sensor Data
- Mobile Sensing – NEW for 2011!
- Cutting-Edge Technologies – NEW for 2011!

Identify specific solutions to your most difficult sensing, detection and control-related challenges on the expo floor.

Sensors Expo brings together the largest and best-in-class showcase of sensing technologies and systems for attendees to evaluate and make informed decisions.



Tuesday, June 7
9:00 AM – 10:00 AM

Opening Keynote:
Dr. Hugh Herr
Biomechatronics Researcher

Don't Miss these Expo Floor Highlights:

CO-LOCATED WITH: Learn today. Design tomorrow. ESC
ENERGY HARVESTING PAVILION
MEMS PAVILION
TECHNOLOGY UPDATE THEATER

SUBSCRIBERS:
Visit www.sensorsmag.com/sensorsexpo to register or call 866-817-5048.
Use discount code A350E for an EXTRA $50 OFF a Gold or Main Conference Pass!

Scan this QR code to Register Now on Your Mobile Device!
Get the free mobile app at http://gettag.mobi

MEDIA SPONSOR: EETimes
PRODUCED BY: QUESTEX MEDIA
OFFICIAL PUBLICATION: sensors

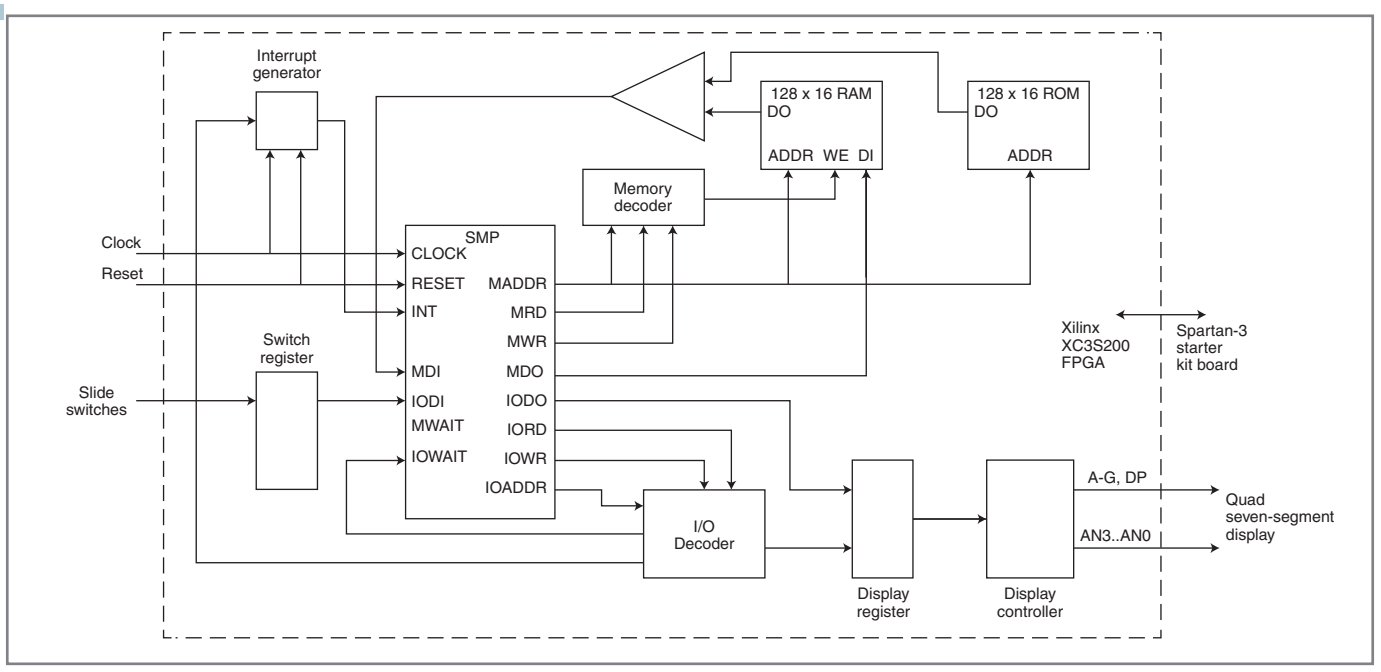


Figure 1—This is a block diagram of the SMP demonstration application that includes an SMP, 128 x 16 RAM, 128 x 16 ROM, the memory decoder, the I/O decoder, the display register, the display controller, the switch register, and the interrupt generator.

that can convert various binary formats into other formats, such as VHDL or Verilog source code. It can also modify the configuration bits for a block RAM in an existing .bit file. To use this utility, I wrote another program that can convert a .mif file into a .mem file:

```
mif2mem <mif file> <mem
  file> <array size>
```

This program works the same as `mif2array` except for the output. Use the following command from the command line to modify the existing .bit file:

```
data2mem -bd <mem file> -bm
  <block RAM memory map
  file> -bt <bit file> -o b
  <new bit file>
```

The mem file is the file created by `mif2mem`. The bit file is the existing .bit file created by ISE. The block RAM map file describes which logical RAMs correspond with which physical RAMs. Listing 4 shows the .bmm file for the SMP. The first line defines the address space for the control store mapped to a RAM36 (36-bit-wide block RAM device) as starting at hex 0 and ending at hex 1FF (i.e., 512 locations). The third line defines the first 36 bits of the RAM mapped to `U1/inst_Mram_mem1`, and the second definition defines the second 36 bits mapped to `U1/inst_mRam_mem11`. The last two lines define the end of the bus block and address space definitions.

The second "programming" of the VHDL from the microcode involves defining the entries of the instruction register map multiplexer look-up tables. Unfortunately, this step is a manual process. The look-up tables are

Listing 5—An excerpt from SMPASM.def, which defines the constants for the SMP instructions, address modes, branch conditions, and registers. The bottom of the listing shows two defined microword formats: `SMPINSTR` and `IMMEDVAL`.

```
;
; Values for Condition field
;

ZERO:       EQU      2B#00
CARRY:      EQU      2B#01
SIGN:       EQU      2B#10
OVR:        EQU      2B#11

;
; Values for Destination and Source fields
;

R0:         EQU      3B#000
R1:         EQU      3B#001
R2:         EQU      3B#010
R3:         EQU      3B#011
R4:         EQU      3B#100
R5:         EQU      3B#101
R6:         EQU      3B#110
R7:         EQU      3B#111

;
; SMP Instruction Field Formats
;

SMPINSTR:   DEF      5VB#00000, 3VB#000, 2VB#00, 3VB#000, 3VB#000
IMMEDVAL:   DEF      16V%:H#0000
```

defined in the IR_Map_Mux.vhd file, but the values for the tables are defined in smp_const.vhd. In the sections titled Pre-Operation LUT, Instruction LUT, and Post-Operation LUT are the constant definitions that define the jump points for each of the maps. The values that are assigned can be determined from the Assembly phase listing file. The pre-operation entry points are designated: LREGREG, LREGIND, LREGIMM, LREG1OP, LINDREG, LINDIND, LINDIMM, and LIND1OP. The post-operation entry points are designated: LREG and LIND. The instruction entry points all have the form L<instruction mnemonic>. Search the Assembly phase list file and find each of the labels for the entry points. The address can be found on the last line of the micro-instruction. The last constant definition in the Instruction LUT section is the value inst_NOP, which does not define an actual instruction; rather it is a place holder for undefined instructions. The value of inst_NOP is assigned the address of the location MAINLOOP, which is the beginning of the main processing loop.

TESTING & VERIFYING

At this point, all of the pieces were in place, and I started to test the SMP. I say that I "started" testing, but actually I had been testing the components all along. I was really verifying that the SMP microcode implemented the programmer's model of the SMP. To do this, I tested a stand-alone SMP using simulation. I verified that the SMP

Listing 6—An example of the macros used to define the load (LD) instruction for all of the possible address modes

```
;
; Load Register Instruction
; LD <dest>, <src>
;

LD.RR:   MACRO      RDEST, RSRC
         SMPINSTR   ILD, REGREG,, RDEST, RSRC
         ENDM

LD.RN:   MACRO      RDEST, RSRC
         SMPINSTR   ILD, REGIND,, RDEST, RSRC
         ENDM

LD.RI:   MACRO      RDEST, VAL
         SMPINSTR   ILD, REGIMM,, RDEST,
         IMMEDVAL   VAL
         ENDM

LD.NR:   MACRO      RDEST, RSRC
         SMPINSTR   ILD, INDREG,, RDEST, RSRC
         ENDM

LD.NN:   MACRO      RDEST, RSRC
         SMPINSTR   ILD, INDIND,, RDEST, RSRC
         ENDM

LD.NI:   MACRO      RDEST, VAL
         SMPINSTR   ILD, INDIMM,, RDEST,
         IMMEDVAL   VAL
         ENDM 16V%:H#0000
```


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

Designed a Brand New Device?

Learn to search online databases to see if your device is cutting edge and may be eligible for a patent by using this new instructional DVD!



MIDWEST PATENT SERVICES
info@midwestpatentservices.com
www.midwestpatentservices.com

Listing 7—The Assembly language program

```
DISPREG:    EQU       H#0010          ; Display Register
SWREG:      EQU       H#0020          ; Switch Register
INTGEN:     EQU       H#0030          ; Interrupt Generator

ORG H#0000

START:      LDSP.I    H#FFFF          ; Initialize Stack Pointer
            LDIV.I    ISR             ; Point interrupt vector to ISR

            LD.RI     R0, H#0000      ; Initialize counter to zero
            LD.RI     R1, H#FFFF      ; Initialize R1 to H#FFFF
            LD.RI     R2, H#FF80      ; Initialize R2 pointer to H#FF80
            LD.NI     R2, H#0000      ; Clear memory location H#FF80 (interrupt flag)
            EI                        ; Enable interrupts

MAINLOOP:   CMP.NI    R2, H#FFFF      ; Does (R2) - H#FFFF = 0? Flag Set?
            CJMP.I    INCRDISP, ZERO  ; If true jump to INCRDISP
            JMP.I     MAINLOOP        ; else jump to MAINLOOP

INCRDISP:   DI                        ; Disable interrupts
            LD.NI     R2, H#0000      ; Clear interrupt flag
            INC.R     R0              ; Increment counter
            IN.RI     R1, SWREG       ; Read the Switch Register
            ADD.RR    R0, R1          ; Add the value to counter
            OUT.RI    R0, DISPREG     ; Output resultant to Display Register
            EI                        ; Enable interrupts
            JMP.I     MAINLOOP        ; Continue looping

ORG H#0070

ISR:        LD.NI     R2, H#FFFF      ; Set interrupt flag
            IN.RI     R3, INTGEN      ; Read interrupt generator (clear interrupt)
            RETI                      ; Return from interrupt
```

fetched instructions and operands correctly, that the SMP responded to interrupts correctly, and that every instruction worked correctly. However, I felt that a demonstration in hardware, even a simple application, would be more satisfying. My goal in the demonstration was to show that the SMP is a functional processor, but at the same time not to get caught up in an involved hardware and software project. Therefore, I wanted the demonstration to use simple hardware and software. I chose the Xilinx Spartan-3 Starter Kit board because the available I/O devices were simple yet sufficient to the task. The features I used on this board were eight slider switches, one push button switch, and a quad seven-segment LED display.

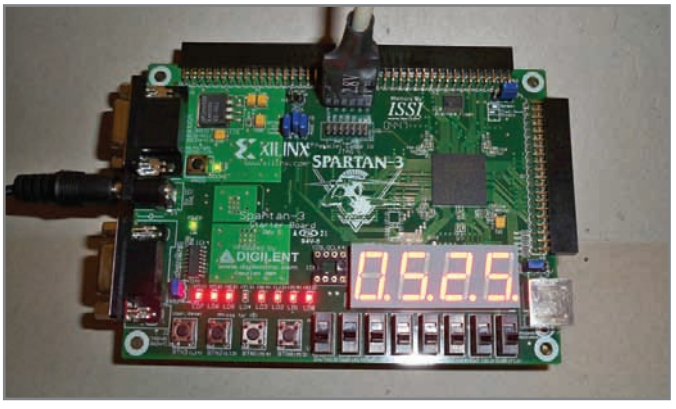


Photo 1—The target board. The demonstration program is running.

Figure 1 shows the SMP demonstration application. It consists of an SMP, a 128 × 16 RAM, a 128 × 16 ROM, the memory decoder, the I/O decoder, the display register, the display controller, the switch register, and the interrupt generator. The memory decoder selects whether the RAM or ROM output is directed to the SMP MDI input, generates the RAM write signal, and generates the memory wait (MWAIT) signal. Likewise, the I/O decoder selects whether the output of the switch register or the display register is output to the SMP IODI input, generates the display register write signal, generates the interrupt generator read signal, and generates the IOWAIT signal. The interrupt generator generates an interrupt every 500 ms. The interrupt is cleared when the SMP reads the interrupt generator register. The display controller scans the four seven-segment digits, selects the corresponding hex digit from the display register, and translates the hex digit into the seven-segment values.

The goal of the SMP software was to demonstrate a few of the SMP's features. The primary requirements were to test instruction execution, interrupt response, the address modes, and access to I/O devices. To implement the program, I used WinTIM again, but this time for assembling SMP instructions. Listing 5 is an excerpt from SMPASM.def, which defines the constants for the SMP instructions, address modes, branch conditions, and registers. At the bottom, two microword formats are defined: SMPINSTR and IMMEDVAL. These formats are used in macros that define the values for

each instruction and address mode. Listing 6 shows an example of the macros used to define the load (LD) instruction for all of the possible address modes. The format for all opcodes is:

```
<instruction mnemonic>.
  <destination operandmode>
  <source operand mode>
```

The source and destination operand modes are: register direct (R), register indirect (N), and immediate (I). Two operand instructions will have both the destination operand and source operand modes, single operand instructions will have either the destination operand or source operand mode, and a few instructions will have neither. The input (IN) instruction uses the source operand as an I/O address, and the output (OUT) instruction uses the destination operand as an I/O address.

