MCU-Based Filtration Monitoring (p. 52) • Design & Program A USB Virtual COM Port (p. 70) • New Developments In Soft-Core Technology (p. 78)

www.circuitcellar.com

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#217 August 2008

EMBEDDED DEVELOPMENT

Two-Stage Bootloader Development

Electric Vehicle Inverter Design

Build A DMX Lighting Controller

Direct Digital Synthesis Explained

Enhance Your Single-Chip App

IR Distance Measurement



0 74470 75349 0 08>

$5.95 U.S. ($6.95 Canada)

SERIAL TO ETHERNET SOLUTIONS



Device P/N: SB70-100CR
Kit P/N: NNDK-SB70-KIT

$69
Qty. 100

SB70

2-port serial-to-Ethernet server
with TTL support



Device P/N: SB72-EX-100CR
Kit P/N: NNDK-SB72EX-KIT

SB72EX

$139
Qty. 100

2-port serial-to-Ethernet server
with RS-232 & RS-485/422 support



Device P/N: PK70EX-232CR
Kit P/N: NNDK-PK70EX232-KIT

$269
Qty. 100

PK70EX232

4-port serial-to-Ethernet server
with RS-232 support

Simple Ethernet connectivity
for serial devices

Works out of the box -
no programming is required

Customize to suit any
application with low-cost
development kit

Features:

10/100 Ethernet

TCP/UDP mode

DHCP/Static IP Support

Data rates up to 921.6kbps

Web-based configuration

Need a custom solution?

NetBurner **Serial to Ethernet Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications. All kits include platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, cables and power supply.

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

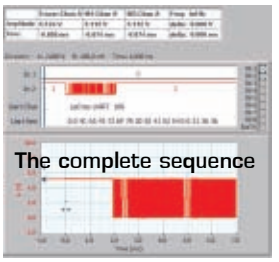
Thinking about your next scope?

Compare the specs:

Item	Tektronix TDS2012B	Agilent MSO6012A	Pico 3206	Link DSO-8502	Bitscope USB 310	TiePie HS3-100	cleverscope CS328A
Analog Input FSD	20 mV – 50V	20 mV – 50V	±100 mV – ± 20V	50 mV – 50V	±513mV – ± 10.8V	±100 mV – 80V	**18 mV – 80V**
Overload Protection	**300 Vrms**	**300 Vrms**	±50V	±150V	±100V	200V	**300 Vrms**
Analog Channel bit resolution	8	8	8	8	8	8 or 12	10, 12 or 14
Can do hardware trigger referenced averaging	Yes (10 bit)	Yes (10 bit)	-	-	-	-	**Yes (14 bit)**
Analog Input Offset	±2V:→200mV then ±50V	**±25V:→200mV then ±100V**	None	±5 divisions	None	None	±3.5V → 200mV then ± 80V
External trigger sampled, stored & displayed?	No	No	No	No	No	No	**Yes**
Samples of storage, all inputs recorded	2.5k	2000k or 8000k	500k	512K	64k	128k	**4000k or 8000k**
Resolution on 2V offset signal	0.078 mV	0.078 mV	15.6 mV	15.6 mV	15.6 mV	15.6 mV	**0.018 mV**
Tracking graph with independent time base	No	No	No	No	No	No	**Yes**
Dynamic Range (raw unprocessed)	48 dB	48 dB	48 dB	48 dB	48 dB	48 dB	**60 dB or 72 dB or 84 dB**
Maximum time resolution	1ns	**0.5ns**	10ns	4ns	50ns	10ns	10ns
Resolution at 20ms width	8 us	**10 ns**	40 ns	40 ns	320 ns	200 ns	**10ns**
Width at max sample rate	2.5us width	1 ms width	5 ms width	320 us width	640us width	1.3 ms width	**20 or 40 ms**
Peak Captured to eliminate aliasing	**Yes**	**Yes**	**Yes**	No	No	No	**Yes**
Trigger delay	**Yes**	**Yes**	No	No	**Yes**	No	**Yes**
Protocol Analysis	No	**Yes**	No	No	No	No	**Yes**
Period Trigger	Yes	**Yes**	Yes	**Yes**	No	No	**Yes**

the review continues on our website at *www.cleverscope.com/review/*

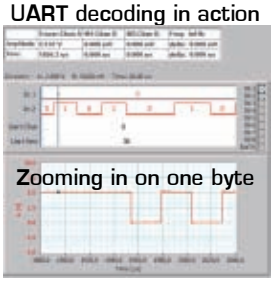
- 100 MHz Bandwidth
- Dual mixed sig trigger
- 2 analog, 8 digital inputs and ext trig
- 10,12 or 14 bit sampling
- 4 or 8 MSamples/channel storage
- 100 MSamples/sec
- Battery option
- USB or ethernet
- Data projector display
- Regular software and firmware updates
- Simple cut and paste into documents



UART decoding in action



Protocol selection and setup screen



Protocol Analysis

USA master distributor

Saelig unique electronics

The CS328A is an engineer's toolbox

Oscilloscope—Tracking Graph—Spectrum Analyzer
Protocol Analyzer—Multimeter—Logger—Result Storage
Maths Equation Builder and Display—Function Generator
XY Graph—Drivers for C, Dephi and Labview
Custom Names, Units and Scaling

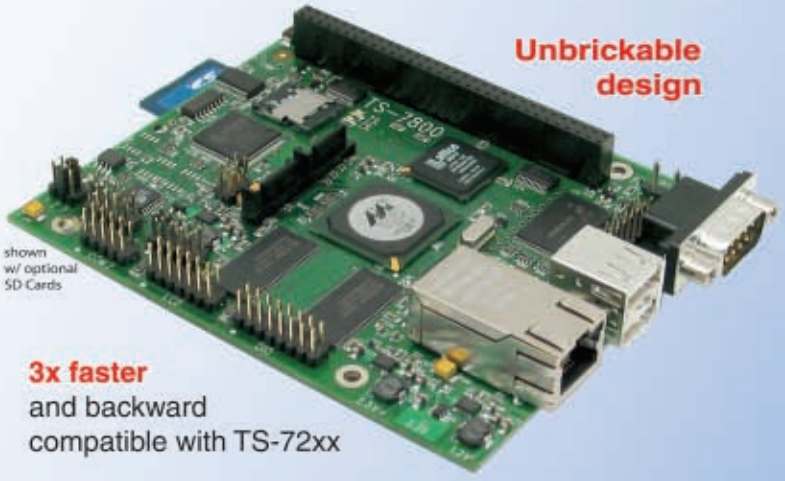
cleverscope

plug & play instruments

www.cleverscope.com

Embedded Single Board Computers

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 programmable 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 in < 2 seconds
- Linux 2.6 and Debian by default

Low Price, Low Power, High Reliability using Linux development tools



TS-7200 shown with optional A/D converter, Compact Flash and RS-485

- options include:
  onboard temperature sensor, A/D Converter 8 channel 12 bit, Extended Temperature, Battery Backed Real Time Clock, USB Flash, USB WiFi

200 MHz ARM9
Power as low as 1/4 Watt

- 8 boards, over 2000 configurations — as low as **$99** qty 100
- Fanless, no heat sink
- SDRAM - up to 128MB — **$129** qty 1
- Flash - up to 128MB onboard
- 10/100 Ethernet - up to 2
- DIO lines - up to 55
- 2 USB ports
- COM ports- up to 10
- Programmable FPGAs
- SD card option
- VGA video
- LCD ready
- Linux, Real Time extension, Debian

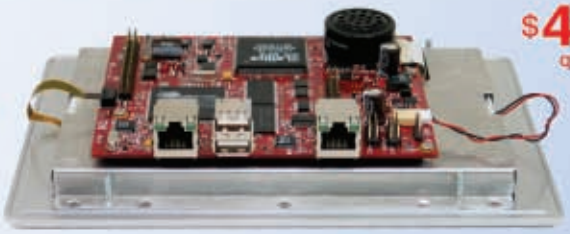
- Over 20 years in business
- Open Source Vision
- Never discontinued a product
- Engineers on Tech Support
- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products stocked and available for next day shipping

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

Featured Products and PC/104 Peripherals

NEW!





$449 qty 1

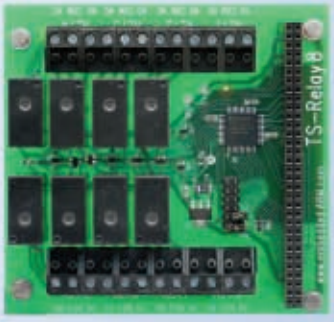
TS-TPC-7390
200MHz ARM9 Touch Panel Computer

- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 64-128MB DDR RAM
- 512MB Flash w/ Debian Linux
- Programmable FPGA- 5K LUT
- 7" Color TFT-LCD Touch-Screen
- 800x480 customizable video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in about 1 second
- Unbrickable, boots from SD or NAND
- Runs X Windows GUI applications
- Runs Eclipse IDE out-of-the-box

NEW!

TS-RELAY8
Eight Software Controlled Relays

- 8 SPDT relays
- Up to 277 VAC @ 5A
- Up to 30 VDC@ 5A
- Software controlled
- 40mA draw per coil
- I/O jumpers



$89 qty 1

NEW!

TS-ADC16
ADC, DAC and Digital I/O

- 16 16-bit ADCs
- Up to 100Ksps (10us)
- 1KB ADC RAM-FIFO
- 4 ADC voltage ranges
- Prog. pacing clock
- Externally triggered
- 4 inputs, 1 output
- 4 12-bit DACs
- 4 16 bit counters

$169 qty 1



see our website for x86 SBCs, peripherals and option details

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

TASK MANAGER

Hone Your Embedded Development Skills

If you are looking for up-to-date information about embedded development techniques and technologies, this is the issue for you. Our feature writers and columnists present in-depth articles about several design projects and advanced technologies that will help you take your embedded development skills to the next level.

Beginning on page 14, Matt Ernst explains how he harnessed the power of a WIZnet Ethernet interface and a Microchip Technology PIC microcontroller to build a DMX lighting controller. The system enables him to remotely control distributed lighting systems.

As you know, most of the ground-breaking intelligent energy solutions of the 21st century will feature well-designed, integrated embedded technologies. During the past several months, we've been featuring such projects in our "Intelligent Energy Solutions" section. This month's project is the electric vehicle inverter described in "Electric Vehicle Inverter Design" (p. 22). The innovative design enables you to power AC induction motors.

Do you have a design that needs to be updated in the field in order to support bug fixes and other features for specific end-user apps? If so, check out Dave Tweed's two-stage bootloader that adds additional capabilities to the native boot processing of the Blackfin chip (p. 34).

Turn to page 44 for the second part of Chris Paiano's series about helpful PSoC design techniques. He describes how to enhance his eight-channel mixer project with DSP effects, a user interface, an intercom mode, and more.

Starting on page 70, Jan Axelson describes the development of a USB virtual COM port. As you know, physical COM ports are becoming a thing of the past. This project proves you can use an MCU with a USB controller to build and program a USB virtual COM port of your own.

In "IR Proximity Sensing," Ed Nisley describes the basics of reliable IR sensing (p. 40). He explains how he updated a hand-mounted chord keyboard with sensors that enable him to activate switches with his thumb. You can use the technique for many of your future designs.