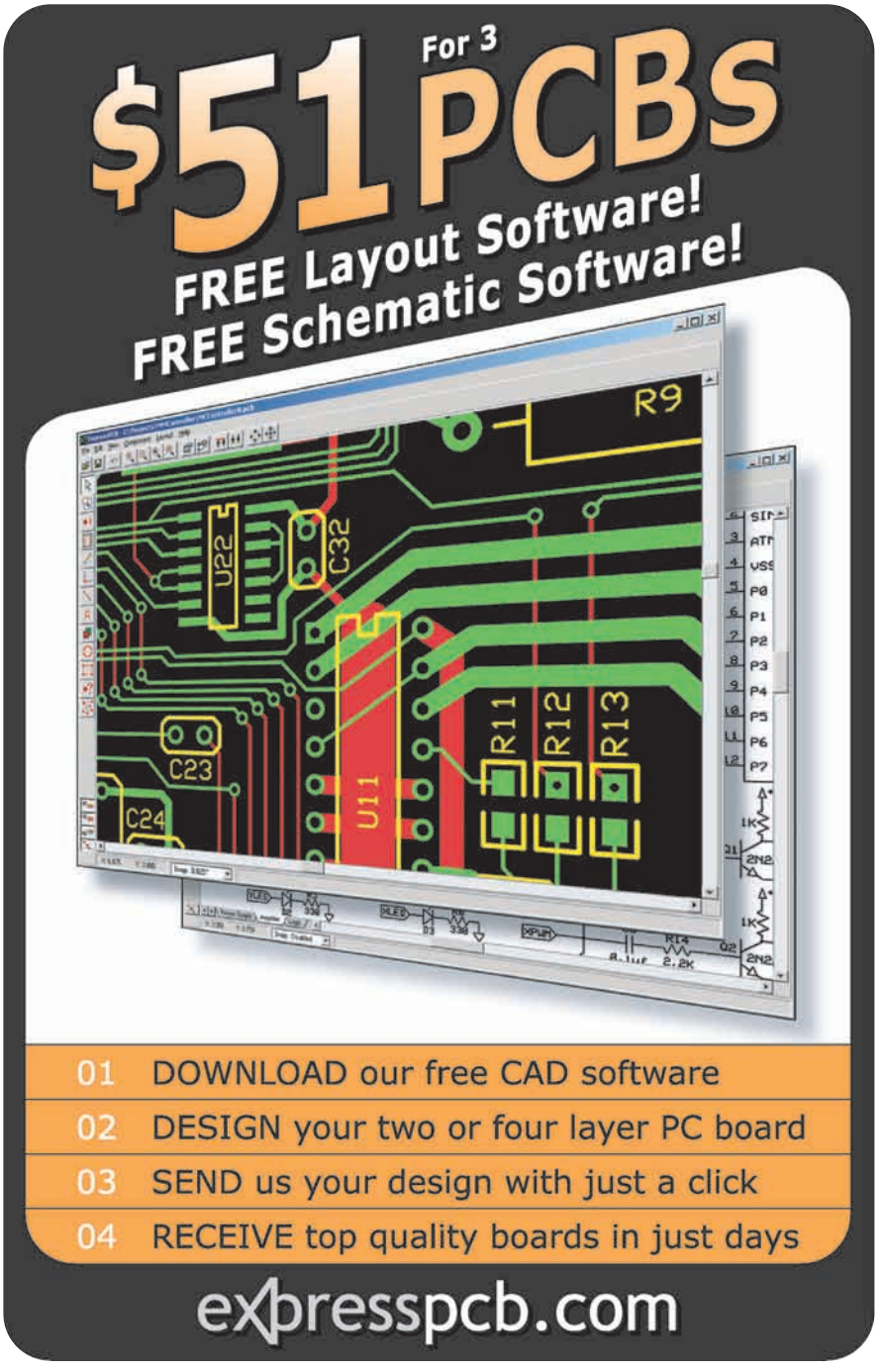
Listing 7 shows the actual Assembly language program. The program consists of a processing loop that loops back to the top of the loop (MAINLOOP) when the interrupt flag is not set; otherwise, it falls through to the increment display code. The interrupt flag is set by the interrupt routine, which responds to interrupts generated by the interrupt generator. INCRDISP increments a counter, reads the value of the switch register, adds it to the counter, and writes the result to the display register. Photo 1 shows the target board with the demonstration program running. The net result of this effort is a quad seven-segment display that increments every half second. While this may not sound impressive, consider that the demonstration program is running on completely reconfigurable hardware programmed from VHDL, microcode, and Assembly code. The real value of this project is the experience I gained by developing a nontrivial application on an FPGA using microprogramming.

THE SOUL OF THE SMP

Now that I have a working custom-made processor, what am I going to do with it? There is certainly a lot I could do to improve the SMP, such as adding additional instructions, adding 8-bit operations, and streamlining its operation. I did not design the SMP with a particular application in mind. I just wanted to design my own processor. It certainly was fun working out the architecture and implementing the design, and it was exhilarating to watch the SMP fetch and execute its first instruction. Whether or not I use the SMP in a future application, I now have all of the tools I need to build microprogrammed machines for various applications.

I started my 2009 *Circuit Cellar* article, "Building Microprogrammed Machines with FPGAs," by talking about Tracy Kidder's book *The Soul of a New Machine.* Now I can say that I actually created a new processor. Certainly the SMP is not as complex as the Data General MV8000, but both were implemented with microprogramming.

In this series, I attempted to show how microprogramming can be used



to build real machines, using a processor as an example. But microprogramming, in its heyday, was just as commonly used to build special processing systems. Using the SMP architecture and a change of microcode, I could just as easily build a special-purpose processor that runs completely on microcode.

You now know how to implement microprogrammed machines in programmable logic. You also understand how CPUs work, albeit just the basics. I hope that I have, at least in some small measure, accomplished my goal. I feel microprogramming can be a useful addition to a digital designer's toolkit. Sure, microprogramming can be difficult, but if you're like me, you'll enjoy the thrill and challenge of manipulating dozens or hundreds of control signals in a complex digital system to bring it alive. Microprogramming provides unprecedented control over complex digital systems, but it is not a skill that's beyond the abilities of mortal engineers. Just ask Judge William Gray who presided over the *NEC v. Intel* case in 1989.

Thomas Mitchell (thmitche@gmail.com) is a registered professional engineer who has worked for the Department of Defense for the last 31 years. He graduated from the University of Delaware with Bachelor's degrees in Electrical Engineering and Physics. Thomas later received Master's degrees in Electrical Engineering and Applied Physics from The Johns Hopkins University. He has worked on numerous high-speed digital designs of components, boards, and systems. Thomas has implemented designs with ECL, TTL, and CMOS using discrete logic (SSI/MSI/LSI/VLSI), programmable logic (PALs, complex PLDs, and FPGAs), microprogram sequencers, and microprocessors.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/250.

RESOURCES

AMD Datasheets and Am2900 Bit-slice course notes, Bitsavers.org, www.computer-refuge.org/bitsavers.

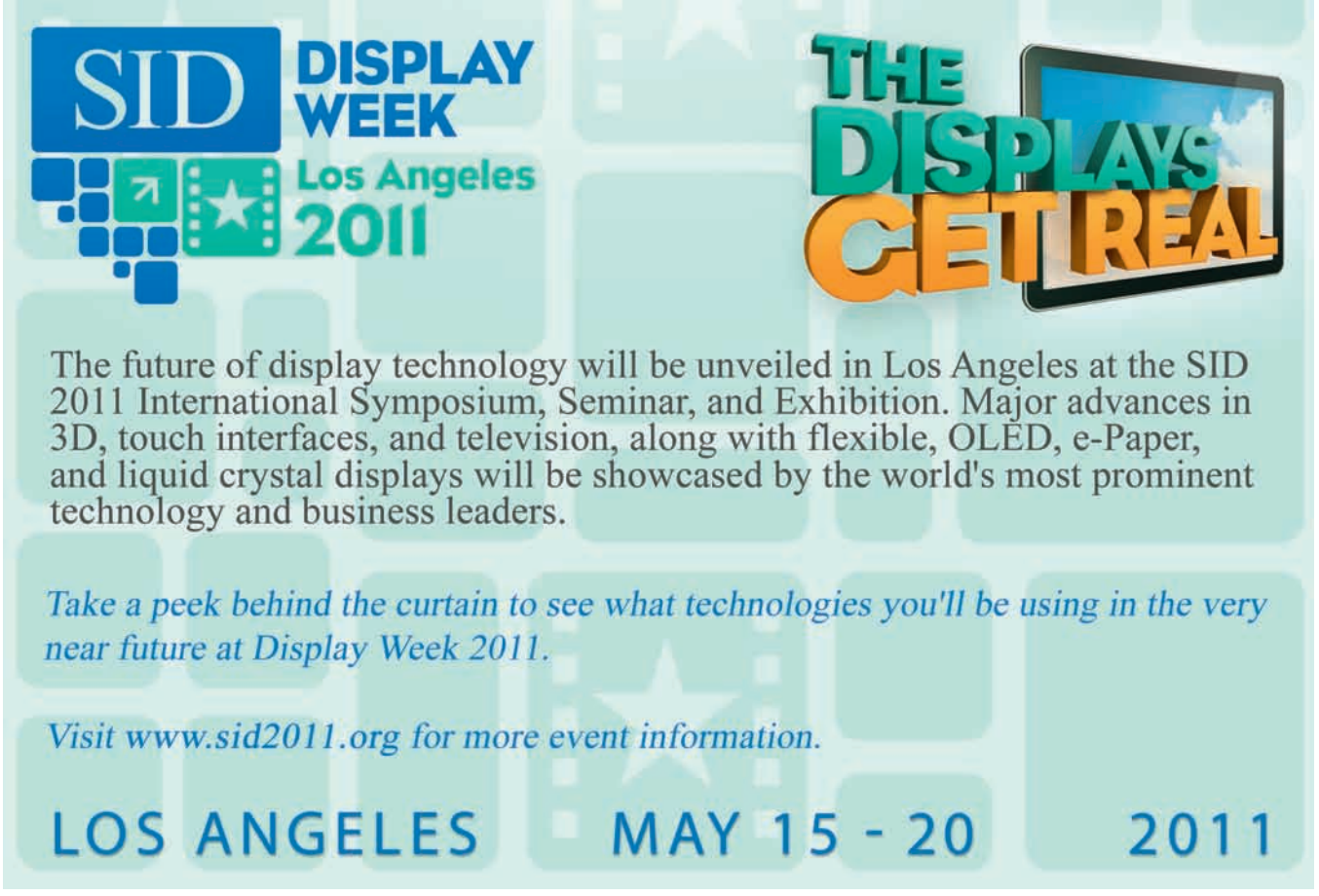
J. Hamblen, "Introduction to WinTIM," The Georgia Institute of Technology, 1999, http://users.ece.gatech.edu/~hamblen/book/wintim.

Xilinx, "Data2MEM User Guide," UG658, 2009, www.xilinx.com/support/documentation/sw_manuals/xilinx11/data2mem.pdf.

SOURCES

TIM Meta-assembler
Texas Instruments, Inc. | www.ti.com

ISE Software and Spartan-3 Starter Kit
Xilinx, Inc. | www.xilinx.com



Store Closing?
Don't search for Circuit Cellar, let it come to you!



It's not about where you read, it's about **WHAT** you're reading.

Innovative Project Designs!
Superior solutions!
Hot products!

GET IT NOW, DELIVERED STRAIGHT TO YOU.

12¢/day

Use the enclosed subscription card or sign-up at CC-Access.com today!

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

by Jaime Garnica (USA)

Testing Platform

Construct a Custom Scope

The Scopey II is a handy oscilloscope and spectrum analyzer. In Oscilloscope mode, it acts as a basic oscilloscope with a sampling rate up to 750 ksps. In Spectrum Analyzer mode, a 256-point FFT is performed to obtain the spectrum of an input signal.

A while back, I built a compact oscilloscope. I was inspired both by a *Circuit Cellar* design challenge and by the demise of my old analog Tektronix oscilloscope. The finished design, built with readily available parts, has a limited bandwidth, but it also has some interesting features that you'd have a hard time finding on a surplus oscilloscope from the 1970s. I call my design the "Scopey II."

Every electronics engineer needs an oscilloscope. There's ample evidence for this. Just look at the numerous amateur oscilloscopes on the market. They typically have sampling rates in the range of 1 to 10 mega-samples per second (MSPS) and have either a small monochrome graphics LCD or use a USB connection and a PC for the display. I intended for my design to be comparable to such scopes, with the distinct advantage of having all the code and hardware designs available for improvement or modification. I also wanted the hardware to be simple so I could later integrate it into other projects where an embedded oscilloscope or spectrum analyzer-type function is useful. I implemented as many functions as were practical in software, and much of what was left is under the control of a Microchip Technology dsPIC30F4011 microcontroller, with only a few leftover bits requiring mechanical switches. All of the parts for the oscilloscope cost me about $50 (see Photo 1).

THE SCOPEY II DESIGN

The design is fairly straightforward (see Figure 1). A typical BNC connector leads to a voltage divider followed by a programmable gain amplifier (PGA). The PGA's output is sampled by the microcontroller's internal ADC and then processed and displayed on a monochrome 128 × 64 pixel display. The various features are controlled by several potentiometers, in conjunction with ADC input pins, as an inexpensive alternative to rotary encoders or switches. The oscilloscope's status is displayed onscreen along with other useful bits of information like the waveform's RMS value and the frequency bin of the largest peak of the FFT output.

HARDWARE & CIRCUITRY

The input stage is shown in Figure 2. A BNC connector is attached via JP1. The signal under examination can be DC-coupled to the input or AC-coupled through a capacitor using a toggle switch, connected via JP9, on the front panel. It is then attenuated by a resistor divider with an approximately 1:8 ratio. A Texas Instruments TLV2374 quad op-amp (IC2A-D) is used to buffer and shift the level of the signal. The first op-amp, IC2A, provides a high-input impedance buffer to the input divider. IC2C provides a low-impedance reference voltage of half the supply voltage (nominally 5 V). IC2A uses the output of IC2C to shift the input signal up 2.5 V so the input to the PGA and then to the ADC is centered in the 0- to 5-V input range of both the PGA and ADC. I use a Microchip Technology MCP6S21 PGA (IC1) to adjust the gain of the input stage. The PGA is controlled by a three-wire SPI, and has eight

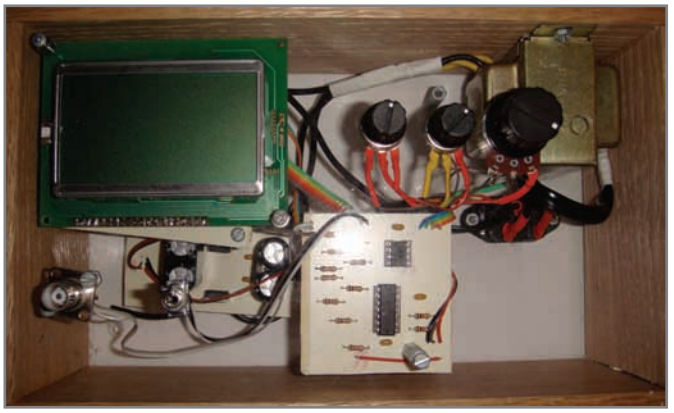


Photo 1—A fully functioning Scopey II oscilloscope and spectrum analyzer

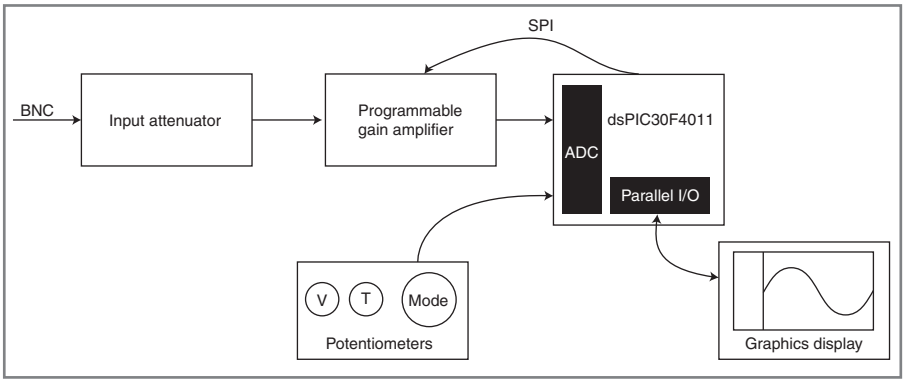


Figure 1—The design features a PGA, microcontroller, potentiometers, and display.

possible gain settings, six of which I used that are powers of two. You can set the total input gain to 1/8, 1/4, 1/2, 1, 2, or 4. On the 1/8 input gain, the maximum input that will not clip is close to 40 V_{pp}. When set at the highest gain setting, the vertical resolution of 64 pixels results in about 20 mV/pixel sensitivity.

Figure 3 depicts the dsPIC30F4011-based circuitry. The PGA's output is fed into an analog input pin (AN3 through JP1) and sampled at a few user-selectable rates (up to 750 ksps). The user can select the sampling rate, as well as the trigger level and slope, using the potentiometers on the front panel. Each potentiometer has its wiper connected to an analog pin, and the other terminals connected to power and ground through JP5. A whopping 14 digital I/O pins are used to control the display, a readily available 128 × 64 monochrome display based on the KS0107/0108 driver/controller combination. The power supply consists of a transformer, a few diodes, and 7805 and 7905 voltage regulators arranged as usual. The original version of this design was built on a pad-per-hole protoboard; however, I rearranged some connections for the current version so it could be easily built on a single-sided board produced using the toner transfer method.

An antialiasing filter on the input is noticeably absent from this design. You can easily add one if necessary, but I left it off so you can view firsthand the effects of undersampling (see Photo 2).

SOFTWARE

I wrote the software entirely in Assembly. Microchip has a nice C compiler for the dsPIC microcontrollers, but I wanted the challenge (and fun) of working with pure Assembly. When the scope starts up, it reads its potentiometers. Based on their positions, it moves to either the Oscilloscope mode or FFT/Spectrum Analyzer mode. In either mode, the software retrieves the desired sampling rate, sets up the ADC for that rate, collects samples until the desired trigger is seen, and then begins storing samples until the 512 sample buffer is full. In Oscilloscope mode, the samples are then scaled appropriately based on the gain settings and displayed. In FFT mode, the samples are scaled, arranged in memory, and transformed to their frequency domain counterparts. The magnitude is then found, scaled, and displayed in a suitable format.

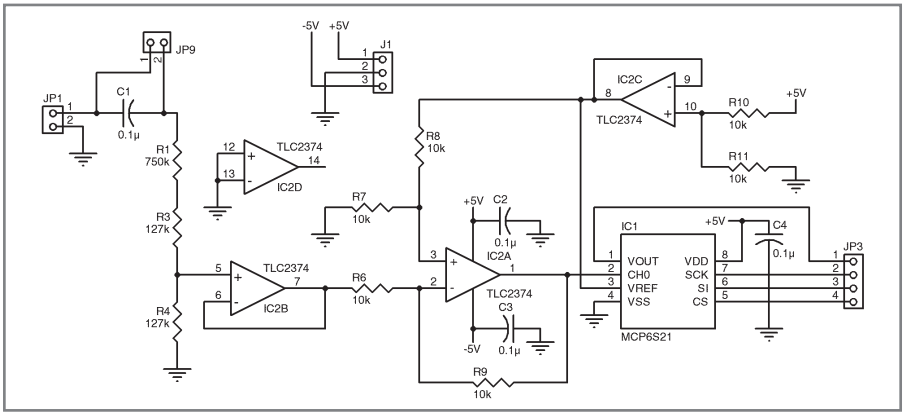


Figure 2—Input stage of the Scopey II. A PGA is used to adjust the overall gain.

The heart of the functionality of this device is the dsPIC30F4011's built-in ADC. It has a maximum sampling rate of 750 ksps. The microcontroller's ADC buffer is a great feature. Instead of interrupting every time a sample is ready, the ADC can go about its business for 16 samples before issuing an interrupt. At an execution speed of about 30 MIPS, you will have enough time for 39 instructions to read the first sample before it is overwritten—a virtual eternity. The functions for setting up and running the scope's functionality are in the oscope.s file, which is posted on the *Circuit Cellar* FTP site. The function `oscSample` is used by both the oscilloscope and the FFT functions. It calls `oscGetVdiv` and `oscGetTdiv`, which read the potentiometers, set the gain of the PGA, and configure the ADC. `oscGetTdiv` configures the ADC using several different functions based on the speed of conversion requested. Several registers must be configured correctly for the ADC to function properly. All of the registers for the ADC are explained in section 17 of the dsPIC30F family reference manual. The first thing to remember is to configure the pin you use as the analog input as an analog pin in the ADPCFG register. The dsPIC30F4011's datasheet indicates how to configure the ADC for proper operation at various sampling rates, as well as how external reference pins and analog inputs must be connected internally to the sample-and-hold (S/H) amplifiers. The ADCSSL register enables you to select which analog input pins will be scanned during conversion. In my case, only one is scanned, AN3. The ADCHS register determines which analog pins are connected to each of the two S/H amplifiers. Both S/H amplifiers are connected to the

same analog input, AN3, because the datasheet specifies the maximum sampling rate at 1 MSPS. To achieve this, both S/H amplifiers must be connected to one input. The errata indicates the maximum sampling rate is 750 ksps, which does not require both S/H amplifiers; but in this case, having both connected has no effect on the operation.

For the oscilloscope's functionality, I want the ADC's output to simply be an integer. But for the FFT, I need the data in signed fractional format so it can function properly with the fixed-point math routines from Microchip. This is accounted for at the end of `oscGetTdiv` by setting the FORM bits in `ADCON1`. This is especially important because using the wrong format for the math routines results in garbage output.

I used two different techniques to achieve the desired sampling rates. For 750 ksps, the ADC alone controls the sampling rate. As soon as one conversion is complete, the next starts and the length of the conversion determines the sampling rate. At all slower speeds, each conversion is done in the same amount of time as at 750 ksps, but the start time of each conversion is spaced out by Timer3 to achieve the desired sampling rate. This is set in the SSRC bits of the `ADCON1` register, where various sources can be programmed as the signal to end sampling and begin conversion. I used the ADC interrupt to implement triggering as well as the retrieval of samples from the ADC buffers. The SMPI bits in the `ADCON2` register enable you to set the number of samples the ADC collects before an interrupt. In this design, the ADC is set to wait for the entire 16-word buffer to be full before issuing an interrupt. During the interrupt, if a trigger has already occurred, then the 16 samples are stored. If a trigger has not occurred, the samples are compared to the trigger value (level and slope) until the trigger is found.

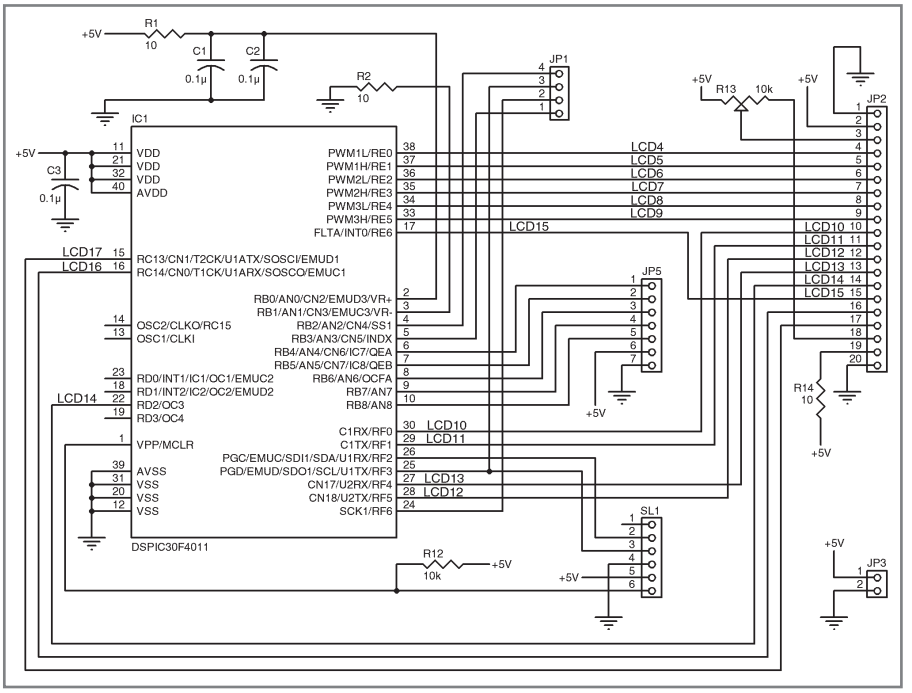


Figure 3—The dsPIC30F4011 connects to the input stage via JP1.

With few exceptions, the software proceeds in a more or less predictable fashion. One interesting aspect of the software is the display driver. As you may know, the KS0107/0108 combo has some idiosyncrasies that make writing software for it a little strange. First, the set will only drive a 64 × 64 screen, so that the entire 128 × 64 screen is accessed as two halves. Next, pixels are arranged in vertical lines of eight, arranged in rows, with eight rows on the screen. For an oscilloscope, the software is essentially mapping a sample from the ADC to one pixel on the screen. Some issues arise. In order to change one pixel, the stack of eight that it is a part of must be read, the individual pixel must be changed, and the entire block must be written back. The heart of the display software is, of course, the function that turns one pixel on or off given just its coordinates. A much simpler option is to use a display with a different controller, but that would be much less fun. Also, displays based on the KS0107/0108 cost virtually nothing on surplus sites, while better controllers—with built-in character generators, serial instead of parallel interfaces, and, gasp, direct access to individual pixels—are quite a bit more expensive. Ah yes, the KS0107/0108 does not have a character generator. So, if you want text, you'll have to code your own font and display functions or borrow them from someone else. I wrote a set of functions for a full complement of character display-like 5 × 8 characters. There are also several functions for drawing or erasing lines used to create eye-pleasing borders around the various data areas on the screen. Also included in the software, but not used in this project, are functions to display 32 × 32 pixel icons.

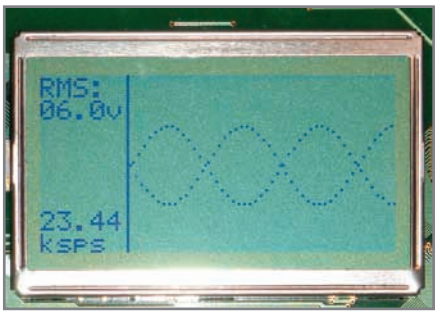


Photo 2—An undersampled sine wave

The most interesting bit of software, by far, is the software behind the FFT function. I would love to say I wrote the actual code that performs the FFT myself, but I don't have to because Microchip is kind enough to provide some useful DSP-type math-intensive functions, optimized specifically for the dsPICs. You have to do a little prep work in order to get things arranged nicely so that Microchip's software will give you the results we seek.

A subtle but interesting problem with using the FFT to obtain the spectrum of a signal is called spectral leakage. In this design, the input to the FFT function is 256 discrete points. You can

NOW, THERE'S EVEN MORE TO DISCOVER.

NEW!



+



+



The upgraded Elektor PLUS subscription!

- All 11 issues including the Summer Circuits edition
- Included in your PLUS subscription: Annual DVD 2011
- **50%** cheaper than normal retail price
- Up to **40% discount** on selected Elektor products
- Elektor is delivered to your doorstep every month
- Read your copy before everyone else
- **NEW:** On your personalized Elektor PLUS website, you have permanent access to the three latest editions of the magazine in PDF format, as well as to a fast Elektor search engine!

NEW:
exclusive access to www.elektor-plus-usa.com!

Your Elektor PLUS subscription gives you **exclusive access** to the new website **www.elektor-plus-usa.com** where the three latest editions of Elektor magazine are available in PDF files (i.e. the current issue and the two preceding ones). With a simple click you download the complete edition (front to back!) or any single article. **www.elektor-plus-usa.com** also supplies the most extensive Elektor search engine found on the web (all issues since 1998).

Subscribe now
www.elektor.com/usa • Phone 860-875-2199

elektor

then identify a bunch of frequencies that fit an entire number of periods into your samples. These frequencies are $f_K = kf_s/N$, where $0 < k < N/2$. If we use one or some combination of these frequencies, everything will look almost exactly like we expect. Sampling a 125-kHz sine wave at 500 ksps will result in all the power of the signal appearing in the correct frequency bin, and be displayed as one line, one pixel wide, on the screen. Now, if we have a signal, perhaps a 124-kHz sine wave, there is a problem. The next closest frequency bin is about 123 kHz. The math doesn't really care what we are trying to accomplish, so it will still give the "correct" answer even if it is not what we really want. Intuitively, one might guess something strange will happen, and that strange thing is spectral leakage. Examining the discrete Fourier transform (DFT), the transform that the FFT implements, the problem is evident. Here is the DFT of a sinusoid of frequency ω, where $\omega_k = 2\pi f_k$:

$$X(\omega_K) = \sum_{n=0}^{N-1} e^{j\omega nT} e^{-j\omega_K nT} = e^{\frac{j(\omega-\omega_K)(N-1)T}{2}} \frac{\sin\left[\frac{(\omega-\omega_K)NT}{2}\right]}{\sin\left[\frac{(\omega-\omega_K)T}{2}\right]}$$

In this simple case of one sinusoidal input at a frequency ω, we see that if ω is one of the f_k frequencies, then the sum is N for that frequency and zero for all others. All the power will be at that frequency, as you would expect. For any other frequency (i.e., not f_k), the sum will be nonzero for all frequencies. In other words, the transform will indicate that your signal has power at frequencies that your otherwise trusty signal generator might disagree with. Another way to think of the problem is that only those certain frequencies f_k will fit nicely into your samples. If you are dealing with a periodic signal, repeating your one chunk of samples will look pretty much the same as the actual signal. If there are other frequencies, though, they won't "line up" correctly, and if you repeat the chunk of samples there will be discontinuities that do not reflect what the actual signal looks like. This second way of looking at the problem suggests one solution: modify the input so that the discontinuities are eliminated. The samples you get directly from the ADC have, mathematically, been passed through a rectangular window. They are the product of the original signal and a function that is one while the samples are collected and zero elsewhere. Using a lot of math, one can explore the full implications of that, but it is sufficient to say that this window shape is not particularly conducive to what we are trying to accomplish. The appearance of the spectrum can be improved, for your purposes and others, by using a different window function. There are plenty of window functions out there, but I chose to use a Hann window, whose shape is given by:

$$w(n) = 0.5\left(1 + \cos\left(\frac{2\pi n}{(N-1)}\right)\right)$$

where N is the number of samples to which the window will be applied, which is 256 in this case, and n is an integer where $0 \le n \le n - 1$. Figure 4 shows a comparison between the shape of a rectangular window and a Hann window. These values were precomputed and stored in a table. To apply the window, the sampled values are multiplied by their corresponding window values from the table—which range from 0 to 1—before being sent to the FFT function.

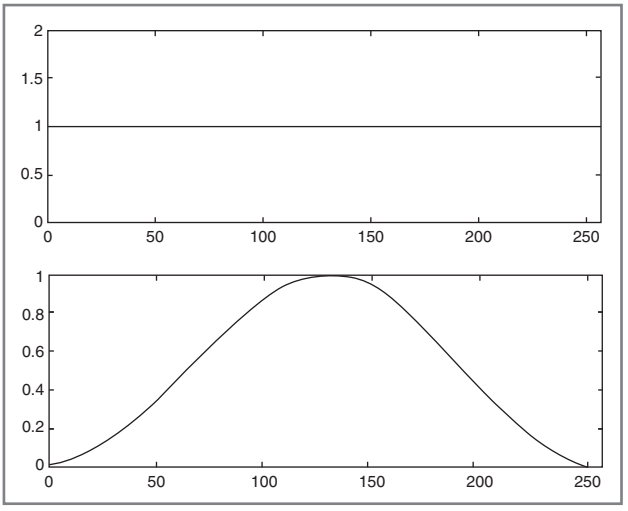


Figure 4—A comparison of a rectangular window and a Hann window

The samples also need to be rearranged a bit before being transformed, as the inputs and outputs of the FFT are complex numbers. Since the sampled signal is real, placeholders with zero value need to be inserted between the real values so that the complex values have a place to be once the data is transformed. The FFT is done in place, and once it's done, you can take the magnitude and then take the log of the magnitude to reflect the relative powers of each frequency.

The interface to the FFT functions provided by Microchip is fairly straightforward when using Assembly. Assuming the ADC is configured to produce values in the correct format, the first step is to insert the zero imaginary part for each sample. Before the execution of `oscFFTarrangeStart` in oscope.s, `w1` is a pointer to the last space in the buffer, and `w0` is a pointer to the last sample value, which is located in the same buffer to which `w1` points. `w3` holds a pointer to the table of values for the Hann window. At `oscFFTarrangeStart`, the imaginary part of each value is set to zero. Then the sample is multiplied by the value from the Hann table and stored, starting from the last addresses in the buffer so that the values stored near the beginning are not overwritten. At `oscFFTarrange` you actually call the Microchip DSP functions. `_FFTComplexIP` does a complex, in-place FFT. `w0` holds the base 2 log of the length of the FFT. `w1` is a pointer to the data to be transformed. `w3` and `w2` point to the coefficients needed to perform the FFT. After the FFT, the data is in the wrong order. `_BitReverseComplex` is provided to rearrange them. The values are complex, so Microchip kindly provides `_SquareMagnitudeComplex` so you can obtain the magnitude. Given the limited screen real estate and the symmetry of the output of an FFT, I only consider the results for positive frequencies, giving 128 frequency bins, 96 of which the display is configured

Join 1,560 Exhibitors & 140,000 Visitors!

Japan IT Week 2011 Spring

Japan's Largest IT Show

Exhibitions:

- 20th Software Development Expo (SODEC)
- 16th Data Warehouse & CRM Expo (D&C)
- 14th Embedded Systems Expo (ESEC)
- 13th Data Storage Expo (DSE)
- 8th Information Security Expo (IST Spring)
- 6th RFID Solutions Expo (RIDEX)
- 5th Direct Marketing Expo (DME)
- 5th Web & Mobile Marketing Expo (Web-Mo Spring)
- 3rd Green IT Expo Japan (GRIX)
- 2nd Cloud Computing Expo Japan (Cloud Japan Spring)
- 1st Smartphone & Mobile Expo (Smartphone Japan)

Dates: May 11(Wed) – 13(Fri), 2011
Venue: Tokyo Big Sight, Japan
Organised by: Reed Exhibitions Japan Ltd.