Do you get your water from a well? Columnist Jeff Bachiochi does. In fact, his neighbors use the well too. Because several households pull from the same water source located on his property, Jeff is the guy who handles its upkeep. Being the inventive engineer that he is, Jeff recently built an MCU-based monitoring system for the well. In "Water War Prevention," he explains how the design enables him to keep tabs on the system without having to "periodically enter the dungeon pit" (p. 52).

At the Embedded Systems Conference in San Jose last April, several readers told me how much they loved Robert Lacoste's solution-focused columns. This month, Robert delivers another handy article. He reintroduces you to direct digital synthesis (DDS) by covering the topics of DDS theory, firmware implementation, and chip-based solutions (p. 60).

Tom Cantrell wraps up this issue with an interesting article about "soft-core" technology. Does it make sense for your current application? Before you answer this question, check out what Tom has to say about the ARM Cortex-M1 and FPGA chips (p. 78).

Happy embedded developing!

cj@circuitcellar.com



CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Ingo Cyliax
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

ASSOCIATE EDITOR
Jesse Smolin

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

MEDIA CONSULTANT
Dan Rodrigues

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

STAFF ENGINEER
John Gorsky

ADVERTISING
860.875.2199 • Fax: 860.871.0411 • www.circuitcellar.com/advertise

PUBLISHER
Sean Donnelly
Direct: 860.872.3064, Cell: 860.930.4326, E-mail: sean@circuitcellar.com

ADVERTISING REPRESENTATIVE
Shannon Barraclough
Direct: 860.872.3064, E-mail: shannon@circuitcellar.com

ADVERTISING COORDINATOR
Valerie Luster
E-mail: val.luster@circuitcellar.com

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

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS

Information: www.circuitcellar.com/subscribe, E-mail: subscribe@circuitcellar.com

Subscribe: 800.269.6301, www.circuitcellar.com/subscribe, 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 $23.95, Canada/Mexico $34.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $43.95, Canada/Mexico $59.95, all other countries $85.** 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 © 2008 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.

USB Connectivity for Embedded Designs



If you need Full-Speed USB 2.0 device, embedded host, dual role and On-The-Go solutions, Microchip Technology has them available today. We offer 8-, 16- and 32-bit MCUs with USB connectivity, providing easy migration with a single development environment. This maximizes pin compatibility and seamless code migration from 20 to 100 pins, enabling you to scale your USB design with ease.

Download FREE USB software including source code:

- Host Stack
- OTG Stack
- Device Stack
- Class Drivers (HID, Mass Storage and CDC Drivers)
- Thumb Drive Support (Mass Storage Driver, SCSI Interface, 16-bit and 32-bit File Management, Application Software)

USB Starter Kits accelerate development of USB designs using 8-, 16- or 32-bit MCUs starting at only $59.98



Core	Flash Program Memory	Pins	USB Type
8-bit	Up to 128 Kbytes	20 - 80	Device
16-bit	Up to 256 Kbytes	64 - 100	Device, Embedded Host, Dual Role, OTG
32-bit	Up to 512 Kbytes	64 - 100	Device, Embedded Host, Dual Role, OTG

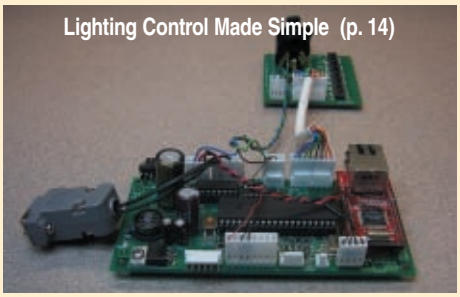
The Microchip name and logo, the Microchip logo and PIC are registered trademarks of Microchip Technology Incorporated in the USA and in other countries. © 2008, Microchip Technology Incorporated. All rights reserved.

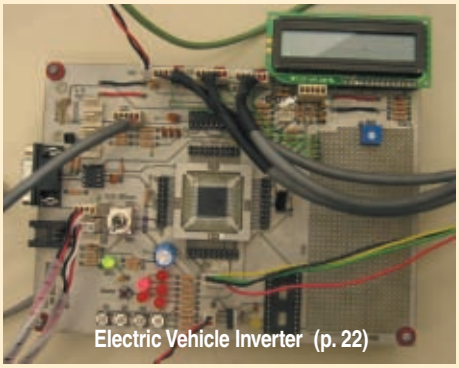
August 2008: Embedded Development

FEATURES

14 **The DMX Portal**
Obtain Lighting Control Via Ethernet
Matt Ernst
Third Place — WIZnet iEthernet Design Contest 2007

22 **INTELLIGENT ENERGY SOLUTIONS**
Electric Vehicle Inverter Design
Build A System For Powering AC Induction Motors
Dan Hall, Tristan Kasmer, Doug Krahn, Adam McIntyre, & Dena Ponech
Subcategory Winner — Microchip 2007 Design Contest


Lighting Control Made Simple (p. 14)


Electric Vehicle Inverter (p. 22)



A Two-Stage Bootloader (p. 34)

34 **A Bootloader For Blackfin**
David Tweed

44 **PSoC Design Techniques (Part 2)**
Add DSP Effects, A User Interface, And More
Chris Paiano

70 **Create A USB Virtual COM Port**
Jan Axelson

COLUMNS

40 **ABOVE THE GROUND PLANE**
IR Proximity Sensing
Ed Nisley

52 **FROM THE BENCH**
Water War Prevention
An MCU-Based Monitor For A Communal Well
Jeff Bachiochi

60 **THE DARKER SIDE**
Direct Digital Synthesis 101
Robert Lacoste

78 **SILICON UPDATE**
Icy Hot
The Soft-Core Concept, FPGAs, And You
Tom Cantrell

DEPARTMENTS

4 **TASK MANAGER**
Hone Your Embedded Development Skills
C. J. Abate

8 **NEW PRODUCT NEWS**
edited by *John Gorsky*

93 **CROSSWORD**

94 **INDEX OF ADVERTISERS**
September Preview

96 **PRIORITY INTERRUPT**
A Reactive Necessity
Steve Ciarcia



AVR® picoPower™

Learn how to combine the AVR® microcontrollers high performance with the lowest possible power consumption on www.atmel.com/avrman

ATMEL

© 2007 Atmel Corporation. All rights reserved. Atmel®, AVR® and logo are registered trademarks and picoPower™ is a trademark of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others. All Characters in this document are created by Mykle and Fantasi-Fabrikken AS 2007.

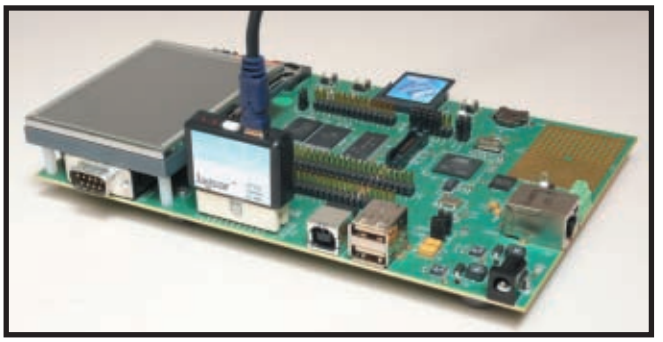
NEW PRODUCT NEWS

Edited by John Gorsky
Visit www.circuitcellar.com/npn for more New Product News.

MMU SUPPORT ADDED TO ARM DEBUGGER

Crossware, a leading embedded software tools developer, has added support for the ARM9 memory management unit (MMU) and fast context switch extension (FCSE) to its **ARM Development Suite**. This enables developers to rapidly exploit the advanced memory management features of ARM9 chips such as the Atmel AT91SAM9263 and AT91RM9200 microcontrollers.

Full debugging is possible when the MMU is enabled even when complex virtual to physical memory translation is used. The debugger understands the memory address translation process and can determine which physical memory is being used for different virtual addresses.



Full debugging is also possible when the FCSE is used. The FCSE allows different processes to easily share the same memory space. The debugger knows which executable code belongs to each process and can select the appropriate debugging information even though the processes share the same execution address.

The ARM simulator will also simulate the operation of the MMU and the FCSE. This enables code, which uses these features to be tested on the developer's PC before being tested on the target hardware. It also enables developers to explore the operation of the MMU and FCSE without any hardware at all.

The ARM Development Suite provides a complete and extremely user-friendly development environment for the ARM family of microprocessors and microcontrollers with its advanced C/C++ compiler, libraries, wizards, simulator, source-level debugger, and the Jaguar USB JTAG debugger interface.

The ARM Development Suite costs around **$2,600**.

Crossware Products
www.crossware.com

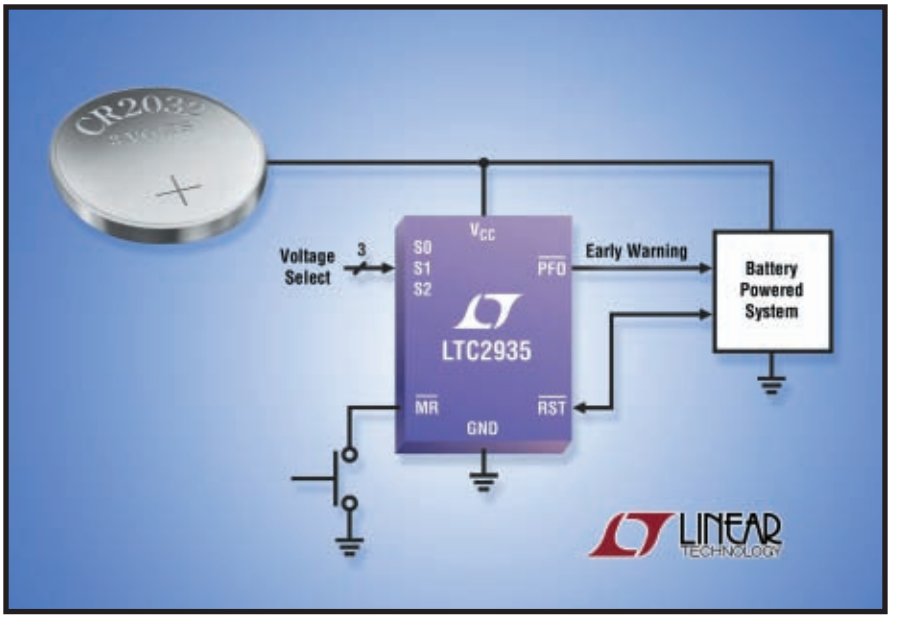
ULTRA-LOW-SUPPLY CURRENT MONITORS

Battery life is of the utmost importance in portable applications, where it is essential to reduce the amount of supply current the device draws during normal and standby operation. The new **LTC2934** and **LTC2935** ultra-low-power supply monitors draw only 500 nA of quiescent current.

The unique features of the LTC2934 and LTC2935 differentiate them in the low-power monitor market and make them excellent choices for single Li+ coin cell, AA, AAA, and other compact battery-powered or "green" applications. The LTC2935 has three binary inputs that allow the selection of one of eight integrated reset thresholds, from 3.3 down to 2.25 V in 150-mV increments. The device's internal precision attenuators allow the part to maintain only 500 nA of supply current and accuracy. The LTC2934 allows an external resistive divider to finely set the reset threshold anywhere from 1.6 to 6 V. The reset thresholds in both devices are ±1.5% accurate over temperature, which helps to minimize supply overdesign. An early warning of a threatening low-voltage condition is provided via a power-fail output. The supervisory circuits monitor V_{CC} and pull the reset output low if the supply voltage drops below the configured reset threshold. The LTC2934 includes two reset timeout periods of 15 or 200 ms, settable via a time-out selection pin, while the LTC2935 offers a fixed 200-ms timeout.

The LTC2934 starts at **$1.15** each and the LTC2935 costs **$1.22** each in 1,000-piece quantities. They are also available today in production quantities.

Linear Technology Corp.
www.linear.com



NEW PRODUCT NEWS

NATIVE HARDWARE DEBUG AND C++ COMPILER SUPPORT FOR PIC24 MCUs AND dsPIC33F DSCs

Embedded Workbench for dsPIC DSC/PIC24 now supports native hardware-level debugging through the Microchip Technology MPLAB REAL ICE in-circuit emulator system.

IAR Systems has strengthened its support for Microchip's range of 16-bit PIC24 microcontrollers and dsPIC33F digital signal controllers by adding native hardware-level debug support through the inclusion of the MPLAB REAL ICE in-circuit emulator system within its C-SPY debugger, an integral component of Embedded Workbench.

Embedded Workbench incorporates a C/C++ compiler, assembler, linker, librarian, text editor, project manager, and C-SPY Debugger in a single IDE, giving you the advantage of an uninterrupted workflow and a single toolbox in which all components integrate seamlessly.

The package also features seamless integration with visualSTATE, a graphical development tool for creating demanding event-driven embedded applications. visualSTATE generates optimized ANSI C code from state machine designs based on UML, providing advanced verification and validation utilities, and the C/C++ code it produces is compact and 100% consistent with the system design. Using the combination of visualSTATE and Embedded Workbench enables you to increase the feature richness and code quality of an application while reducing implementation complexity and improving maintainability.



Please contact IAR Systems directly for price information.

IAR Systems
www.iar.com

MODULAR TMS320F28*x* DSC DEVELOPMENT KITS

Making it easier to jump-start 32-bit-based digital power and embedded control designs, five new **Experimenter Kits** for the TMS320F28*x* digital signal controllers (DSCs) are now available. The modular kits enable the rapid prototyping of DSC-based communications infrastructure, industrial and consumer applications with interchangeable processor card modules, controlCARDs, Experimenter Kits with breadboard areas for full access to device signals, and application-specific DC/DC and AC/DC digital power development kits. To further reduce development time, each kit includes code examples and full hardware design details along with Texas Instruments's Code Composer Studio (CCStudio) 32-KB limited IDE.

By adopting the new TMS320C2000 controlCARDs, OEMs can rapidly build prototypes and even full production systems using the complete controller subsystem that comes on a small (90 mm × 25 mm) removable 100-pin DIMM form-factor card. Key analog, digital I/O signals, and JTAG test pins are available through the controlCARD interface. Only a 5-V input is needed because all other power levels are provided on the board. The first two controlCARDs are based on the TMS320F28335 floating-point DSC (running at 150 MHz with 512 KB of flash memory) and the F2808 fixed-point DSC (at 100 MHz and 128 KB of flash memory). The F28335 and F2808 controlCARD cost **$69** and **$59**, respectively**.**

Starting at **$89**, the C2000 Experimenter Kits include either the TMS320F2808 or floating-point F28335 controlCARD and a docking station. Full access to all controlCARD pins is provided by the docking station, along with a four-pin RS-232 interface, and a breadboard area adding components.

Texas Instruments, Inc.
www.ti.com



JTAG EMULATOR FOR FREESCALE ColdFire V2/V3/V4 DEVICES

The **J-Link ColdFire BDM 26**—a JTAG emulator for Freescale ColdFire V2/V3/V4 devices—is now available from SEGGER Microcontroller.

The J-Link ColdFire BDM 26 enables fast and reliable connections from the development system to your target hardware. Together with Freescale Semiconductor's CodeWarrior Development Studio or the IAR Systems YellowSuite, this emulator sets new standards for professional debugging solutions at affordable prices.

The J-Link ColdFire BDM 26 offers the big advantage of programming the embedded flash memory directly from the IDE. There is no need to exit debugging and use a different tool to program the internal flash. In addition, the J-Link ColdFire BDM 26 offers high download speeds into the target system to maximize productivity of valuable design resources.

The J-Link ColdFire BDM 26 costs **$299**.

SEGGER Microcontroller
www.segger.com

NEW PRODUCT NEWS

RTOS SUPPORT FOR STM32 FAMILY

You now have the opportunity to combine the efficiency, flexibility, and scalability of the **RTXC family** with the high-performance, quick-time-to-market STMicroelectronics STM32 family. This combination of software and hardware brings new degrees of freedom to the MCU market because it offers the best of both worlds for designers choosing between a 16- and 32-bit solution.

A broad range of embedded software solutions are available including RTOS, design tools, 10/100 Ethernet software, USB stacks, and file systems. Known for its efficiency, high performance, and small footprint, the RTXC family of solutions provides an easy-to-use, flexible solution for developers who decide to use the STM32 platform.

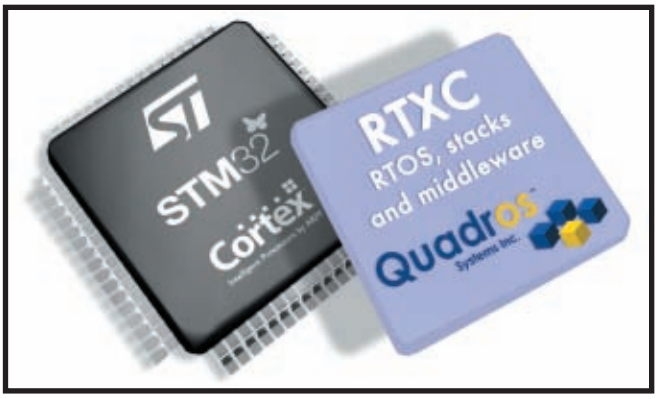
When using the STM32 family, you can select the embedded development tool that best fits your needs. RTXC products are available with support for the IAR Embedded Workbench for ARM (EWARM) and the RealView microcontroller development kit (MDK-ARM) from Keil.

The RTXC family comprises four kernel architectures offering distinct performance advantages for each of the major processing models. The RTXC/ss (single stack) utilizes a lightweight specialized executive to support high-data-rate and signal-processing applications. The RTXC/ms (multistack) is optimized for control processing using an event-driven, prioritized, preemptive scheduler.

The RTXC/dm (dual mode) marries RTXC/ms and RTXC/ss. It is ideal for convergent processing applications, which combine DSP/dataflow and RISC/control processing in a single-core processor. Both RTXC/ms and RTXC/dm are brought together under RTXC/mp (multiprocessing) in various combinations to support multicore and multiprocessor implementations.

Contact Quadros Systems for pricing.

Quadros Systems, Inc.
www.quadros.com



OFDM-BASED POWERLINE COMMUNICATION MODEM

The **MAX2990** is a new orthogonal frequency division multiplexing (OFDM)-based, power-line communication (PLC) modem. This device employs advanced broadband communication techniques to deliver cost-effective, two-way data communication over AC and DC power lines at speeds up to 100 kbps. By using existing power lines, it reduces the need for external cables to interconnect between network nodes.

The MAX2990 complies with international power-line signaling regulations, including CENELEC, FCC, and ARIB. This highly integrated SoC is ideal for applications requiring high data rates over long distances, such as automatic meter reading, energy management and load control, lighting control, and building, industrial, and home automation.

The MAX2990 uses OFDM technology with DBPSK modulation and forward-error correction to provide robust data communication in the presence of narrowband interferers, group delays, jammer signals, impulsive noise, and frequency-selective attenuations. Consequently, the MAX2990 is the industry's only broadband PLC chip that transfers data at the 10- to 490-kHz frequency range.



Advanced networking techniques ensure a reliable, highly secure communications network. Specifically, a CSMA/CA scheme controls the data traffic flow in multiple-nodes distributed networks, and an automatic repeat request function ensures the delivery and receipt of incoming packets. The MAX2990 also integrates a fast DES encryption/decryption coprocessor to enhance data security.

The MAX2990 combines the physical (PHY) and media access control (MAC) layers in a single chip that also integrates Maxim Integrated Products's 16-bit RISC MAXQ microcontroller. The MAX2990 includes 32 KB of flash memory to run the MAC code and user-defined custom applications, plus 8 KB of SRAM for data memory.

The MAX2990 starts at **$8.50** in 1,000-piece quantities.

Maxim Integrated Products, Inc.
www.maxim-ic.com

NEW PRODUCT NEWS

AUTOMOTIVE BATTERY-MONITORING TRANSDUCERS

The **HAB 60-S** is a new family of current transducers for automotive battery-monitoring applications. These transducers have been designed to measure DC, AC, or pulsed currents up to ±100 A. A new ASIC incorporated into the units offers resolutions 2.5 times better than previous models and a two-fold improvement in offset error, coupled with a significant reduction in price.

The transducers use open-loop, Hall-effect technology that simplifies both installation and servicing by removing the need to cut the cable carrying the measured current. They provide a PWM output signal proportional to the primary current being measured and operate from a unipolar 5-V supply. Temperature-measurement capability can be integrated with the addition of a fourth connection pin to the transducer package.

Output resolution of the HAB 60-S transducer is 0.03 A with a linearity of 0.2%. Electric offset error is typically 0.075 A across the temperature range from –10° to 65°C and 0.15 A across the full range from –40° to 125°C.



A water-tight housing and sealed connector provide full environmental protection in engine compartment applications. Principal applications are expected to be in the measurement of battery pack currents in electric, hybrid, and conventional vehicles. The transducers are fully certified to automotive standards.

The HAB 60-S costs **$30**.

LEM
www.lem.com

PIC10/12/16 ANSI C COMPILER

The **C PRO** is a new ANSI C compiler, with omniscient code generation (OCG), supporting the PIC10/12/16 MCU family. It achieves higher code densities and better RAM utilization than comparable compilers for the PIC16/17 without using any C extensions or assembly code. By optimizing interrupt-related contexts, the OCG technology in this new compiler reduces interrupt latency by 40% to 60%.

The compiler automatically handles memory banking without requiring special qualifiers. It optimizes the size of each pointer variable in your code based on its usage. It also eliminates the need for many nonstandard C qualifiers and compiler options, and it produces more optimal interrupt context switching code.

The C Pro compiler also integrates into the MPLAB IDE, MPLAB ICD2, and the HI-TIDE 3 IDE. It includes library source for standard libraries and sample code for I/O drivers. Also included are a macro assembler, linker, and a preprocessor.



The compiler runs on multiple platforms including Windows (including 64-bit Vista), Linux, and Mac OS X. The C Pro compiler costs **$1,495**.