Show Management, Reed Exhibitions Japan Ltd.
18F Shinjuku-Nomura Bldg., 1-26-2 Nishishinjuku, Shinjuku-ku,Tokyo 163-0570, Japan
TEL: +81-3-3349-8504 FAX: +81-3-3349-8500
E-mail: sodec@reedexpo.co.jp

www.japan-it.jp

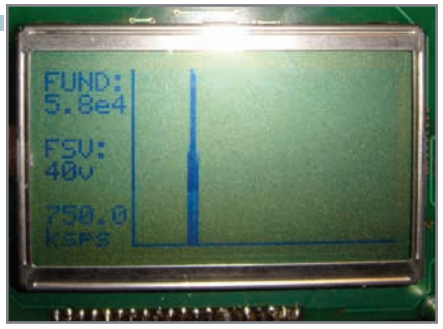


Photo 3—The spectrum of a 58-kHz sine wave

to show, as 32 pixels on the left are reserved for displaying settings and the frequency of the largest spectrum component. The previous FFT results are stored at `oldWave` and the new results are stored at `newWave`. Each point is scaled so that the largest peak goes to the top of the screen. The values of the spectrum are not individually useful at this point; they are mostly useful for the comparative magnitudes of the various frequency components. At `oscFFTlogish`, the values are converted using the table logTab to base-10 log values, scaled so that a full-scale value places its pixel at the top of the screen (0 being the top of the screen and 63 being the bottom). Photo 3 shows a 58-kHz sine wave's spectrum. Photo 4 shows the spectrum of a 1-kHz square wave shown in Photo 5.

PROTOTYPE

I constructed the original prototype on a pad-per-hole prototyping board using a combination of point-to-point wiring and wire-wrapping. I made a later version with boards using the toner transfer method for resist on single-sided PCB material. The process involves using a laser printer to print the mirrored artwork for the board on either commercially available transfer paper or one of several alternative materials, one of which is glossy photo paper. The toner is then transferred to the board using heat and pressure, typically from a laminator or clothing iron. This method is a relatively fast and affordable alternative to photosensitive resist, which requires a UV light source, developer solution, and the photoresist or presensitized boards. The component selection is not critical. This design is not intended to be a precision instrument. The prototype was built using 5% resistors and gives satisfactory results.

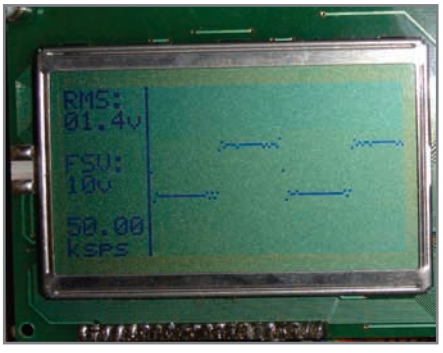


Photo 5—The square wave used for the spectrum in Photo 4

TESTING AT THE BENCH

I developed the Scopey II oscilloscope in order to indulge my own interests in test equipment and to add a useful tool to my workbench. I succeeded. The product of my toiling is a three-IC device that can be easily constructed without much financial strain and would be useful for analyzing signals up to a few hundred kilohertz.

Now you can build your own. Upon completion, you will have a device for probing circuits and fiddling with knobs. (And when you tire of this, you can give it to a child, grandchild, niece, nephew, or even an inquisitive sibling who still has that spark of amazement about the world around them.) Of course, you'll also have a compact platform around which you can build a host of intelligent monitoring projects.

There is something Zen about test equipment. I get a lot of satisfaction when I build something that is useful. There is an additional philosophical satisfaction when I build something that I can reuse for future projects. And it satisfies a deeper, uniquely human, urge to find ways to see what we cannot see. From telescopes and microscopes to gas chromatographs and particle accelerators, we are always seeking ways to explore our reality. All the while, sitting quietly at our benches is the humble oscilloscope, helping us accomplish that very task. Often taken for granted, this remarkable companion gives us the corrective lenses we need to see into the world of electronics.

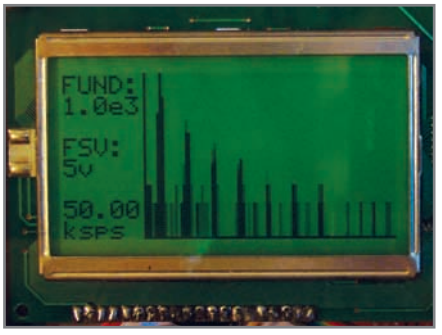


Photo 4—The spectrum of a 1-kHz square wave. The frequency of the square wave is the highest peak. You can see the odd harmonics that compose the square wave.

Jaime Garnica (jbgarnica@gmail.com) earned a B.S. in Electrical Engineering in 2007. He is currently pursuing his Ph.D. In addition to playing around with microcontrollers, Jaime enjoys the occasional Cocoa programming adventure.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/250.

RESOURCES

Microchip Technology, "dsPIC30F Family Reference Manual," DS80169E, 2004.

J. Smith, *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications Second Edition*, Stanford University, 2010.

SOURCES

dsPIC30F4011 Microcontroller and MCP6S21 PGA
Microchip Technology, Inc. | www.microchip.com

LCD-00710 Framed graphical LCD
Sparkfun Electronics | www.sparkfun.com

TLV2374 Precision amplifier
Texas Instruments, Inc. | www.ti.com

sensors expo & conference

Pre-Conference: June 6, 2011
Conference & Expo: June 7-8, 2011
Rosemont, IL • Donald E. Stephens Convention Center

25+ YEARS OF EXCELLENCE

www.sensorsmag.com/sensorsexpo





The Only Industry Event in North America Exclusively Focused on Sensors and Sensor Integrated Systems! Find the Solutions to Your Sensors and Sensing Technology Challenges!

Gain the knowledge you need from leading experts and peers in the sensors industry.

This year's Conference Program includes more Technical Sessions in Tracks covering the topics that are most important to you and your business, including:

- MEMS
- Energy Harvesting
- Wireless Sensor Networking
- Novel Approaches to Measurement & Detection
- Wireless Sensor Data
- Mobile Sensing – NEW for 2011!
- Cutting-Edge Technologies – NEW for 2011!

Identify specific solutions to your most difficult sensing, detection and control-related challenges on the expo floor.

Sensors Expo brings together the largest and best-in-class showcase of sensing technologies and systems for attendees to evaluate and make informed decisions.



Tuesday, June 7
9:00 AM – 10:00 AM

Opening Keynote:
Dr. Hugh Kerr
Biomechatronics Researcher

Don't Miss these Expo Floor Highlights:

CO-LOCATED WITH:
Learn today. Design tomorrow.
ESC

MEMS PAVILION

ENERGY HARVESTING PAVILION

TECHNOLOGY UPDATE THEATER

SUBSCRIBERS:
Visit www.sensorsmag.com/sensorsexpo to register or call 866-817-5048. Use discount code A305E for an EXTRA $50 OFF a Gold or Main Conference Pass!

Scan this QR code to Register Now on Your Mobile Device!
Get the free mobile app at http://gettag.mobi

MEDIA SPONSOR: CIRCUIT CELLAR

PRODUCED BY: QUESTEX MEDIA

OFFICIAL PUBLICATION: sensors

by George Martin (USA)



Design Development (Part 1)

Planning and Tools

You can read and write C code. The next step is to start building something. Follow these tips as you prepare to develop your next product.

I covered C programming in many of my previous articles. If you followed along, you should now have a sound foundation in the language. The C programming language was developed to design control systems for AT&T. With all of the modern progress in system design, you can apply the language tools to your advantage. It's time to design a project.

A DOSE OF C

You should be able to read and understand C code. Go slowly. Just break the code down to the basic language operations. You can look up answers to your language questions on the 'Net. And because C is so universal and popular, you can ask your associates. Also look at the flowcharts and code presented in *Circuit Cellar*. It's become a strong point in all of the articles.

Writing C code is a bit more demanding than simply reading it. You will find that the more code you write the easier it becomes. Writing C code is comparable to learning a foreign language: it just takes practice and time. Also, every time I get to look at C code that someone else has written, I learn. Perhaps it's not in a style that I want to adopt, but it's still informative to see that code.

The C programming language is structured. It comes in a basic set of rules and keywords. By this, I mean you don't get graphics or keyboard routines as part of the language (as perhaps with Visual BASIC). However, the language and compilers let you easily add graphics and keyboard libraries to your code. This is one of C language's great features. You can include external libraries and even publish your own libraries. I also believe that this trim language lets you focus on the design of the system, and you are not held back by trying to fit your design into a complicated language.

As we go about the design process, I'll introduce other languages (Java and BASIC) and perhaps an RTOS or two to get the job done. Don't worry, your foundation in C will serve you well.

THE PRODUCT

It would be great to say we are going to design the next blockbuster product. But let's start with a simple plan and get comfortable with the product design process. Hopefully, you will soon understand how this all unfolds. Watch out for design requirements that describe the "perpetual motion machine." They are difficult to design. Keep your plans realistic.

All sorts of clever projects are available on the 'Net. (I've seen some on hackaday.com.) Recently, I read about a CNC-controlled cake decorator (good idea) and a flaming fireball controller for Guitar Hero. (I'm not so sure about that one.) A while back Steve Ciarcia sent me a link to a product that plugged into a cigarette lighter in the car. It listened to the interference generated by the engine, determined the engine speed, and emitted a sound like a Ferrari or monster truck engine at that same speed. (Great idea.) All are interesting projects. It would be interesting to know if any of them are making any money!

THE PLAN

I am designing a spring tester for my personal use. It probably isn't fair to make you read about that design, particularly if you have little or no interest in measuring springs. I am actually describing the design process you can use for your particular requirements. You should be able to modify components from this description and

add your own specific modules. And, by "modules," I am referring to both the software and hardware modules. I'll also take suggestions for various interfaces and write about how I'd implement those designs. So, if you would like to learn how to interface to an LCD or an SD card for file I/O, let me know.

I think we can best categorize this design as an instrument. It's not a toy or a game. That's not my background, although I once did a chorus of synchronized talking Big Birds for the New York Toy Show. I will say some toys are getting rather sophisticated.

Figure 1 is a block diagram of the system. If you've been following my column, you recognize that I'm using (or misusing) the Unified Modeling Language (UML). It's a good program to capture your thoughts. And a picture is certainly worth 1,024 words.

Let's take a look at what Figure 1 is all about. This diagram looks rather generic. Not much value here, you say. Well, that's why I'm the expert. Actually, most of my systems look like this and the devil is, of course, in the details. I find that putting thoughts down on paper is always a good idea.

Figure 1 shows a user who can see the display, enter key strokes on the keyboard, and have some effect on the input signals. For the spring tester, this might be moving a lever that compresses the spring. Some of the other input signals the user cannot directly affect. An example might be the room temperature. There's a CPU, and I added local storage and connections to a PC or the 'Net.

The power supply (PS) touches each module. (I omitted those connections for now.) The PS could be a battery, a plug into the wall, a plug into a USB connector, or power over the Ethernet.

One of the blocks is named Interface to PC. I'll probably cover the PC in a dedicated column. There is just too much to cover. But, before we come to that point, let's get further along with the design.

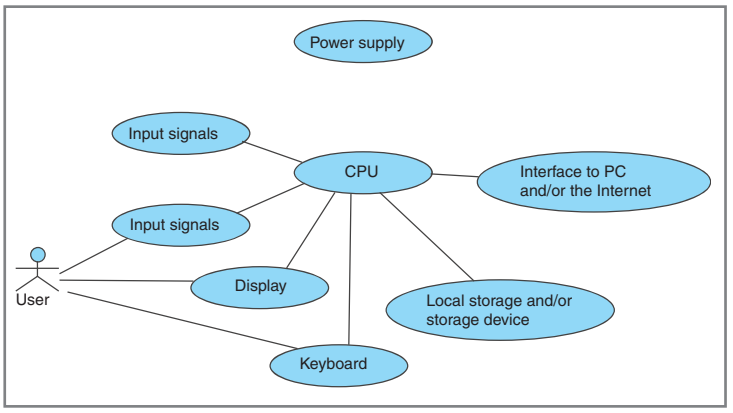


Figure 1—System block diagram

For the CPU, I suggest a new one such as Texas Instruments's Stellaris, Freescale's Kinetis, or Renesas's RX. These are high-power units with all of the peripherals that I've included in the block diagram and more. They all have C compilers and cost less than $20 in low quantity and closer to $5 in volume. That makes the selection of the CPU easier, and I don't think you can go wrong with these or others of this class of CPU. They all also have evaluation boards that we can use to build our product.

THE TOOLS

What tools are necessary for this design project? Well, you need a PC to run the compiler and test and debug the code. I started the design with an evaluation board from the CPU manufacturer. That's the easiest part on the toolset. Most of the products used in design challenges *Circuit Cellar* has run in past years would be suitable to use in this spring tester. You also need force gauges and calibration weights to calibrate this system.

I envisioned this product to consist of a stepper motor and a lead screw to compress the spring and give a displacement value, a load cell to measure the force on the spring, and a fixture to hold everything in place. Some mechanical design and fabrication are necessary to complete this project. So, you can build everything with a table saw, a router, and a drill press, or you'll have to contract with a machine shop for fabrication.

You might need to prototype some interface electronics on a separate printed circuit board. This is especially true for stepper motor drive and signal conversion with the strain gauge. It's likely that the CPU demo boards do not have this level of interface.

TASKS & SCHEDULE

Before you can make a schedule, you need to identify what needs to be done. At this point, you might label this activity as brainstorming; but after you get a few designs under your belt, you will see that you are actually laying down some valuable groundwork.

What would a customer like to have in this product? They likely want methods to measure the characteristics of a spring, compare the measurements of a spring to a standard, calibrate the device, save the results of a measurement, recall the results of a measurement, print the results of a measurement, perform a go/no-go type of test, gather statistical manufacturing data, update software, work with different spring rates, and measure leaf-type spring.

This is only a list of functional requirements. You also need to consider price, size, weight, accuracy, and much more. If you are going to make a product to sell, you'll need to know how you're going to sell that product. Are you selling online or through a distributor? Price, warranty, repair policy, and advertising campaigns are important factors to consider. That is a lot more than we'll ever be able to cover in bimonthly columns in *Circuit Cellar.* We'll leave the sales side of the product design out of the discussion. I would caution you to develop a parts and manufacturing budget very early on if you're going to develop a product to sell.

At this point, you should be able to generate a datasheet. You may want to wait until a prototype is built, but I suggest not. The sooner you get facts on paper the sooner you will be able to spot problems and opportunities.

With that datasheet you can compare it to other existing products and you can give it to prospective customers and get their input. Also you can begin to think about how you're going to sell your product.

SYSTEM DESIGN

I used to start coding at this point because I thought that was the fastest way to get to the real design issues. But I believe it's better to use higher-level tools, like UML, to think about the system design. Just exactly what is a spring tester? Look it up on the 'Net and you'll find several examples (mostly for valve springs used in automobile engines). This tester is a device that measures and reports spring displacement versus force. This is called the spring rate. I'm mainly interested in small coil springs. The force will be 10 to 25 lb. and the displacement will be 1″ to 2″.

Start the project by considering only coil springs in compression. The data you need to gather is displacement versus force. These readings come in pairs and they are referenced or compared to the desired spring rate. Ignore absolute accuracy at this point.

Envision loading a spring to be tested into a fixture and then running the stepper motor that moves the lead screw to compress the spring. As the test mechanism takes out all of the slack, the force measured goes from zero (or perhaps a level better described as noise) to some nonzero reading. At this point you've started up the spring-rate curve. Each spring has an installed length and a working length. At the beginning of compression, you won't measure the working spring rate, and at the end of compression you get into what is known as coil bind. This is where the coils are actually touching each other and the force shoots up. Somewhere between these two end points lies our information.