HI-TECH Software
www.htsoft.com

SEGGER
Software Trials Available
JTAG (J-Link™)
+++ ARM7/9 Cortex M3 +++
▪ USB to JTAG
▪ Fast 720kb/s Download Speed
▪ Serial Wire Debug (SWD) Support
▪ Multicore Debugging Support
▪ Auto JTAG Speed Recognition
The J-Link can be coupled with a number of available software modules to fit your application needs.
J-Flash is a stand-alone application used with the J-Link to program internal and external flash devices.
J-Link RDI permits the use of the J-Link with an RDI compliant debugger.
J-Link GDB Server is a remote server for GDB.
J-Link Flash Download is a module used to download your program into flash even if your debugger does not have a flash loader.
J-Link Flash Breakpoint permits you to set an unlimited number of software breakpoints while debugging in flash.
J-Link SDK is a standard DLL that extends the full functionality of the J-Link to your proprietary application.

JTAG Isolator
We also offer a JTAG isolator which can be used to offer electrical isolation between your target hardware and the J-Link. This is essential when the development tools are not connected to the same ground as the application. It is also useful to protect the development tools from electrical spikes that often occur in some applications, such as motor control applications.

Special Offer
Includes the J-Link, J-Link GDB Server, and the J-Link Flash Download.
www.segger-us.com/ncu.html
J-Link Non-Commercial (NCU) Bundle
www.segger.com

NEW PRODUCT NEWS

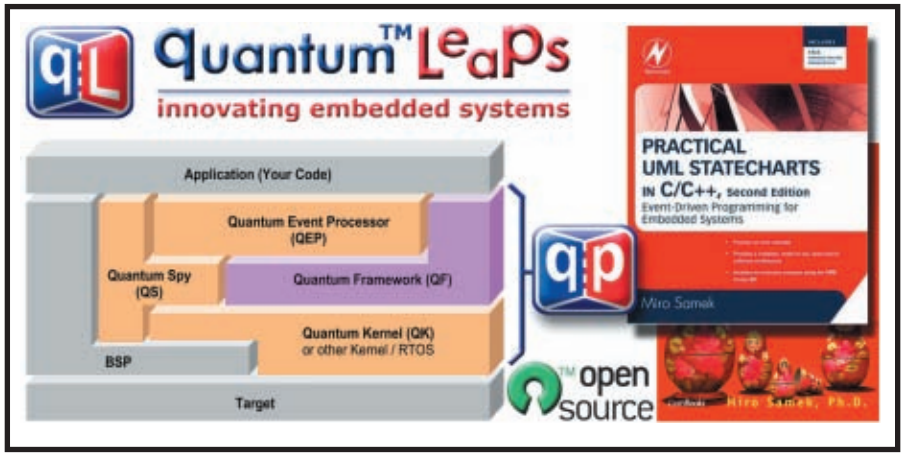
EVENT-DRIVEN FRAMEWORKS FOR EMBEDDED SYSTEMS

Quantum Platform (QP) version four enables the building of well-structured embedded applications—such as a set of concurrently executing hierarchical state machines (UML state charts)—directly in C or C++ without big tools.

QP consists of a universal UML-compliant event processor (QEP), a portable real-time framework (QF), a tiny run-to-completion kernel (QK), and software-tracing instrumentation (QS). Current versions of QP include QP/C and QP/C++, which require about 4 KB of code and a few hundred bytes of RAM, and the ultra-lightweight QP-nano, which requires only 1 to 2 KB of code and several bytes of RAM.

QP can work with or without a traditional RTOS or OS. In the simplest configuration, it can completely replace a traditional RTOS. QP includes a simple non-preemptive scheduler and a fully preemptive kernel (QK). QK is smaller and faster than most traditional preemptive kernels or RTOSs, yet offers fully deterministic, preemptive execution of embedded applications. QP can manage up to 63 concurrently executing tasks structured as state machines.

QP/C and QP/C++ can also work with a traditional OS/RTOS to take advantage of existing device drivers, communication stacks, and other middleware. QP has been ported to Linux/BSD, Windows, VxWorks, µC/OS-II, and other popular OS/RTOSs.



QP is available for immediate download under the GPL version two open-source license. Alternatively, QP can also be licensed under the terms of traditional closed-source licenses, which expressly supersede the GPL and are specifically designed for licensees interested in retaining the proprietary status of their code.

Single-product commercial licenses cost **$995**, **$1,995**, and **$2,995** for QP-nano, QP/C, and QP/C++, respectively. Product line and custom licenses are available as well.

Quantum Leaps
www.quantum-leaps.com

PROGRAMMABLE LOGIC STARTER KIT

The **MACH64 Programmable Logic Starter Kit** takes you from mystery to mastery in the black art of CPLDs. CPLDs are great starting points if you are interested in programmable logic technology. They enable you to seamlessly move into their bigger brothers, FPGAs, when you're ready.

The power of CPLDs is that with a software-based tool you can write "code" that is compiled into a hardware description that is then downloaded and flashed into the CPLD, changing its behavior to your exact specifications. By mastering this technology, you can develop your own chips that run at blazing speeds as well as design complex systems that would be impossible with discrete TTL chips.

The MACH64 kit is two kits in one. It is a complete Lattice ispMACH 4064 series development kit with a built-in programmer, which supports external targets as well. It is also a powerful educational kit that teaches CPLD technology and programming from the ground up (applicable to any CPLD).

The kit comes complete with everything you need to experiment, design, and program with CPLDs. The 250-plus page manual starts off with the technology of CPLDs and then eases you into the ABEL language used to program CPLDs. The numerous challenging hands-on projects include basic logic gates, counters, state machines, ALU design, audio generation, NTSC, and VGA video generation, and much more!

Everything you need to build all of the labs is included in the kit along with extra parts for your own creations: resistors, capacitors, LEDs, transistors, diodes, and more. Design your own chips.

The MACH64 kit costs **$159.95**.

Nurve Networks
www.nurve.net



NEW PRODUCT NEWS

HIGH-SPEED 870-nm INFRARED EMITTER WITH WIDE VIEWING ANGLE

The **TSFF5510** wide-viewing-angle infrared emitter with its uniquely designed lens has a viewing angle of ±38°, which enables significantly better performance than standard 5-mm emitters. The combination of a wide viewing angle, high-power output up to 1 A, and high speed makes the TSFF5510 infrared emitter ideal for infrared audio and video data transmission in free-air data transmission applications with high modulation frequencies or high data transmission rates.

With the industry's lowest forward voltage of 1.45 V at 100 mA and 2.15 V at 1 A, the TSFF5510 emitter provides the basis for uniform operation performance, especially in a series circuit layout. The device features switching times of 15 ns, meeting the requirements for high-modulation operation of 23 MHz and supporting data transmission rates of up to 16 Mbps.

Suitable for high-pulse-current operation, the TSFF5510 offers a peak wavelength of 870 nm, 55-mW optical power at 100 mA, and radiant intensity of 32 mW/sr at 100 mA. The device offers a compact leaded package with a lens radius of 2.35 mm, a lens height of 3.8 mm, a lens width of 4.8 mm, an epoxy bottom plane diameter of 5.9 mm, and a lead distance of 2.54 mm.



U.S. delivery in 100-piece quantities costs **$0.45**.

Vishay Intertechnology, Inc.
www.vishay.com

EVENT DATA RECORDER WITH INTEGRATED F-RAM MEMORY

The **FM6124** is the industry's first F-RAM-based event data recorder (EDR). The FM6124 is an integrated event-monitoring solution that continuously monitors state changes, stores them in the F-RAM, and alerts the system to the changes. Like a programmable logic controller, the FM6124 features simple device settings and data retrieval for easy system integration and a shorter design-in cycle.

The FM6124 is designed for broad industry use in the industrial control, medical, and metering markets. The EDR can perform in a host of application (e.g., activity/equipment/environmental monitoring, maintenance scheduling, power system management, automotive/industrial automation event recording, vehicle/pedestrian traffic counting, and surveillance systems).

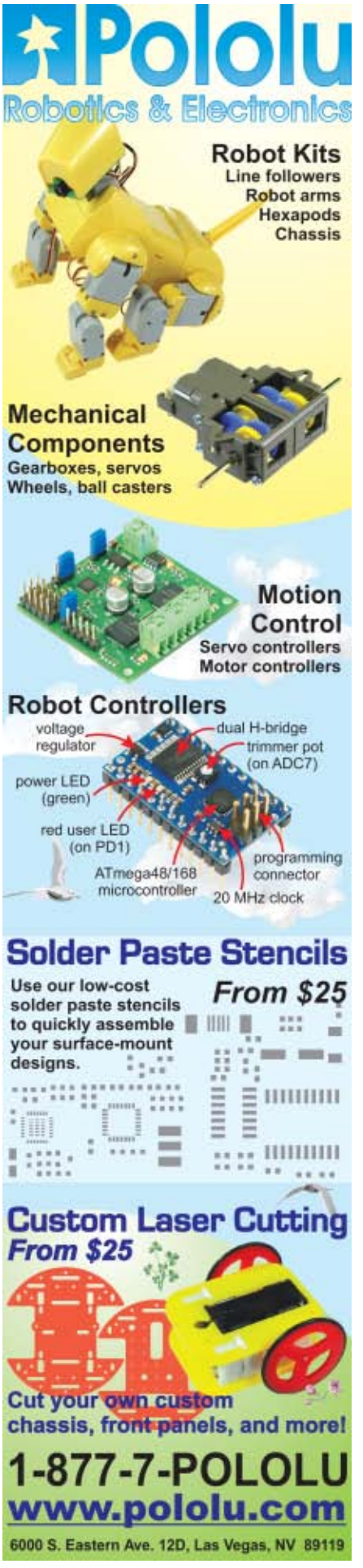
The FM6124 features 32 KB of F-RAM memory that can be used to store event records. Up to 24 KB of F-RAM can be configured to store event/user data. The on-chip RTC with a calendar enables event time stamping and can function as a system clock and calendar.

The EDR includes 12 digital inputs that can be individually configured to trigger event recording on either a rising or falling edge. The FM6124's F-RAM memory can store up to 4,000 event records. The device features an I²C interface that sustains communication speeds up to 100 kbps. The I²C interface allows for the flexible placement of the FM6124 chip (away from the host system and closer to the equipment and sensors it is monitoring). Up to four FM6124 devices can share the same I²C bus.



The FM6124 starts at **$7.50** for quantities of 1,000.

Ramtron International Corp.
www.ramtron.com



2007 WIZnet iEthernet DESIGN CONTEST
THIRD PLACE CONTEST WINNER

The DMX Portal

Obtain Lighting Control Via Ethernet

Ready to build your own DMX lighting controller? Matt's design enables him to remotely control up to 512 channels through an IP-based network or directly interface them to embedded systems with a serial connection. It is perfect for distributed lighting systems where low-cost Ethernet wiring is a better option than RS-485 wiring.

The DMX Portal is a self-contained lighting control unit that you can use to control moving lights or special effects equipped with a DMX interface (see Photo 1). The project provides a low-cost, flexible way to interface embedded systems with DMX devices, and to allow DMX control to be distributed over long distances using Ethernet.