Will your instrument have all of the I/O (keyboard, display, and printer) to display all of the measured data and settings? I don't suggest presenting the data in list form. Viewing hundreds of readings showing displacement and force would just be a waste of time. But what I might want to see is the spring rate between two points. These points would be the resting length and the compressed length. But how would we enter these length numbers? I could see the code might have a database of standard spring part numbers internally, and then we could just scroll through the list to select the spring we are going to test. But then how did I set up those standard springs?

Another output might be a graphics-type display that plots the spring data readings as we are running the test. This would look like a line passing through zero and the slope would be the spring rate. Now this type of display could be useful, but what if I wanted to zoom in or magnify a region of the display? How would I do that?

Well, let's just move all of the data over to a PC. We could then use the PC's display to see the data and the keyboard to input all of the setup variables. This is a good solution, but it requires a PC to be part of the instrument. That's probably fine for me, but could you turn this into a product? Would it be too complicated for a spring maker to use? I think so. How are you going to communicate with the stepper motor and force gauge? Should you use serial port, USB, Ethernet, or move a memory card between the units?

Then you have the saving, recalling, and comparing of all of the data. You could put a file system for a CF card in the system. Then it's easy to move that card between the unit and the PC. Again, this is sufficient for my use, but it's complicated for the spring maker.

As you can see, early design choices can have a great impact on the final design. Since I designed this for my own use, I had the option of going with a microcontroller that runs the stepper motor, reads the force gauge, and has a few simple buttons to open and close the mechanism. I knew that option would get me closer to a working unit, and then the unit to PC communication would be a required design step. This approach would also commit me to save and graph the data on a PC. That's fine, but I would have to be careful that the code is easily ported to a microprocessor-based standalone unit if the product is ever going to be sold as a standalone system.

WHERE ARE WE NOW?

Well, you've had a first pass at the UML diagram of the main sections of the design. Plus, you know the features the product should include. You also spent some time questioning the operation of the unit and the partitioning of the design between a standalone approach and a unit that requires a PC to operate.

At this point it's time to write up a list of parts, a budget, and a task schedule. I encourage you to take these steps.

I plan to start looking for parts to implement the design. I'll also gather data on the CPU reference designs I'm recommending. You should either think about spring testing (and send me your comments), or think about your own product and map out the steps I described. See you next time.

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 information. He is also a nationally ranked revolver shooter.

SOURCES

Freescale Kinetis
Freescale Semiconductor, Inc. | www.freescale.com

Magic Draw
No Magic, Inc. | www.nomagic.com

RX Family
Renesas Electronics Corp. | www.rxmcu.com/USA

TI Stellaris
Texas Instruments, Inc. | www.ti.com

ICbank
Gateway to Electronics

3 PORT INTERFACE
RS-485 to Ethernet Converter



Only
$170



RS-485 to Ethernet Converter

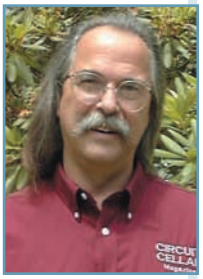
Powerful feature

- Protocol converter RS485 between Ethernet
- Offer TCP/IP Communication to Devices with RS485 I/F

Specification

Network	: TCP, UDP, DHCP, ICMP, IPv4, ARP, IGMP, PPPoE, Ethernet, Auto MDI/MDIX , 10/100 Base-TX Auto negotiation (Full/half Duplex)
Serial	: RS485 3 Ports, 1,200~115,200 bps, Terminal block I/F Type
Control program	: IP Address & port setting, serial condition configuration, Data transmit Monitoring
Accessory	: Power adapter 9V 1500mA, LAN cable
Etc	: - DIP Switch(485 Baud Rate setting) - LED: Power, Network, 485 Port transmission signal

by Jeff Bachiochi (USA)



Error Checking

Data security isn't a perk. It's a requirement. This article covers the topics of error checking, checksums, and the cyclic redundancy check. Error checking matters.

Remember when we used highly condensed polycyclic aromatic hydrocarbon (bitumen) for its sealing properties? Oh, you weren't around B.C.? Maybe you remember using sealing wax to protect your document. Oh, you no longer wear that signet ring? Do you have a PIN? Most people today have more than one. While tampering may be on the minds of many people, it is often the integrity of the data itself that is of most concern. As most data is exchanged via some sort of serial link, today we employ various means of examining its integrity. This might be a byte at a time in hardware, as between UARTs, or by the packet via software. A UART can automatically append an extra parity bit to each byte it transfers, as an indication that the data was received without error. A single bit error (or at least an odd number of bit errors) can produce an error flag.

When sending data within a network, the actual data is often broken into pieces. Each piece is wrapped by the open systems interconnection (OSI) model so that its correct reception can be ensured and the piece can be reassembled with others back into the original data. Each wrapper contains its own technique to ensure packet integrity. The most common approach is to use a checksum or a cyclic redundancy check (CRC).

An application will have little control over packets and how they employ packet integrity. However, the same techniques can be used by your application to protect its data. Checksums can be pretty simple to use. You can think of a UART using parity as the equivalent of a 1-bit checksum. Each of the 8 data bits, either a 1 or a 0, is added together:

Byte = 0b00110001
0 + 0 + 1 + 1 + 0 + 0 + 0 + 1 = 3

This total AND 1 equals the parity of the byte:

3 = 0b11
0b11 AND 0b01 = 0b01

An alternate approach to this is to XOR each bit value with the previous result:

Byte = 0b00110001 (result initialized to 0)
0 XOR result (0) = 0
0 XOR result (0) = 0
1 XOR result (0) = 1
1 XOR result (1) = 0
0 XOR result (0) = 0
0 XOR result (0) = 0
0 XOR result (0) = 0
1 XOR result (0) = 1

The second approach might not seem any simpler, but it has the advantage of not requiring a huge accumulator, which may be required when using the first approach, if the data string is large. Whereas the UART's parity spans a single byte, strings or blocks of data often use byte, word, or larger checksum values. When using a byte checksum, the total of all the bytes in the string (or block) is ANDed with 0xFF (limited) to 8 bits wide. A word checksum can handle more bytes before being limited by its 16-bit width.

While any rollover of a checksum's width decreases its effectiveness in indicating a data

Name	Polynomial	Used In
CRC-16-IBM	x16 + x15 + x2 + 1	Bisync, Modbus, USB, ANSI X3.28, many others; also known as CRC-16 and CRC-16-ANSI
CRC-16-CCITT	x16 + x12 + x5 + 1	X.25, HDLC, XMODEM, Bluetooth, SD, many others; known as CRC-CCITT
CRC-16-DNP	x16 + x13 + x12 + x11 + x10 + x8 + x6 + x5 + x2 + 1	DNP, IEC 870, M-Bus
CRC-16-DECT	x16 + x10 + x8 + x7 + x3 + 1	Cordless telephones

Table 1—Here are a number of popular 16-bit CRC forms

error, it still boils down to the odds. That's "odds" as in, "Are you a betting man?" and not odds and evens. Errors that develop between the transmission and reception are dependent on the transmission medium and the environment. The cause is not the focus of the discussion, as the design and adoption of these transmission systems give us confidence in their low inherent error rates. However, implementing a data integrity method can result in identifying an integrity issue. The simplicity of checksums makes it ideal for basic error checking. Some simple errors can be missed by the checksum method. The same issue with two errors in a parity-checked byte exists with a checksum. If errors occur in the same bits of two different bytes, a checksum error won't be found. For example: 240 + 33 = 273, which has the same checksum as: 241 + 32 = 273.

The checksum's width makes no difference (8 bits, 16 bits, or even 64 bits). A checksum is incapable of recognizing that this kind of error has occurred.

The context of the data can often help point out errors. For instance, it is fairly easy to see a bit error in a string of text. But, if the data is a binary block, there is no context from which you can extrapolate the correction information.

I've got a couple of projects coming up that use a CRC instead of a checksum as a method of error detection. This seems to be a good time to look at how this works.

CYCLIC REDUNDANCY CHECK

A CRC is a more effective means of error detection than the checksum method because each piece of data (nominally a byte) can affect the full data width of the CRC, rather than just its own data width. CRCs can determine a single-bit error, a two-bit error, any odd number bit error, and many burst errors. To accomplish this, the additive method of calculating checksums is replaced with a division method. If we consider the data as the dividend, what determines what the divisor is?

A divisor is dependent on the width of the CRC and the polynomial chosen for its effectiveness based on probable error types. I won't discuss this further here. (If you're a math enthusiast, refer to the titles in the Resources section at the end of this article.) There are multiple standard divisors for 16-bit CRCs. Because we are dealing with base-2 or binary math, the divisors are given as polynomials. The CRC-16 standard uses $2^{16} + 2^{15} + 2^2 + 2^0$ or 0b11000000000000101 (0x18005). The X.25 standard uses $2^{16} + 2^{12} + 2^5 + 2^0$ or 0b10001000000100001 (0x11021). You may have noted that each of these is actually 17 bits in width, but you will see shortly that the MSBit (and it is always a "1") can be discarded, allowing the remaining 16 bits to be held in a word register. For the remainder of this discussion, I'll use the CRC-16 standard as the (polynomial) divisor. First, let's review what happens to the 17th bit.

We will treat the data as one potentially huge binary dividend and divide it by the divisor (0x18005). Instead of using some potentially humongous divide routine, we need to keep things in manageable chunks. This is accomplished by doing some base-2 long division. Let's take a look at dividing the single byte 0x31 by 0x18005. We need to pad the data byte 0x31 (and any data string) with 16 bits of zeros (the width of the CRC we are using).

```
0x31 becomes 0x310000 = 0b00110001000000000000000
0x18005 = 0b11000000000000101

                                  0010001 (Quotient)
1100000000000000101 | 00110001000000000000000 (Dividend)
(divisor)             00000000000000000
                       0110001000000000
                       00000000000000000
                        110001000000000
                        110000000000000101
                         000010000000010100
                         000000000000000000
                          00100000000101000
                          00000000000000000
                           0100000001010000
                           0000000000000000
                            1000000010100000
                            1100000000000101
                              1000000010100101 = CRC value 0x80A5
                              (remainder)
```

Note that the divisor (polynomial) is 17 bits long. (The MSBit is in red.) The MSBit of the dividend determines whether we subtract the polynomial (if it's a "1") or 0. This choice always produces an MSBit of "0" in the result (which reduces the result's width). Bring down the next digit (in green) and the result again gains a bit of width. The subtraction (normally used in long division) is simplified to an XOR function (disregarding carries). After repeating the function eight times (once for each bit of original data), the final result (remainder) is the CRC value for the data. *The quotient is not used!*

So what about that 17th bit? Look back at the previous example. If we start by shifting the data (dividend) left into a 16-bit register 17 times, we end up with the first bit shifting out (into the carry) and the 2nd through 17th bits (in blue) in the 16-bit register. If the carry holds a "1," then the (16-bit) polynomial is XORed with the dividend register. If the carry holds a "0," then a (16-bit) zero is XORed with the dividend register. You'll note that you don't really have to XOR 0, as the result is equal to the

Form	CRC-16	CRC-16	CRC-32	CRC-32
Polynomial	0x8005	0x8005	0x04C11DB7	0x04C11DB7
Width	16-bit	16-bit	32-bit	32-bit
Initial CRC register value	0x0000	0x0000	0xFFFFFFFF	0xFFFFFFFF
Final XOR value	0x0000	0x0000	0xFFFFFFFF	0xFFFFFFFF
Reflect data (byte)	No	Yes	No	Yes
Reflect CRC	No	Yes	No	Yes
Test sequence	"123456789"	"123456789"	"123456789"	"123456789"
Test CRC	0xFEE8	0xBB3D	0xFC891918	0xCBF43926

Table 2—This parameter model for two CRC-16 and two CRC-32 variants shows how drastically they can alter the resulting CRC. The last two parameters are optional, but they include a way of determining that the algorithm produces the desired effect. These include a string of test data (i.e., 0x31, 0x32, 0x33…) and the resultant 16/32-bit CRC.

dividend. At this point, the result is the CRC for one bit of data.

Should there be more data (bits), we prepare to repeat the function by shifting in the next bit of data, which forces out a new value into the carry. By this you can see that in the dividend the 17th bit is held in the carry. The 17th bit of the divisor (polynomial) isn't really needed, it's implied. When the carry is "1," the 17th bit of the polynomial (which is always a "1") will always be eliminated by the XOR. So, we don't need it. This enables the actual polynomial 0x18005 to become 0x8005, a nice 16-bit value.

PARAMETERIZED MODEL

I previously mentioned CRC-16 and X25 as two 16-bit standards using different polynomials. In fact, there are more where these came from. While the polynomial itself is a good start (see Table 1) to providing a CRC routine for encoding and decoding your data, there are often other factors involved. A parameterized model can present these possibilities in a simple list form. This list, illustrated in Table 2, shows the polynomial discussed earlier, along with its potential variants. These include the initial value of the CRC, a value that must be XORed with the final calculated CRC, whether the data (input bytes) are to be initially reflected (LSBit first), and if the final CRC value must be reflected. Optional test data helps to give some level of confidence to your algorithm.

CALCULATING

The algorithm for encoding a CRC is the same for decoding it. The data is fed in and the algorithm produces a CRC. The calculated CRC is appended to the data at the

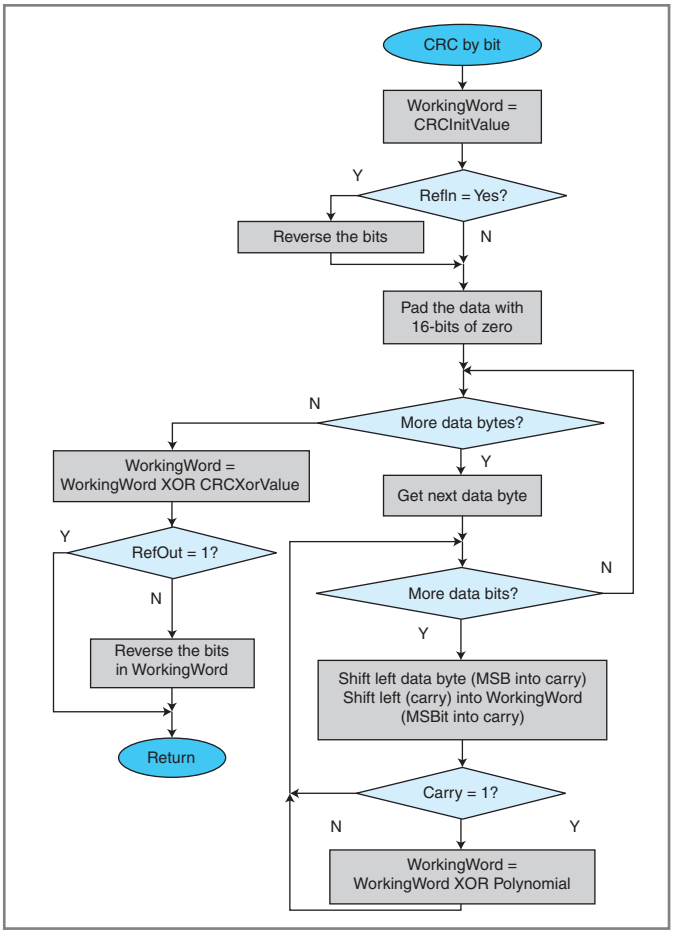


Figure 1—To calculate a CRC value for data, the algorithm must perform some process based on each bit of the data. The resultant CRC value remains a constant predetermined width.

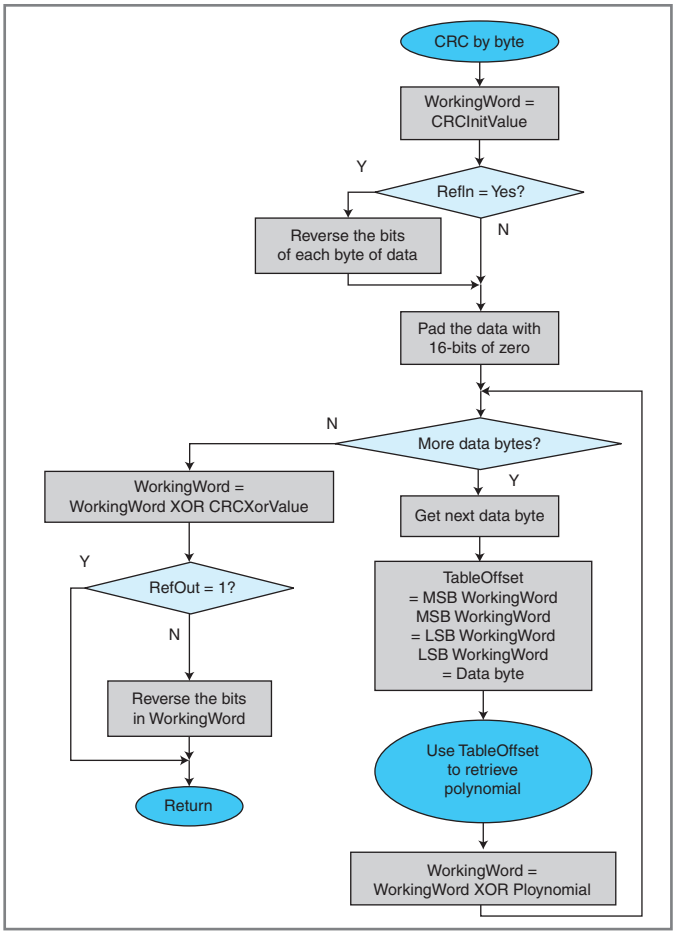


Figure 2—Based on previously calculated CRCs for all byte values, a look-up table can speed up on-the-fly CRC calculations by reducing bit treatment to byte treatment.

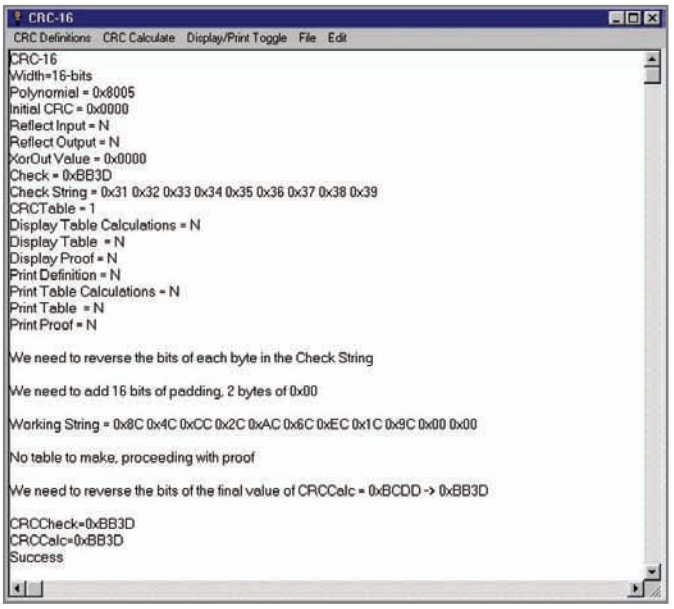


Photo 1—Running my CRC application on my PC displays the present CRC parameters followed by user choices, the used algorithm steps, and finally a comparison of the calculated CRC with the check CRC. The CRCTable variable (1, 4, or 8) determines whether the algorithm calculation used is by the bit, nibble, or byte.

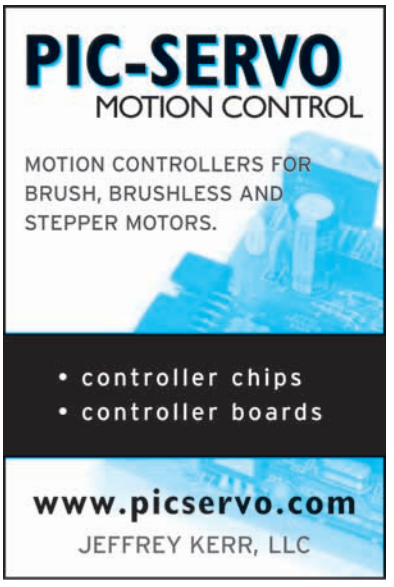
transmission and stripped off at the reception end. If the received data is passed through the same algorithm, then the calculated CRC should match what was appended to the data.

Figure 1 depicts a possible algorithm for finding a 16-bit CRC value for a string of data. Each data byte in the string (or buffer) gets reflected, if necessary, and 16 bits of 0 are appended to the data. There are two loops within the algorithm, an outer loop to retrieve each data byte and an inner loop to shift each bit of the data byte into the WorkingWord register. The bit that pops out of the other end is used to determine whether or not to XOR the polynomial with the WorkingWord. This shifting and XORing is repeated until all the bits of all the bytes have been operated on. The result left in the WorkingWord register becomes the CRC, unless either of the two remaining operations changes it further.

Calculating a CRC for a long string of data can take some time. If you will be doing a lot of CRC work, or it just needs to be done faster, you can consider other options. If you are willing to give up some program space, a CRC table might be the right approach. You will need to decide how much space you are willing to give up for table use. For a 16-bit CRC, tables will range from 16 words for a nibble table to 256 words for a byte table. Anything larger will cost just too much valuable space. Figure 2 shows how the algorithm for a byte table might be used to speed up the CRC calculations. The inner loop in Figure 1 has disappeared and it is replaced by a precalculated table holding the results of all possible byte CRC calculations. A similar algorithm can be used based on nibbles. The nibble table is much smaller, but you will have to do twice as many lookups for the same amount of data.

BUILD ME A TABLE

I like building some quick sample code to make sure I understand the algorithms. Liberty Basic gives me a good playpen for fooling around with code. One benefit to this is the ability to easily create a table for any variation of a 16-bit CRC form. I started this BASIC application to simply try out the algorithm shown in Figure 1. While I started by hard coding in the constants of a specific 16-bit parameter model, once I got correct results with the algorithm, I had to make it more flexible. I added a drop-down menu so each parameter can be changed. The application uses the CRCTestString as default data and pumps it through the algorithm (looping through each bit of each byte of the string), finally comparing the result to the CRCTestCRC. Photo 1 shows a run displaying only the proof results. The proof can be displayed as well (see Photo 2), which shows operations on each bit, broken down into byte-sized chunks.





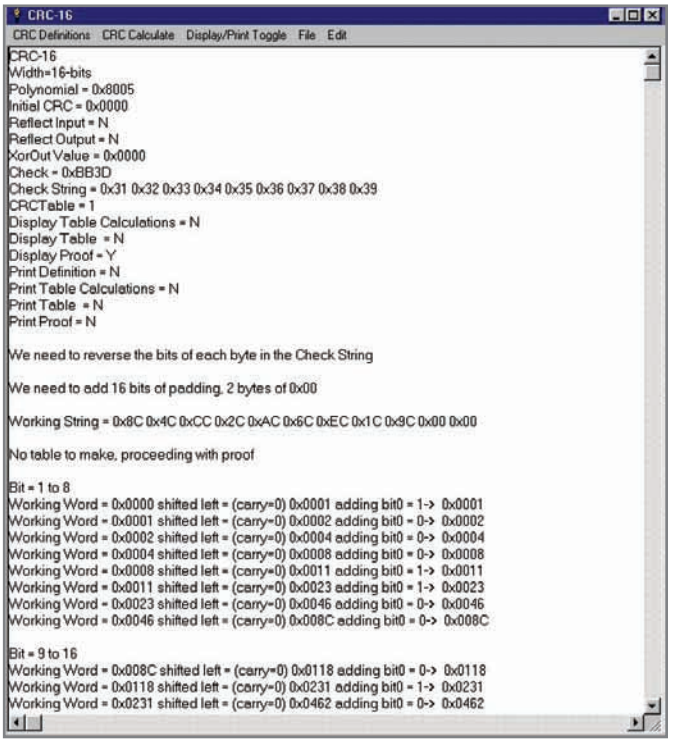


Photo 2—Toggling display/print options allows different amounts of data to be displayed or printed. In this case, DisplayProof = Y. Each bit operation is shown in groups of data bytes.

Once the algorithm had been proven based on the correct CRC value calculated over all the bits of the test string, I added code to use the algorithm to calculate CRC based on all possible single-byte input values. The input byte value is an offset into an array of calculated CRC values for all byte values. Photo 3 is a table displayed in rows of eight CRCs. This byte table can be copied and placed into any application in which you want to implement this method of calculating CRCs.

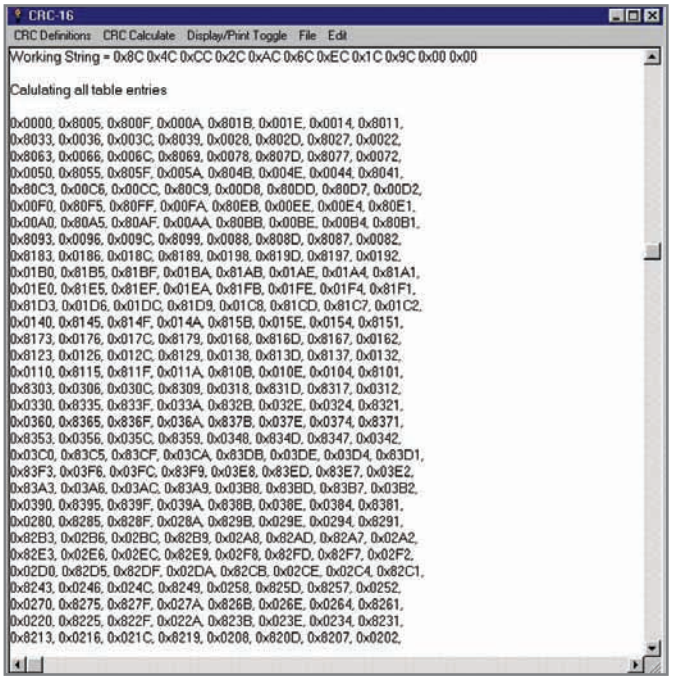


Photo 3—With CRCTable = 8 and DisplayTable = Y, a table of 256 CRCs (for bytes 0x00 to 0xFF) is calculated and displayed.

Note that the first 16 values are also the CRCs for a nibble table. Suppose your data is the single-byte 0xFF. After padding and reversing the data, you'd have 0xFF0000. WorkingWord is initialized with 0xFF00. The fifth nibble is pushed into the WorkingWord, and out pops the nibble 0xF, leaving 0xF000 in the WorkingWord. The table entry at offset 0xF is 0x0022 (see Photo 3). 0xF000 XOR 0x0022 = 0xF022. The sixth nibble is now pushed into the WorkingWord and out pops 0xF, leaving 0x0220. The table entry is again 0x0022. 0x0220 XOR 0x0022 = 0x0202. Why am I showing you this? Well, if you look at the last table entry in the byte table (see Photo 3) you'll see that 0x0202 is indeed the CRC for the byte 0xFF. This shows the relationship between nibble calculations and byte calculations. All of this is based on bit calculations for finding the CRC of a string of data.

APPLICATION DOWNLOAD

You can download a copy of this application from the *Circuit Cellar* FTP site and learn how CRCs can be created. This is a brute-force implementation, so please be kind. I know you "C" guys probably have canned routines for doing CRCs within your applications, and that's fine. I present this for those of you who are interested in knowing more about the inner workings. Some of us are interested in what's inside the black box.

WHO KNOWS?

Methods like the checksum and the CRC provide a level of confidence to data transfer integrity. You need to be confident that what goes into a transmitter is exactly what comes out of the receiver. Biologically, DNA goes through this process during reproduction. To replicate, DNA must unzip into two half strands (like a ladder with breaks through each rung). Each half must be rebuilt as an exact copy of the original or you have an error, chromosome damage. What is the term used to describe the error-checking process that goes on to protect against DNA integrity errors?

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/2011/250.

RESOURCES

A. S. Tanenbaum, *Computer Networks,* Prentice Hall, 1981. *(See Section 3.5.3 on pages 128 to 132.)*

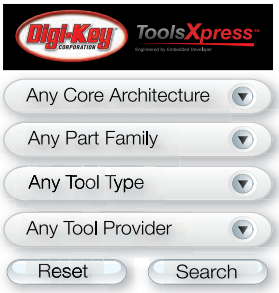
R. Williams, "A Painless Guide to CRC Error Detection Algorithms," Version 3, 1993, www.ross.net/crc/download/crc_v3.txt.

www.DevtoolsXpress.com

Pick a Tool. Any Tool.

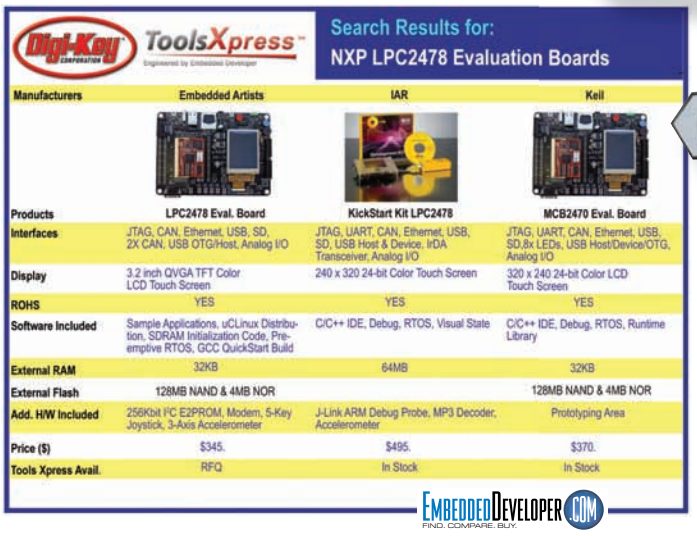
Find the Right Development Tool, Compare it to Other Tools, Evaluate It, and Buy It from Digi-Key Tools Xpress -- Without Leaving Our Site.

The Digi-Key Tools Xpress intuitive research engines are used by engineers worldwide to locate, compare and evaluate hardware or software development tools.



Digi-Key Tools Xpress, engineered by Embedded Developer, is the only site in the industry where engineers can quickly find, compare and buy the leading development tools.

Compare before you buy: tools are listed side-by-side, with features and performance specs, availability, and prices, so you can make an educated decision!



Digi-Key ToolsXpress — Search Results for: NXP LPC2478 Evaluation Boards

Manufacturers	Embedded Artists	IAR	Keil
Products	LPC2478 Eval. Board	KickStart Kit LPC2478	MCB2470 Eval. Board
Interfaces	JTAG, CAN, Ethernet, USB, SD, 2X CAN, USB OTG/Host, Analog I/O	JTAG, UART, CAN, Ethernet, USB, SD, USB Host & Device, IrDA Transceiver, Analog I/O	JTAG, UART, CAN, Ethernet, USB, SD,8x LEDs, USB Host/Device/OTG, Analog I/O
Display	3.2 inch QVGA TFT Color LCD Touch Screen	240 x 320 24-bit Color Touch Screen	320 x 240 24-bit Color LCD Touch Screen
ROHS	YES	YES	YES
Software Included	Sample Applications, uCLinux Distribution, SDRAM Initialization Code, Pre-emptive RTOS, GCC QuickStart Build	C/C++ IDE, Debug, RTOS, Visual State	C/C++ IDE, Debug, RTOS, Runtime Library
External RAM	32KB	64MB	32KB
External Flash	128MB NAND & 4MB NOR		128MB NAND & 4MB NOR
Add. H/W Included	256Kbit I²C E2PROM, Modem, 5-Key Joystick, 3-Axis Accelerometer	J-Link ARM Debug Probe, MP3 Decoder, Accelerometer	Prototyping Area
Price ($)	$345.	$495.	$370.
Tools Xpress Avail.	RFQ	In Stock	In Stock

EmbeddedDeveloper.com FIND. COMPARE. BUY.