Most programmable lighting is designed for stage performances. Standalone lighting boards are the most common playback controllers/programming interfaces used in these applications. The programming interface provided by the lighting boards is implemented via slider controls and buttons similar to an audio mixer. This type of interface is most appropriate for stage performances that change frequently or need to be adjusted on the fly. In applications such as permanent installations or automated applications, the programming doesn't need to change frequently, but the applications usually require the lighting controller to be controlled from another system instead of through physical controls. For this reason, the DMX Portal does not have a physical programming interface. Instead it offers RS-232 and Ethernet communication interfaces and two different protocols, which allow it to be used in a variety of different applications.

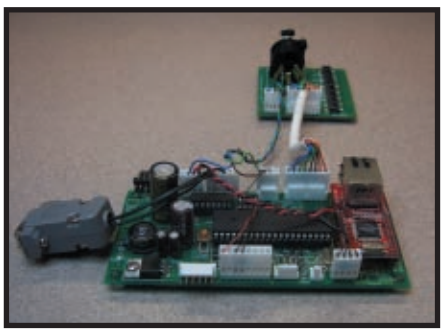


Photo 1—The DMX Portal is a compact lighting control unit. It's designed around a WIZnet 5100 Ethernet controller and a Microchip Technology PIC18F4620. The WIZnet development board is secured to the right side of the PIC development board. The five-pin XLR connector and RS-485 level converter are on the add-on board behind the PIC development board.

The system has an effects engine to automatically generate timed fades with simple commands. It also allows user-defined scenes to be saved and recalled when a command or digital trigger is received. These features enable the DMX Portal to fit into a variety of applications, ranging from a simple virtual lighting board emulated on a PC to a self-contained lighting control unit in an embedded system.

To keep project costs down, I built the DMX Portal around a Microchip Technology PIC18F4620 microcontroller and a WIZnet W5100 Ethernet interface. The complete system costs less than $50, making it extremely cost-effective in comparison to other DMX controllers.

DMX PROTOCOL

To understand the DMX Portal, it is useful to first understand how DMX works. At the highest level, DMX is nothing more than a serial transmission of 8-bit values. Data is transmitted at 250 kbps in frames that consist of the following sections: BREAK, MARK AFTER BREAK (MAB), START CODE data slot, and up to 512 channel data slots (see Figure 1).

The term BREAK means a low state where the voltage on the (+) DMX data line is lower than the voltage on the (–) DMX data line. The term MARK means a high state where the (+) line has a higher voltage than the (–) line. The START CODE and channel data slots each contain 11 bits, which are 4 µs in length. The first bit is the start bit and is always low. The next 8 bits are the data portion of the slot, with the least significant bit first. The final 2 bits are stop bits and are always high. The START CODE can have different values for the 8 data bits, but it usually contains a value of 0x00 to indicate that the following data slots represent individual channel data. Delays may be added between any of the data slots as long as the data lines remain in the high (MARK) state and the delay does not exceed 1 s. The optional delay is useful because it allows time for the processor to attend to other tasks periodically during DMX transmission. The only limit on the delay is that the next frame must be sent no more than 1.025 s after the start of the previous frame. This

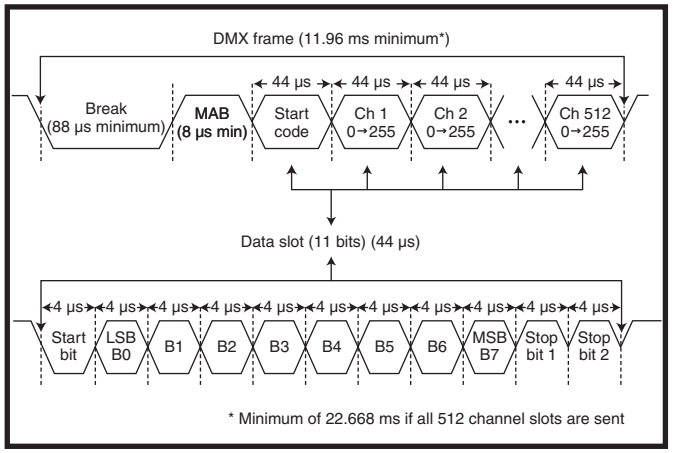


Figure 1—The DMX protocol transmits data in frames that consist of an 88-µs break, an 8-µs mark after break, and up to 513 data slots. The start code defines the type of data contained in the start slots following it, which makes the DMX protocol flexible enough to control different types of devices over a shared cable.

requirement ensures that data on the DMX line is always refreshed at a minimum rate. Many programmable lights are designed to go into a low-power state if no DMX data is received within a certain amount of time, effectively building an automatic power switch into the protocol.

The physical layer of DMX is a five-wire interface using a five-pin XLR connector. Pin 1 is Shield (ground), pin 2 is Data (–), pin 3 is Data (+), pin 4 is Optional Secondary Data (–), and pin 5 is Optional Secondary Data (+). The DMX specification requires the use of a five-pin XLR connector. The connector provides primary and secondary RS-485 data channels. The data lines (pins 2 and 3) use the RS-485 signaling specification also known as EIA-485. The secondary data channel (pins 4 and 5) is almost never used in practice and is the biggest source of differences among vendors.

Prior to the DMX512-A specification, vendors sometimes used a three-pin XLR connector and did not include pins 4 and 5 at all. Three-pin XLR connectors are commonly used for microphone connections and this practice allowed people to accidentally connect DMX and audio devices, potentially damaging equipment.

RS-485 also requires cable that has 120-Ω differential impedance. Standard microphone cable does not have this impedance, which can lead to poor signal integrity and data errors. Other manufacturers sometimes used pins 4 and 5 to deliver power, a practice that is also prohibited by the DMX512-A specification.

RS-485 was designed to be a multidrop interface that allows multiple receivers to be on a single line. The DMX512-A specification allows up to 32 receivers on a single line without buffering the signal. Each receiver may respond to one or more of the 512 DMX channels that can be transmitted on a single line. Common examples of multichannel receivers are dimmer packs with multiple AC outputs and moving lighting that uses multiple channels to control the intensity, x-axis, y-axis, and color of the light. Because DMX often involves long cable lengths and multiple receivers along the length of the cable, signal integrity is important to prevent bit errors. Using the proper impedance cable and terminating the end of the cable with a matched resistor will eliminate most signal integrity problems.

ADVANTAGES OF THE DMX PORTAL

Earlier in this article, I described the typical "lighting board" style of the DMX controller. When using this type of controller to create a light show, the parameters of the show are typically entered into the controller's memory as "scenes." A scene is basically a snapshot of the current state of all the DMX channels in use. When finished, the show consists of a large number of these scenes, which can be recalled in a timed sequence to generate changing lighting similar to flipbook animation.

The aforementioned method works well for some applications, but there are situations where it has disadvantages. One example is architectural lighting in a building such as a restaurant. Restaurants typically have many rooms and tend to keep the lights lower for dinner than they do for breakfast and lunch. You may want to design a system that uses DMX-controlled dimmers for each room and automatically dims the lights in each room at the start of dinner. You may also want the ability to adjust the lighting in each room separately to compensate for the amount of light from windows or for special events. In this situation, scenes are a poor programming method because the exact level needed for each dimmer channel is not always the same. A standard lighting board would require you to have a scene defined for each possible combination of brightness in each room. Even if you allow only five discrete brightness levels per room and had four rooms, this would result in 625 scenes to cover every possible combination. Because the DMX portal gives you a way to programatically

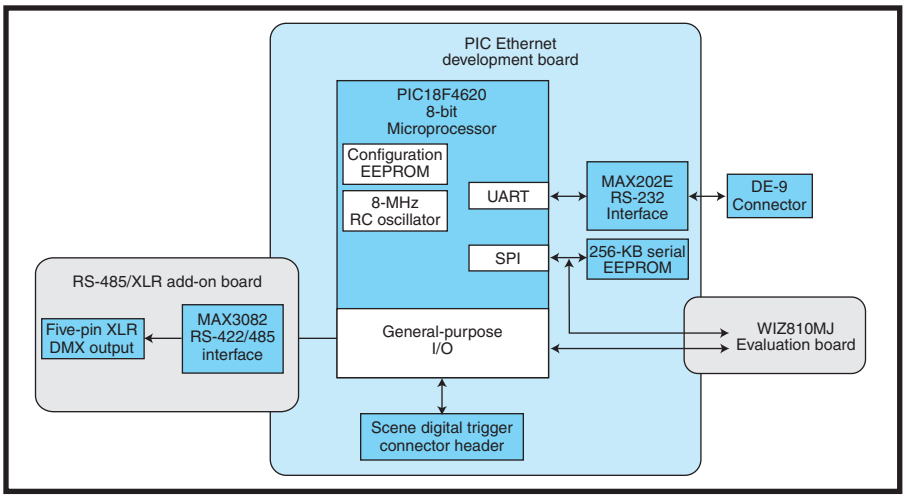


Figure 2—The DMX portal includes three boards. The PIC18F4620 microcontroller has more than enough power to handle the functionality of the DMX features, command processing, and the RS-232 interface. The WIZnet evaluation board offloads all of the processing requirements for an Ethernet interface and connects to the PIC18F4620 via the SPI port.

control the level of each channel and recall saved scenes, the task would be much simpler. A scene could be defined for the default levels of every room at the start of each meal period. If levels other than the defaults are required, the commands in the DMX Portal control protocols make it easy to modify the levels of individual channels on the fly. With this level of control, you can easily implement a system that provides many more manual brightness levels for each room without programming a ridiculous number of fixed scenes.

HARDWARE

As you can see in Photo 1, the DMX Portal prototype consists of three boards. Figure 2 shows how the boards are connected. The main board is a simple PIC development board, which contains power supplies, an RS-232 interface IC, and a 256-KB serial EEPROM. The board also has a few LEDs and headers for the PIC's debug lines and the remaining I/O pins (see Figure 3).

A PIC18F4620 microcontroller is the core of the system. It is responsible for maintaining the current state of all DMX channels, generating the DMX output stream, processing incoming commands, and generating effects like fades. This project uses many of the PIC18F4620's peripheral hardware features. The internal EEPROM is used to store user-defined settings that are applied when the system is turned on (such as whether to enable DMX output, a value to initialize all channels to, and IP settings for the Ethernet interface). The PIC18F4620's serial communication hardware is used to implement the RS-232 asynchronous interface and SPI communication to the external EEPROM and the W5100 Ethernet controller.

The RS-485/XLR add-on board is a simple PCB that contains an RS-485 interface IC to translate the 5-V CMOS output of the PIC18F4620 to a differential output meeting the RS-485 fault-tolerance specifications (see Figure 4). This board also contains a five-pin XLR connector defined by the DMX512-A specification as the proper connector for DMX interfaces and eight tactile switches I used to debug the trigger functionality of the DMX portal.

The final board is the WIZnet WIZ810MJ evaluation board, which contains the W5100 hard-wired TCP/IP stack IC. The board handles all of the low-level details of the UDP Ethernet interface for the DMX Portal. It also provides the passive components and the RJ-45 connector required for the Ethernet interface. Data is transferred to the microcontroller through the SPI port rather than the parallel interface, but this could be easily changed if high throughput via the Ethernet interface is required.

FIRMWARE

I wrote the firmware for this project completely in assembly with Microchip's MPASM compiler. Even though writing in assembly can be more time consuming and make complex algorithms more difficult to read, I had two reasons for using this language. The first was that I wanted full control to optimize the algorithms as much as possible for the PIC18F architecture and my specific application. The second reason was that I wanted this project to be usable by other people as a basis for PIC or DMX projects.

Using assembly requires no licensed

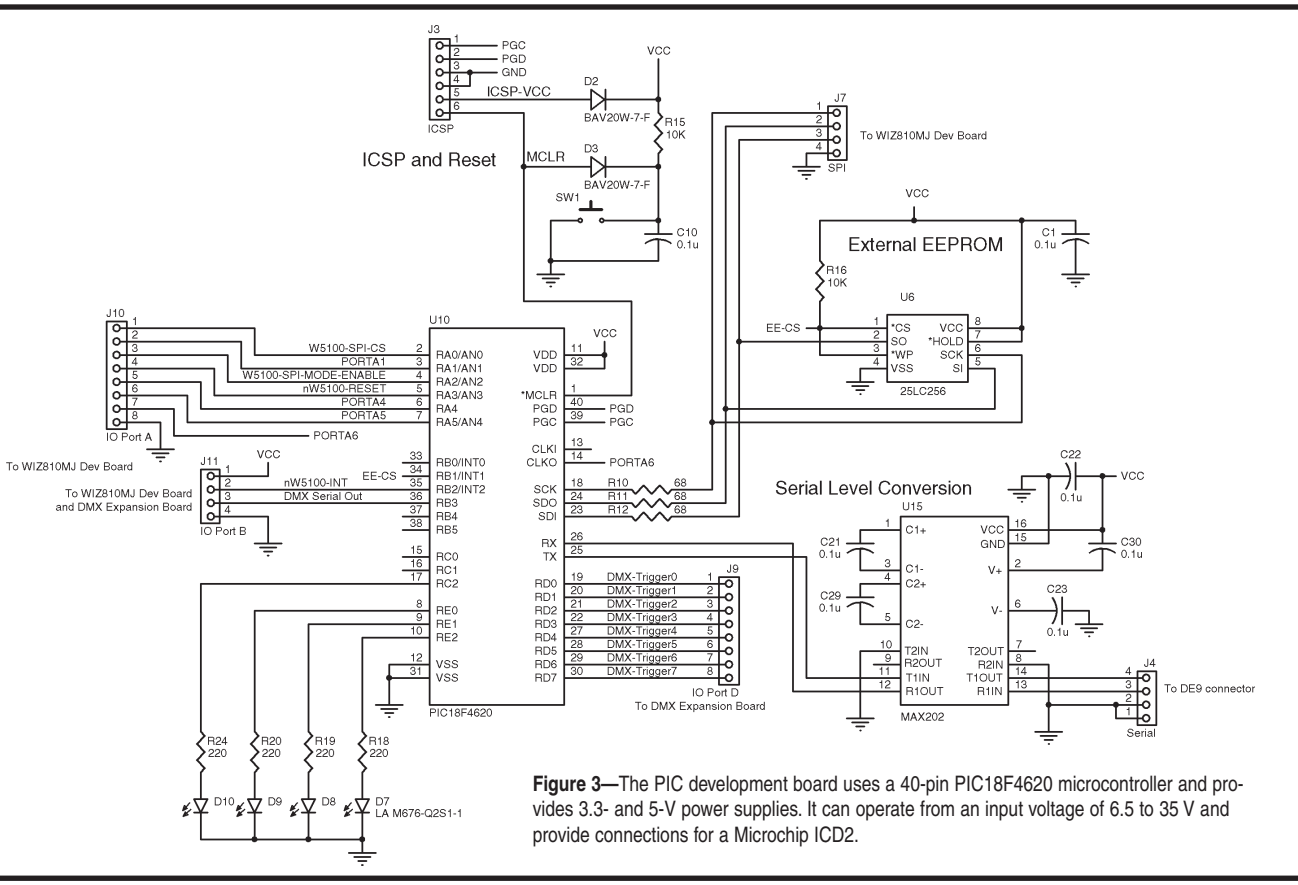


Figure 3—The PIC development board uses a 40-pin PIC18F4620 microcontroller and provides 3.3- and 5-V power supplies. It can operate from an input voltage of 6.5 to 35 V and provide connections for a Microchip ICD2.

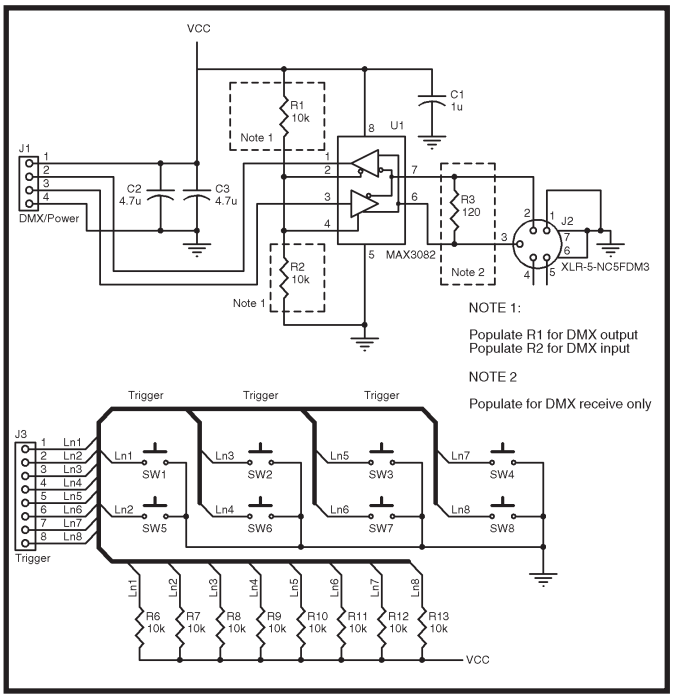


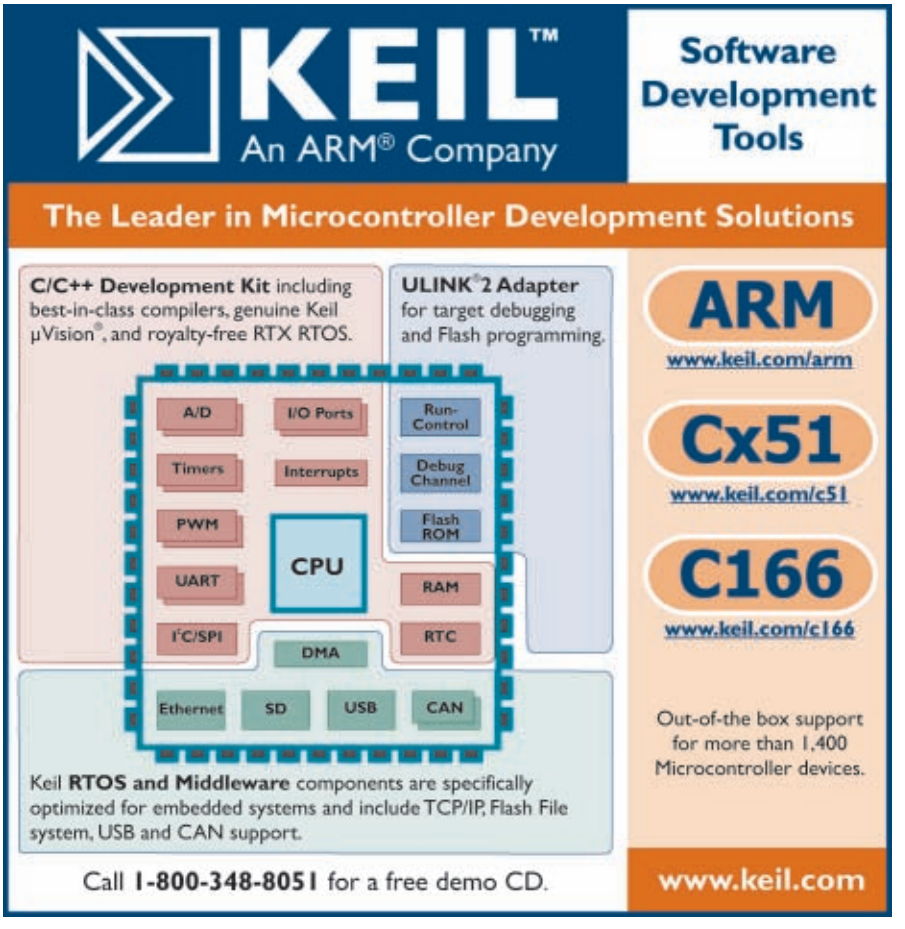
Figure 4—The DMX expansion board contains a standard DMX connector, a CMOS-to-RS-485 converter IC, and eight push button switches to debug the digital scene trigger functionality.

software compilers and facilitates a better understanding of what is going on at the hardware level. This understanding helps a lot when debugging code and also helps you write efficient code the first time. During the process of writing the firmware for the DMX portal, I developed a library of useful macros and subroutines for math, string, and utility operations. The time put into writing and debugging these code building blocks will help reduce the development time of future assembly-based projects and help bridge the readability gap between assembly and C.

Multitasking in this project is completely interrupt driven. When the device first powers up, all hardware peripherals and software variables are initialized before interrupts are enabled. The RS-232 and Ethernet interfaces have dedicated interrupts that call a process to collect the received data into a circular buffer within the PIC data memory. Once a complete command is detected inside the circular buffer, a blocking command processor routine is called that validates the command and executes the appropriate action. Because this command processor routine is blocking, it is important that the commands are executed efficiently so they do not interfere with receiving the next command or refreshing the DMX output. Writing in assembly was particularly helpful here because it was straightforward to analyze how many CPU cycles were required for each command to execute and optimize them as much as possible.

Hardware timer interrupts are used to trigger the execution of the DMX refresh routines and provide timing cues for the automatic fade engine. The DMX output refresh was designed to be a blocking process like the command processor. This simplifies dealing with commands that change the contents of the DMX frame buffer. The DMX frame buffer is a 512-byte section of the PIC18F4620 data memory that stores the current 8-bit values for each of the 512 DMX channels. Because the DMX protocol allows for delays between each data slot, it would have been possible to write the code so the DMX refresh was not blocking. This would have improved the system's ability to deal with large amounts of incoming command data during refreshes. But it would also have required care in dealing with updating the frame buffer in the middle of a refresh.

Another reason the refresh was made to be a blocking process was that it effectively gave the highest priority to the DMX output. If one of the command interfaces was flooded with incoming data, it could have been



possible to starve the DMX refresh process of execution time. If this happened, and the DMX process was not blocking, the system might not have met the minimum DMX refresh rate and some lights could have gone into Auto Shutdown mode. Different applications are likely to have differing requirements for priority given to DMX output and command processing, but you can modify the code to suit your needs.