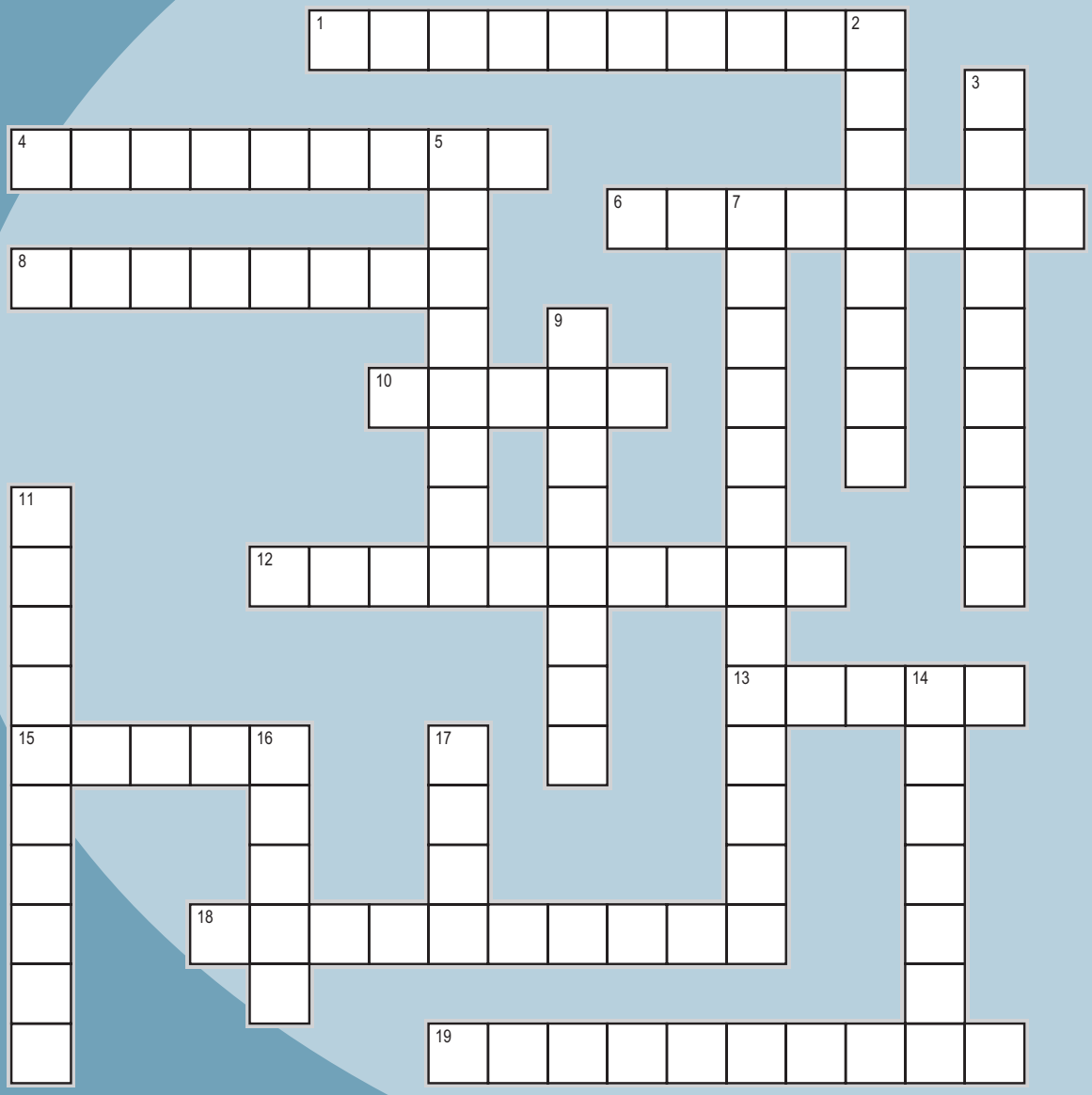


Join the thousands of engineers worldwide who use Digi-Key Tools Xpress for their development tool needs.

Digi-Key® CORPORATION

ToolsXpress℠
Engineered by Embedded Developer

CROSSWORD



Across

1. HP (hint: not a printer maker) [two words]
4. Not a lot of logic
6. Tap into your computer's memory
8. Ensures things are running smoothly
10. To exterminate, in computer terms
12. Sensitive to the degree
13. An update a bird may comprehend
15. Software used to talk over the 'Net
18. Needs to run before an OS can start
19. A unique ID that follows IEEE-managed rules [two words]

Down

2. A quick response
3. Founded a binary system [two words]
5. Used to measure electrical resistance
7. Performs an intervention when things go wrong [two words]
9. What it means to *, ·, or ×
11. Stop error [two words]
14. Used to trail off
16. Exposure to ultraviolet light makes it disappear
17. Design language (related to Cain)

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

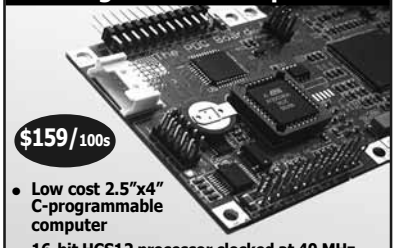
IDEA BOX

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, PO Box 180, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter Wostrel at 800.454.3741, 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.

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

bob-4
Low-cost video data overlay module
Features:
Automatic text scroll and crawl
'TTL-232' and SPI control ports
Any size for custom fonts
Vector and bitmap graphics
NEW REDUCED PRICING
www.decadenet.com
DECADE ENGINEERING
503-743-3194 Turner, OR, USA

Flashlite 186



- 186 processor @ 33 MHz
- DOS w/ Flash File system
- 44 Digital I/O lines w/ CPLD
- Console / Debug Serial Port
- 7-34V DC or 5V DC power • 2 Serial Ports
- Accepts 8MB DiskOnChip • 2 16-bit Timers
- 512K DRAM & 512K Flash • Watchdog Timer
- Expansion options with Peripheral Boards

$69 QTY 1

$99 Development System

Development kit includes:
- Flashlite 186 controller
- Borland C/C++ ver 4.52
- FREE Email Tech Support
- Serial Driver library
- AC Adapter and cable

Call 530-297-6073 Email sales@jkmicro.com
On the web at www.jkmicro.com
JK microsystems

Evaluate * Educate * Embed
Adapt9S12
Modular Prototyping System
For education & development

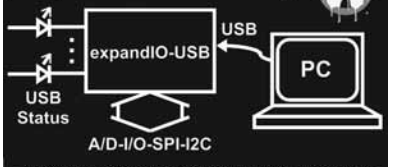


freescale Alliance Member

Topologies Supported:
Planar, Stack, Backplane, Tower
MCU Families Supported:
S12A, C, D, E, NE, XD, XE, XS

Toll-free (USA & Canada):
1-877-963-8996
www.TechnologicalArts.com

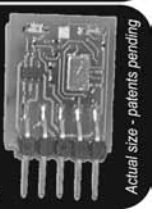
Revolutionary new expandIO-USB chip



- Ideal for adding USB to sensors & peripherals
- No drivers needed for Windows, Mac, Linux
- No microcontroller programming required
- Also check out our USB-232 USB to UART

www.hexwax.com - Buy from Mouser & Farnell

Amazing PIC programmer
Most devices supported
ICSP, SQTP, & copy limits
$32 at Digikey & Mouser
www.flexipanel.com



Actual size - patents pending

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

Optimized C Compiler for PIC® MCU

· Decrease code size by 40%
· Use less expensive MCU with smaller program memory
· Save hundreds of dollars in component costs in your projects



CCS Compilers Save You Money

Compilers Starting at $150

www.ccsinfo.com/SAVE
sales@ccsinfo.com
262.522.6500 x35

Add USB to Your Designs

Chips, code, protocols, embedded hosts, wireless options, debugging, **USB 3.0** and **SuperSpeed** too!



USB Complete
The Developer's Guide
Fourth Edition
Jan Axelson

ISBN 978-1-931448-08-6 $54.95
Lakeview Research LLC **www.Lvr.com**
By the author of *Serial Port Complete*

INTEGRATED
PLCs with Integrated Ethernet



New 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*
- *Laddder +BASIC Programming*

$229 and $295
before OEM Qty Discount

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

TRIANGLE RESEARCH INTERNATIONAL

CIRCUIT CELLAR®
CC-ACCESS
SUBSCRIBERS RESERVED AREA

Sign up for a CC 2.0 subscription today and receive automatic entry to CC-Access!

- Gain instant access to source code & project files
- Search for your favorite articles at supersonic speeds
- Quickly download your issue PDF when you are a CC 2.0 Digital or Combo subscriber
- Easily maintain account information
- Receive VIP treatment with special offers on products & services you can't live without!

All this and more at...
www.CC-Access.com
(Already a subscriber? Log in now!)



Weather Instruments for PCs

AAG electrónica

www.aagelectronica.com

MaxSonar
Perfect for OEMs & Engineers

XL-MaxSonar-EZ
• High acoustic power
• Real-time calibration
• Tiny size
• $39.95 USD qty. 1

MaxSonar-WR (IP67)
• Industrial packaging
• Weather resistant
• Standard ¾" fitting
• $99.95 USD qty. 1

LV-MaxSonar-EZ
• Power-up calibration
• Smallest MaxSonar
• Low power, 2.5V-5.5V
• $29.95 USD qty. 1

MaxSonar-WRC (IP67)
• Compact packaging
• Weather resistant
• Quality narrow beam
• $99.95 USD qty. 1

www.maxbotix.com

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

Senix®
Distance Measurement
Dimension
Proximity
Position
Level
Industrially Rated
Non-Contact Sensors
PC & Industrial Interfaces
Free PC Software
Water Tank
118.4
GALLONS
SenixMETER for PC
802-453-5522
cci@senix.com
ultrasonicsensors.com

Exactly why is USB hard?
Part# QUSB2 - $149 qty 1
* 25MB/sec data bus, bit-bang, RS-232, I2C Master, SPI Master, FPGA port
* Software library and example code
* New, high-performance kernel drivers for Windows, Linux and MacOS
QuickUSB®
www.quickusb.com

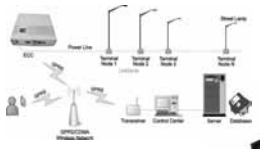
BPS BusBoard Prototype Systems
Prototyping PC Boards
Many Patterns
BusBoard
StripBoard
PadBoard
ProtoBoard
ProtoBoard-2H
PowerBoard
SMTBoard
SMTpads
PC BreadBoards
See our site:
www.BusBoard.us
Available From
JAMECO ELECTRONICS
MOUSER ELECTRONICS
a ttl company
amazon.com amazon.co.uk

i2cStick™
Personal, Powerful...
Pocket-Size
MCC
Micro Computer Control
www.mcc-us.com

Display with Touch Screen
• Embedded, Standalone, C/C++ Programmable...
• Low Cost User Interface with Ethernet, USB, CF, RS232, ADC, DAC, I/Os, Relays...
• U-Drive™ 5.7" QVGA TFT, CAN, Ethernet, CF, ADC, DAC, ...
start at $379 OEM
60+ Low Cost Controllers with TFT, ADC, DAC, UARTs, 300 I/Os, solenoid, relays, Compact-Flash, 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

www.can232.com
Only $108 €89
The original (Controller Area Network) CAN to RS232 serial converter.
CAN232 Features:
Free sample programs
8-15VDC supply via CAN
Timestamp in mS
Small size 2.7" by 1.2"
100% Bandwidth up to 125Kbit
Both 11 & 29 bit ID support
32 Message Receive FIFO
Works up to 1Mbit CAN
Simple ASCII protocol
Supports RTR Frames
Max 230Kbaud RS232
Firmware upgradable
No drivers needed
OS independant
CE Approved
CANUSB Features:
Free ActiveX component
PC, MAC & Linux support
Both 11 & 29 bit ID support
Simple CAN logger included
Free Threaded Windows DLL
Firmware upgradable via USB
Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.
No need for external power
Works up to 1Mbit CAN
Supports RTR Frames
USB 2.0 Full Speed
Free USB drivers
CE Approved
Only $154 €129
www.canusb.com

Power-Line Communication, Street Lighting System, UHF EPC Gen2 RFID Reader, Arduino Shields (WiFi)



■Power-Line Communication Street Lighting System



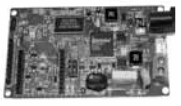
■Power-Line Communication Modules
- ➢ Low-Speed Modem Pro
- ➢ Low-Speed Shield for Arduino – The first one in the world!!!
- ➢ High-Speed Modem
- ➢ High-Speed Modem for DMX512

■Arduino Shields: WiFi, GPRS, Charger, IR, Smart Outlet

■JPEG Serial Camera

■Cellular modules

■GPS, Bluetooth modules



■ LS6410 embedded system with Win CE 6.0, Linux, and Android.

www.linksprite.com

Control by USB or Web!





TCPmaker for control over the Web

Easy as 1-2-3:
1. **Define Your Data**, with variable names that YOU create.
2. **Lay Out Your Content** (with gorgeous web-ready screen controls that you can ***grab on to***) using TCPmaker's drag & drop Visual Page Designer.
3. **Generate Your Code**, for all Microchip C compilers, to "wire it all together."

NO PC PROGRAMMING AT ALL – just point your web browser at your device!

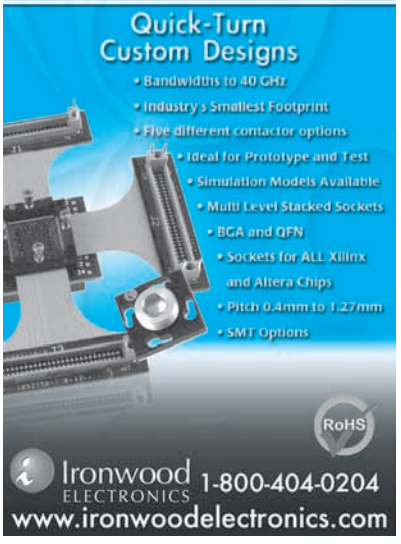
Simple Upgrade Management System

Low cost integrated system manages upgrades across different processors & connectivity types:
> Lets your end users upgrade your PIC firmware safely, simply, and securely.
> **Automatically delivers upgrade by email.**
> Simple for non-technical end users.
> **Encrypted system** protects your firmware from theft or from being programmed into the wrong device.
> **Grows with your product line:** for a new product w/ different processor or connectivity type, just add another SUMS bootloader.

Trace SYSTEMS, Inc.

From the makers of HIDmaker FS:
www.tracesystemsinc.com
888-474-1041

BGA & QFN Sockets



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

microEngineering Labs, Inc.
www.melabs.com 888-316-1753

PICBASIC PRO™ Compiler



Bridges the gap between ease of use and professional level results. The simplicity of the language allows both hobbyist and engineer to master it quickly, but the engineer appreciates what's happening under the hood. This is a true compiler that produces fast, optimized machine code that's stable and dependable.

Supports:
More than 300 PIC® Microcontrollers
Direct Access to Internal Registers
In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging in MPLAB

PICBASIC PRO™ Compiler $249.95

USB Programmer for PIC® MCUs

$89.95 *(as shown)*

RoHS Compliant



Programs PIC MCUs including low-voltage (3.3V) devices. Includes Software for Windows 98, Me, NT, XP, and Vista.

With Accessories for $119.95:
Includes Programmer, Software, USB Cable, and Programming Adapter for 8 to 40-pin DIP.