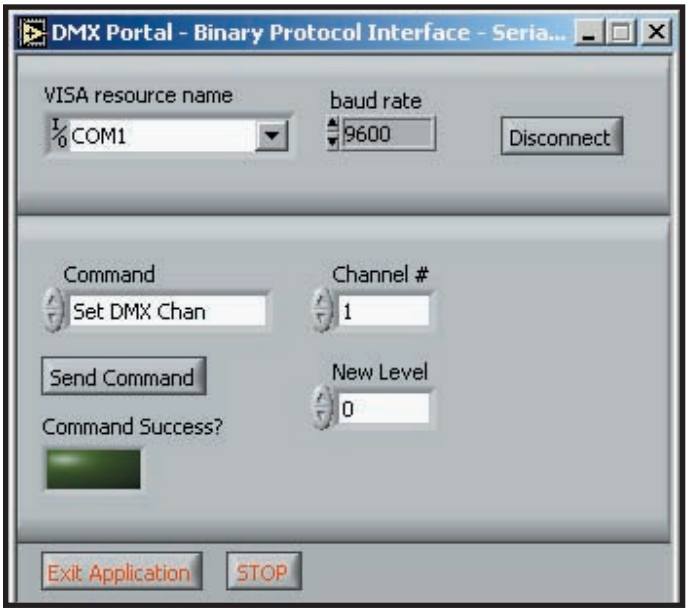


Photo 2—Using LabVIEW, it is easy to create utilities that communicate over standard PC I/O ports and have professional-looking graphical user interfaces. This utility allows the DMX Portal to be accessed through the RS-232 interface using the binary protocol.

WORK WITH THE W5100

The W5100 was a good fit for this project because implementing the software required for a simple TCP/IP stack with assembly would have been a time-consuming task. The W5100 handled all of the details and required me to write only a few subroutines to read from the device and calculate offsets into the chip's buffer memory to find my data. Some of these calculations required 16-bit math, which is not natively supported by the PIC18F family. Luckily, I had already written a library of basic 16-bit math functions for use with the command processor.

The firmware stores all of the configuration parameters—such as the IP address, gateway, subnet mask, and MAC address—in the internal EEPROM so they can be used to configure the W5100 during power-up. These parameters can be modified via the ASCII command protocol and will be automatically saved to the EEPROM each time they are changed.

COMMAND INTERFACES

The DMX Portal provides Ethernet and RS-232 interfaces so it can be connected to a variety of systems. The Ethernet interface is useful if the system into which you want to integrate DMX functionality has Ethernet connectivity, or if it will be located a long distance away from the DMX portal. The RS-232 interface is useful for connecting to embedded systems because they frequently have serial output capabilities. Plus, the RS-232 interface offers a reliable, low-cost connectivity option with little software overhead.

Each of the command interfaces supports two different command protocols. The first protocol is ASCII text-based and provides a large set of easily readable commands to control every aspect of the DMX portal. This protocol is most useful if you want to control the DMX Portal through a standard terminal program. Because the commands are text-based, more processing overhead is required to transmit and process commands using this protocol.

The second protocol is a compact binary code based on a protocol developed by the open-source USB DMX project. (Refer to the Resources section of this article.) This protocol reduces overhead to a minimum and has all of the commands required to control the DMX output. It is useful if you have configured the non-DMX parameters of the device using the ASCII protocol and want to make changes to the DMX output as quickly as possible. A command can be sent that will switch between the two protocols while the device is running to allow flexibility between available commands and communication efficiency. Another advantage is that the USB DMX protocol allows the DMX Portal to be controlled by software that supports the USB DMX protocol. I'll cover one such application, named FreeStyler, in the next section.

SOFTWARE INTERFACES

Because the DMX Portal has many combinations of communication interfaces and protocols, there is no ready-made application that can easily interface with it in all of the possible modes. I used the LabVIEW graphical programming language to build custom interfaces that could control the DMX Portal via the RS-232 or Ethernet connections using the ASCII or fast binary protocols (see Photos 2 and 3).

Programming with LabVIEW was helpful because it has libraries for communicating via a serial port or TCP/IP. It also has many examples that can be easily modified. The other benefit is that LabVIEW enables you to build a functional GUI with little work. The code for the utility interfaces I wrote could easily be enhanced to behave like the final user interface of a project.

To demonstrate how to make a final user application, I wrote a virtual lighting board application based on the same code I used for my utility interfaces (see Photo 4). The virtual interface mimics the user interface provided by lighting boards by providing sliders that can be attached to sets of DMX channels. Buttons for other common features—such as temporarily setting all lights to off, also known as a blackout, or disabling the DMX output to enable the lights to go into Power Down mode—are also provided.

Earlier, I mentioned that the fast binary protocol was based on the protocol of the USB DMX project. A major reason for this decision was that there is a free lighting control program available called FreeStyler that provides excellent

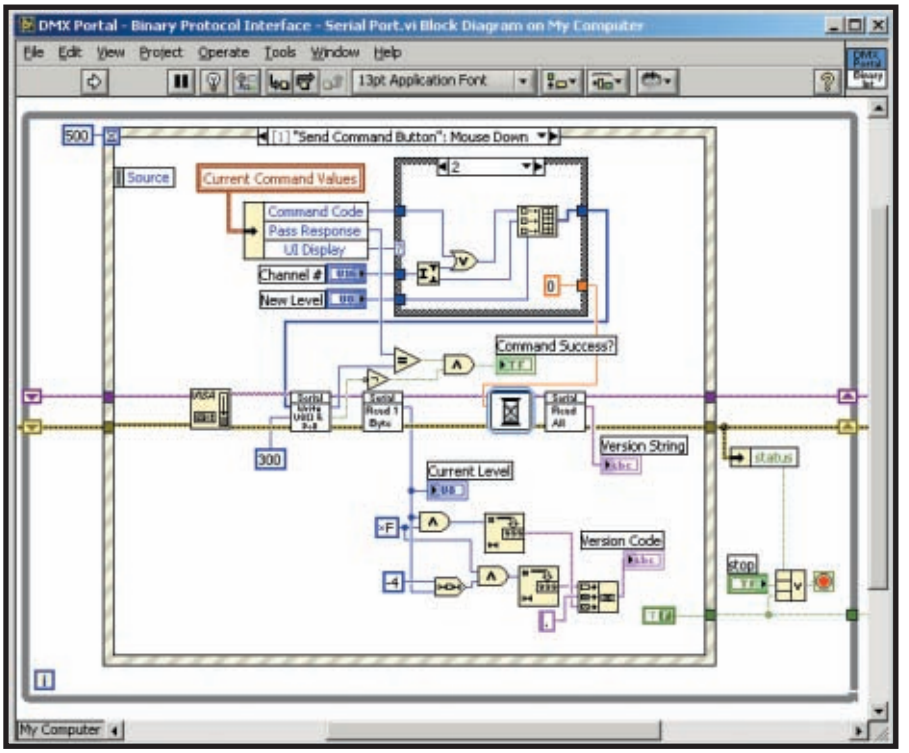


Photo 3—The code in LabVIEW is written graphically using a programming principle known as data flow. Each block in the diagram represents a function that will execute once data has arrived on each of its inputs. Outputs of a block are connected via wires to the inputs of other blocks or they are connected to control and indicator symbols, which correspond to graphical items on the GUI called the front panel.

support for many commercial programmable lighting fixtures and makes it easy to generate complex lighting effects. (Refer to the Sources section.) FreeStyler already supports the USB DMX binary protocol, so by allowing the DMX portal to support the same commands, I effectively received support for this application with no extra work. The USB DMX project uses an FTDI chip that appears as a virtual COM port in Windows.

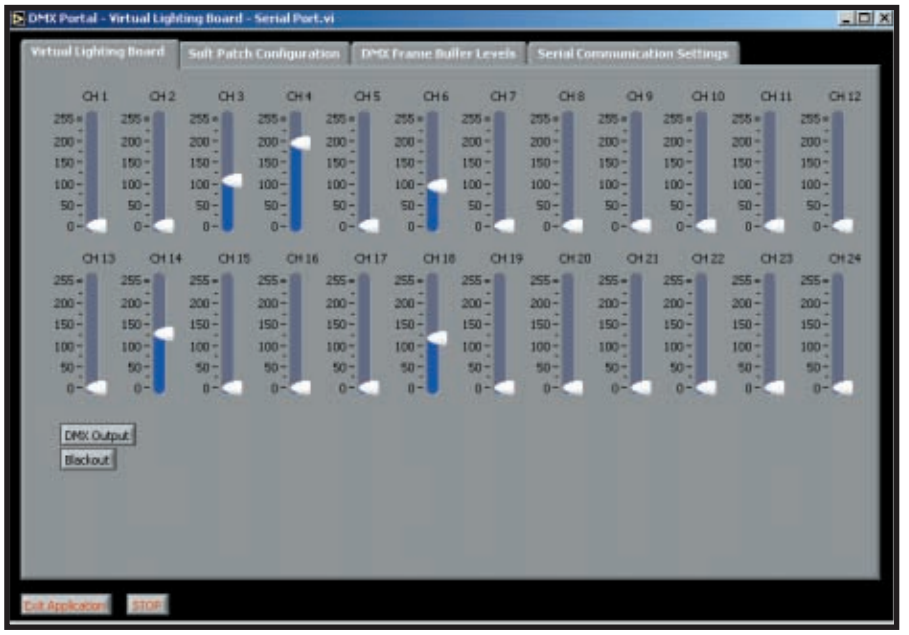
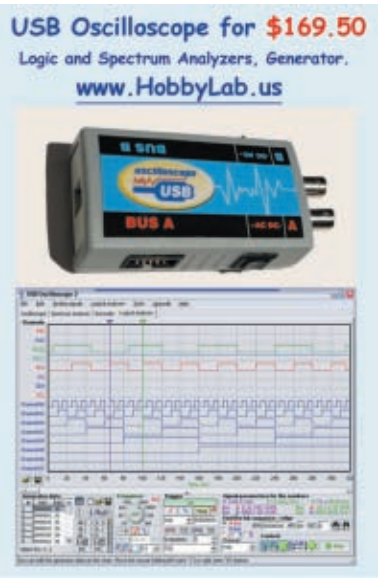


Photo 4—This application was written to demonstrate how a final user interface for the DMX Portal could be written using LabVIEW. It provides features commonly found on a standard lighting control board, such as level sliders and blackout controls.







All you need to do is put the DMX Portal into Binary Protocol mode and then point FreeStyler to the correct COM port.