See our full range of products, including books, accessories, and components at:
www.melabs.com

CROSSWORD ANSWERS from Issue 249

Down

2. LIGHTEMITTINGDIODE—Used to brighten things [three words]

3. HYPERTEXTMARKUPLANGUAGE—Used to weave the Web [four words]

7. ROBOT—Mechanical, not real

8. NANOSECOND—10^{-9} s

9. ROTFL—Common text message: "Rolling on the floor laughing"

11. PASSWORD—Required to login

13. HEATSINK—Creates fluidity, such as liquid or air

17. OMEGA—O

20. OPAMP—Used to build electronic circuits [two words]

Across

1. LOCALBUS—Makes direct connections [two words]

4. ZUCKERBERG—Social networking guru

5. THREAD—A group of messages

6. PARENTHESIS—Used to enclose text

10. SSH—Protocol allowing the exchange of data between networked devices

12. KELVIN—Between J and L

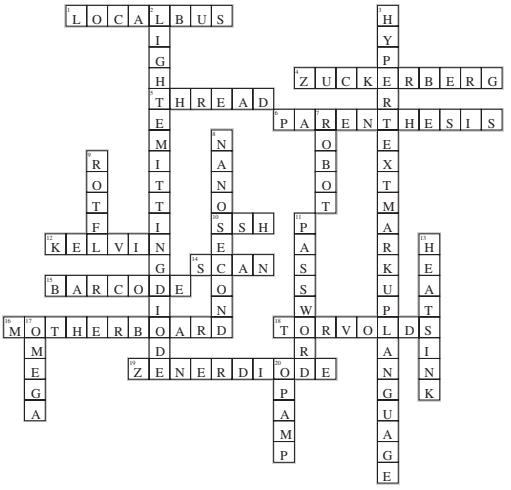
14. SCAN—One way a computer reads data

15. BARCODE—Data, a series of lines [two words]

16. MOTHERBOARD—Another word for main board or system board

18. TORVOLDS—Creator of Unix-type operating systems

19. ZENERDIODE—Permits current forward and backward



INDEX OF ADVERTISERS

Page		Page		Page		Page	
76	AAG Electronica, LLC	53	ExpressPCB	76	Lakeview Research	12	Saelig Co., Inc.
75	All Electronics Corp.	43	ezPCB	77	Lawicel AB	63	Sensors Expo & Conference 2011
51	AP Circuits	75	FlexiPanel Ltd.	13	Lemos International Co., Inc.	77	Senix Corp.
35	ARM	23	Futurlec	78	LinkSprite	54	SID Int'l Symposium
77	Bitwise Systems	43	Grid Connect, Inc.	77	Maxbotix, Inc.	32	Tag-Connect
77	BusBoard Prototype Systems Ltd.	25	Holtek Semiconductor USA	77	MCC, Micro Computer Control	2, 3	Technologic Systems
7	Comfile Technology, Inc.	11	Humandata Ltd.	78	microEngineering Labs, Inc.	75	Technological Arts
76	Custom Computer Services, Inc.	67	IC Bank	51	Midwest Patent Law	77	Tern, Inc.
33	CWAV	44, 45	Imagineering, Inc.	75	Mosaic Industries, Inc.	78	Trace Systems, Inc.
75	Decade Engineering	78	Ironwood Electronics	5	Mouser Electronics, Inc.	76	Triangle Research Int'l, Inc.
31, 59	Elektor	C3	Jameco	C2	NetBurner	9	WIZnet Co., Inc.
23	EMAC, Inc.	71	Jeffrey Kerr, LLC	29	NXP Semiconductors		
49	ESC Chicago	13, 32	JK microsystems, Inc.	C4	Parallax, Inc.		
61	ESEC Japan	75, 77	JK microsystems, Inc.	10	Pico Technology Ltd.		
73	Embedded Developer	19	Labcenter Electronics UK	24	Pololu Corp.		

PREVIEW of June Issue 251

Theme: Communications

The Chameleon Platform: Low-cost Hardware Development with a USB AVR and CPLD

Wireless Authentication: RFID Checkout System Design

The Net Butler: Construct a Multifunctional Network Controller

Build a Fixed Cellular Phone with Emergency Auto-Dial

THE DARKER SIDE **Stay Cool!:** Temperature Management Primer

THE CONSUMMATE ENGINEER **BITs and BITEs:** Tests for Ensuring System Safety

ABOVE THE GROUND PLANE **Plastic Extruder Thermal Analysis**

FROM THE BENCH **Vehicle Diagnostics (Part 1):** The CAN Protocol and OBD-II

ATTENTION ADVERTISERS

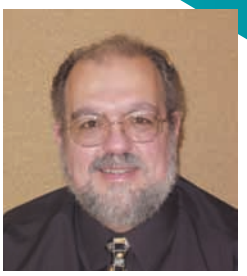
July Issue 252 Deadlines

Space Close: May 12
Materials Close: May 19

Theme
Internet & Connectivity

Call Peter Wostrel
now to reserve your space!
800.454.3741 or 978.281.7708
e-mail: peter@smmarketing.us

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Happy 250th

Believe it or not, I almost missed this. Sometimes, when I'm down at the cottage, I lose track of events. It wasn't until I saw a proof of C. J. Abate's "Task Manager" (p. 4) that I realized this is the 250th issue. Wow, that's 23 years!

I know a lot of you have been with me since day one (feeling kind of old too, I bet). Many of you know the history of *Circuit Cellar* and how we all got here, but now that *Elektor* and its international audience have joined *Circuit Cellar,* I find I've been getting more questions about how it all started and why we are who we are. Perhaps the 250th issue, 23 years later, is the perfect time to recap the story for those who don't know it.

First of all, the "Circuit Cellar" is a real place. Back in 1977, I was an engineer at a big aerospace company that believed any computer not delivered in six trailer truck loads was too insignificant to bother with. I was left on my own to experiment and document the great things happening with these "new-fangled" microprocessors. I even wrote a couple of articles for a new "small systems journal" called *BYTE*. The success of the articles was immediate, and after the editor, Carl Helmers, visited my house and saw my basement "lab" and all of its equipment, he suggested that I start a column called "Ciarcia's Circuit Cellar." The circuit cellar "lab" still exists today, and it only takes one whiff of solder fumes to remind me of everything back at the beginning.

"Ciarcia's Circuit Cellar" was unique. It was a 10- to 14-page monthly article that presented a technical problem and then described the detailed construction of a piece of hardware that solved that problem. (The articles are posted on Google Books.) The degree of complexity in the projects advanced with the prevailing technology. Early projects were mostly parallel- or serial-port-connected expansion peripherals because that was the prevailing computer connection. Later projects included more standalone devices, home control systems, and commercial-quality embedded controllers and peripherals. (I guess we were on a roll.) I have to admit that at the conclusion of my time with *BYTE*, a few projects even went a bit overboard. One of the last "Ciarcia's Circuit Cellar" complete-instruction DIY projects 24 years ago was a VGA Mandelbrot generator with 64 processors!

Somewhere in the middle of my tenure with *BYTE*, the journal was purchased by McGraw-Hill. Column popularity and the advertising revenue generated by "Ciarcia's Circuit Cellar" pretty much kept the publishing company off of my radar and out of my face until a few years later when it decided to commit the publishing equivalent of hari kari. The premise of "Circuit Cellar" and the popularity of *BYTE* was always that computers were ubiquitous—not that a specific computer should be everywhere, but the knowledge to design and use any computer should be everywhere. I didn't reject that IBM had a good computer and a massive following. I rejected that McGraw-Hill changed *BYTE*'s editorial direction to PC-everything and dictated that nothing else existed from then on. I was given a choice to take the money and be PC-compatible (and quiet) or collect my marbles and leave.

The rest, as they say, is history. But perhaps you don't know that either. My last article in *BYTE* was in late 1988. Coincidentally (or not) *Circuit Cellar Ink* magazine started the month before I left *BYTE* (it had "Ink" added to its name initially for trademark reasons). The good news was that I didn't have to do it all myself—far from it. There were a number of people upset about the changes at *BYTE*, and I had a lot of support. When *Circuit Cellar Ink* started as a bi-monthly, the staff included *BYTE*'s former publisher, its circulation director, its chief editor, and me (along with the staff of great people behind "Ciarcia's Circuit Cellar" including Ed Nisley, Jeff Bachiochi, and Ken Davidson). We started the magazine so quickly it didn't have a lot of advertising, but it was on national newsstands almost immediately.

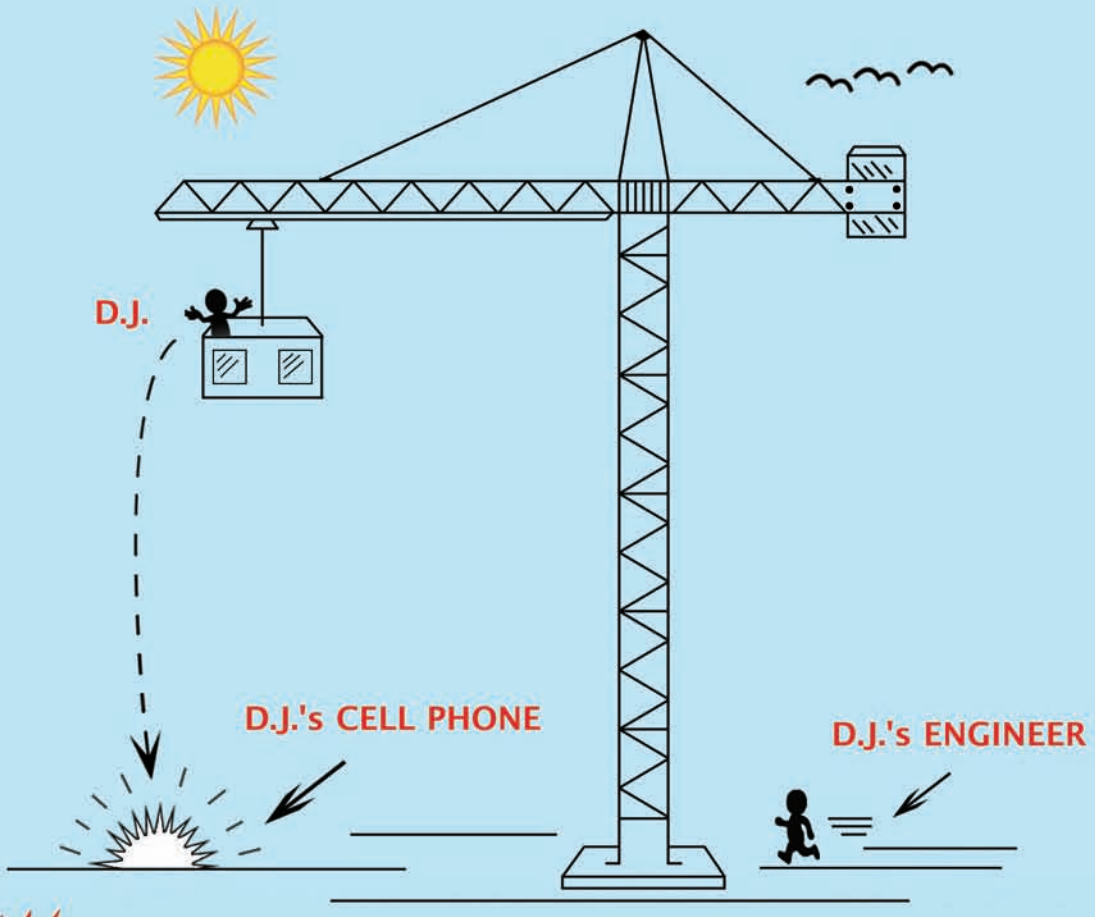
I think my idea for a *BYTE* column and an embedded design magazine is still pretty sound. C. J. told you the title of the first issue of *Circuit Cellar* magazine. Now that you know a bit more of the magazine's history, perhaps you can understand that at the time it was my way of saying to McGraw-Hill: "Inside the box still counts." (And, by the way, I'm putting my money where my mouth is!) Of course, the proof is in the pudding, as they say.

Comparing the publishing economics of *Circuit Cellar* and *BYTE* is ridiculous, but let's just say that if enough readers subscribe you don't trash a magazine. If my knowledge is correct, McGraw-Hill closed *BYTE*. Then CMP took it, had it a while, and closed it too. (UBM Techweb is the latest owner and will be starting an online version soon.) *Circuit Cellar* (first in *BYTE* and then as a magazine) has existed uninterrupted for 34 years and is still going strong! Joining with *Elektor* adds new audiences, resources, and products that will certainly result in *Circuit Cellar* magazine being around for many more years to come.

steve.ciarcia@circuitcellar.com



Can You Solve This?





Sign up for our e-newsletter and get a FREE comic book.
www.Jameco.com/adventure1

Radio disk jockey D.J. Smith organized a stunt to raise donations for a charity by conducting his radio show for one week while suspended from a 150-feet tall crane. A cell phone with a solar charger served as D.J.'s only link to the radio station. His little home in the sky included foil-wrapped freeze-dried meals, bottled water, binoculars, pencil, notebook, paper clips, rubber bands, pocket knife, MP3 player, and a portable radio. The weather cooperated with light wind, clear skies and pleasant temperatures. All went well untill D.J. dropped his cell phone to the concrete below. The stunt rules prevented any physical contact between D.J. and the ground. How did he continue his show? (hint: D.J.'s engineer was an Alexander Graham Bell history buff). Go to www.Jameco.com/teaser10 to see if you are correct. While you are there, sign up for their catalog. Forrest M. Mims III

1-800-831-4242 | www.Jameco.com

JAMECO® ELECTRONICS



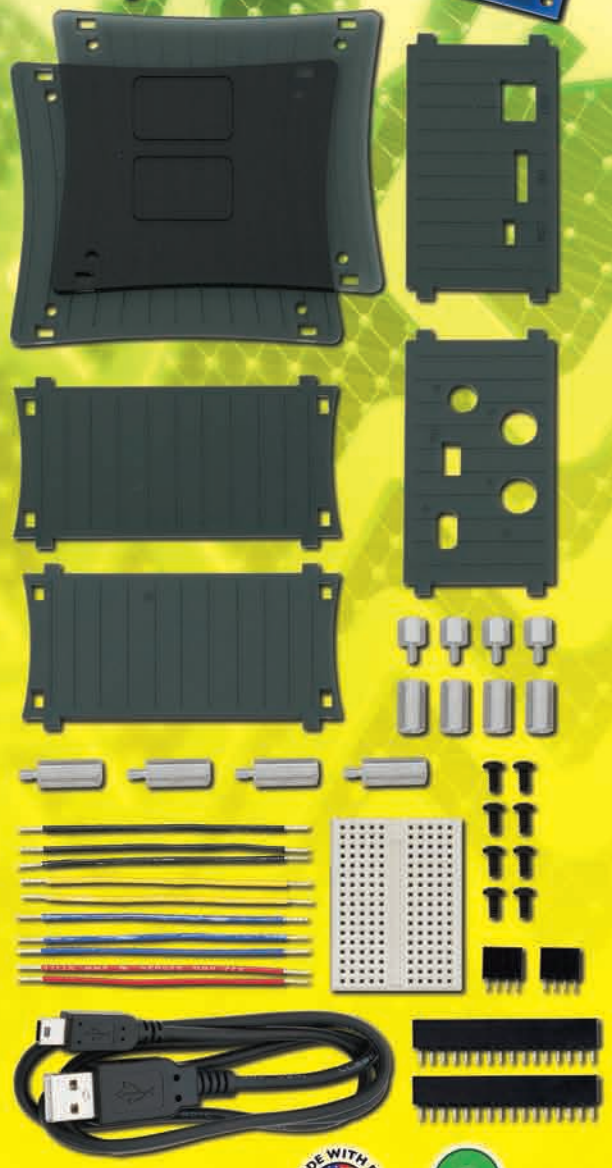
PROPELLER MULTICORE STARTUP SPECIAL!

$79.99

A great way to play with the multicore Propeller P8X32A microcontroller at a great price! This open-source, made-in-the-USA, limited-time-only kit includes a Gadget Gangster Propeller Platform USB, a ProtoPlus module, and a cool pinstriped acrylic enclosure to house your completed project!

Combining a 64 KB EEPROM, 5 MHz removable crystal, 1.5 A power regulation, USB, and a microSD card slot on a compact, breadboard and protoboard friendly module, the Propeller Platform USB is an easy-to-use development tool for the multicore Propeller microcontroller. All 32 Propeller I/O are available via pin sockets, along with 5 and 3.3 V regulated power. The ProtoPlus module adds video and audio to your Propeller Platform and includes a prototyping area. Traces in the prototyping area are also marked in the silkscreen, so it's easy to see how the traces are connected. **We have several tutorials available online to get you started.**





Order the **Propeller Multicore Starter Special** (#910-32316; $79.99) at www.parallax.com or call toll-free at 888-512-1024 (M-F, 7AM-5PM, PDT).

Friendly microcontrollers, legendary resources.™

Prices subject to change without notice.
Propeller, Parallax and the Parallax logo are trademarks of Parallax Inc.

PARALLAX INC

www.parallax.com

"ParallaxInc" on Twitter, Facebook, and YouTube