FUTURE DEVELOPMENTS

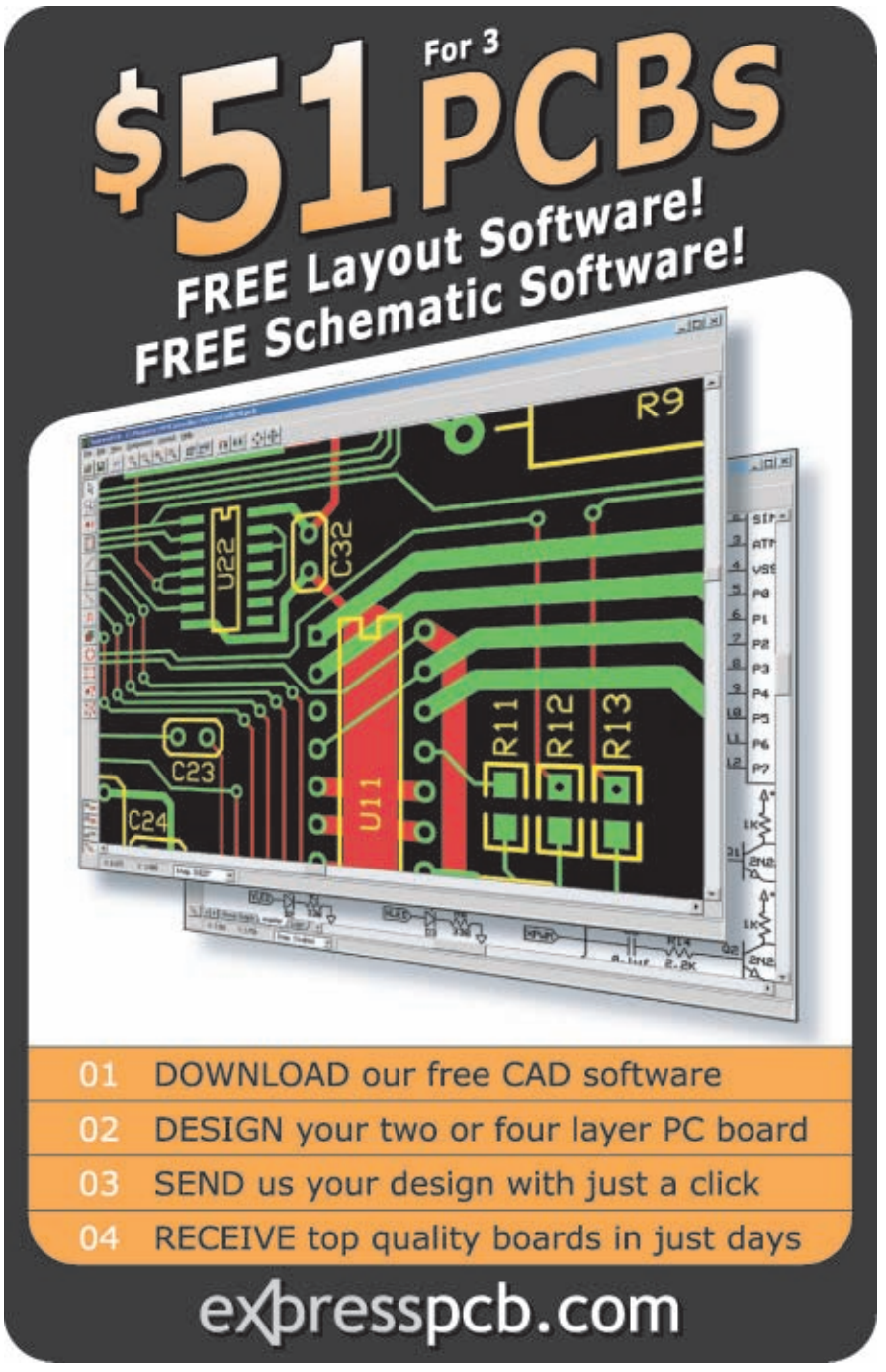
Like any project, there is always room for improvement. I would like to improve the DMX Portal's RS-232 hardware buffer. The PIC18F4620 provides a 2-byte hardware buffer to help give the processor time to process incoming data if it can't get to it immediately. As I discussed in the Firmware section, I chose to make some processes block the execution of other interrupt-driven processes to deal with time-sensitive requirements or race conditions involving modifying memory in the middle of these tasks.

There is a downside to this approach. When the DMX output is enabled, the processor is fully occupied for 22.668 ms while a frame is sent. Because the DMX output refresh consists mostly of wait commands to generate the proper data rate timing, speeding up the processor would not reduce the amount of dead time where the processor could not handle incoming data. Even at the slow data rate of 9,600 bps, a single byte takes only about 1 ms to transmit. So, it is possible to send more than 2 bytes of data over the RS-232 link before one DMX frame refresh is completed. This can result in data being lost and commands that were sent not being processed.

The best way to maintain a constant DMX output refresh rate while still being able to tolerate large momentary bursts of command data is to implement the serial interface with a second small low-cost microcontroller. This would enable the data memory of the second microcontroller to act as a large command buffer. It would also make the serial interface as robust as the W5100 Ethernet interface, which already contains a large memory buffer.

Another approach would be to use an external hardware UART for the DMX output. This would eliminate the need for the processor to sit in wait loops to generate the correct data rate. It would also enable the processor to handle incoming data in the gaps between bytes. The downside to this approach would be that the DMX output refresh rate would become dependent on the amount of traffic received on the communication ports. This may not be a problem, but it would require a bit more care in implementation because the DMX standard requires a minimum refresh rate. If too many commands are sent during a frame refresh, this rate may not be met.

For the prototype, I used a relatively slow 8-MHz system clock because it could be generated from the PIC18F4620's internal RC oscillator. There is no reason why I can't run the PIC18F4620 faster with an external oscillator. Doing so would help improve the performance of some tasks such as command processing. Having a faster clock to reduce the processing time would help make the system run much smoother without



costing too much.

One improvement that could be made to the Ethernet interface would be to change from the UDP protocol to the TCP protocol. UDP has no mechanisms to guarantee packets are not lost. It also has a maximum payload size that can be a problem with some of my commands that generate large amounts of response data. A TCP-based link would automatically try to resend dropped packets, reorder packets that are received out of sequence, and detect a communications failure. TCP also behaves as a constant datastream, so it would not be subject to a maximum payload length like UDP. The W5100 supports TCP. Thus, it would not be difficult to make this transition. I think it would greatly improve the quality of the communication interface on the DMX Portal.

I also want to improve the data storage space for user-defined scenes. For simplicity, I chose to use a serial EEPROM for my storage, but it offers only limited space. An SD card would have been a much better choice because it offers a low-cost storage solution that easily holds an entire show's worth of scene data. Because SD cards can be accessed through a SPI just like the EEPROM, supporting this feature would require only adding code to handle the FAT file system. An additional benefit to the SD card approach would be that scene data could easily be written or backed up to a PC without actually being connected to the DMX Portal. It is possible to write a software interface that performs these functions using the available communication interfaces, but it is not as simple as just reading a file from an SD card.

Matt Ernst (mbernst@gmail.com) is a graduate of the University of Wisconsin-Madison School of Engineering. He has a strong interest in the automation and control systems used in the entertainment industries. Matt is a staff analog hardware engineer at National Instruments. He designs high-speed test and measurement hardware.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

RESOURCES

FreeStyler, http://users.pandora.be/freestylerdmx/.

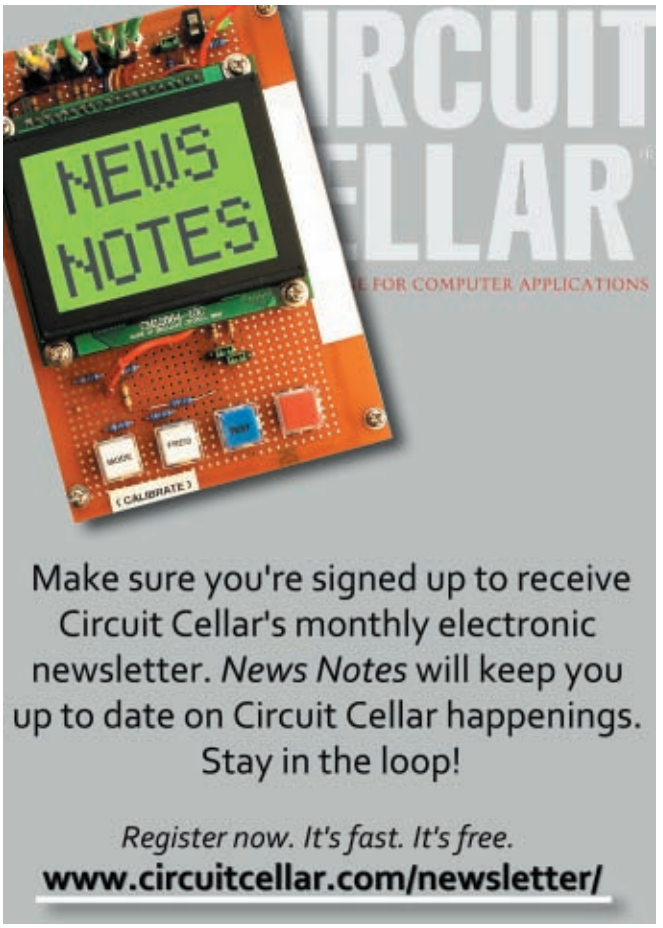
B. Suffolk, "USB DMX Project," www.usbdmx.com/protocol.html.

United States Institute for Theatre Technology, Inc., "DMX512-A Specification," www.usitt.org.

SOURCES

MPLAB IDE and PIC18F4620 Microcontroller
Microchip Technology, Inc.
www.microchip.com

W5100 Ethernet controller and WIZ810MJ evaluation board
WIZnet, Inc.
www.wiznet.co.kr/en





Electric Vehicle Inverter Design

Build A System For Powering AC Induction Motors

EFFECTIVE USE OF MOTOR CONTROL RESOURCES



The Electric Vehicle (EV) Inverter project is a starting point for the conversion of a vehicle with a gas engine to one with an electric motor. The inverter was built using a control board and software based on Microchip Technology's MC-1 development board. The finished design can power a wide range of AC induction motors.

by Dan Hall, Tristan Kasmer, Doug Krahn, Adam McIntyre, and Dena Ponech

The world is in the process of advancing alternative sources of energy, and the concept of electric vehicle (EV) conversion is becoming more and more popular. But anyone interested in this concept will discover that the process of converting a gas-powered vehicle to electric power is fairly cost-prohibitive. This challenge led instructors in the Electronics Engineering Technology program at Camosun College in Victoria, BC, to pitch the idea of an "open-source" EV conversion kit to our class to take on as our final project. We accepted and the EV Drive Team was formed.

The open-source concept of an EV design has numerous benefits, such as accessibility, cost, and advancement. You should be able to easily obtain all of the inverter's components (if not purchase an assembled system), and once constructed, it will be able to power a wide range of AC induction motors (ACIMs). Obtaining a typical three-phase high-power inverter for driving an ACIM can cost between $8,000 and $25,000. In this article, we will present a cost-effective method of applying an alternative source of energy. Any technically minded person should be able to complete this project for around $2,500.

Our inverter was constructed with a control board and software based on Microchip Technology's MC-1 development board and three dual-IGBT modules with gate driver boards from Powerex. We used these development tools to build a basic 100-kW three-phase inverter in an open-source process, which can be continually improved on.

All of the inverter's components of the inverter design are "off-the-shelf" units readily available at several electronics distributors. The parts are subject to change if the open-source design is modified to suit a user's needs or improved by advancing technology. This aspect of the inverter's design provides for the non-restrictive application of the system. It gives you the freedom to choose a motor that was not originally designed for a vehicle. For example, we used an ACIM that was intended to drive an elevator.

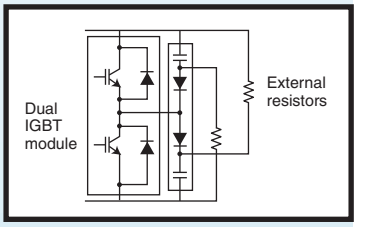


Figure 1—This is a Powerex CM400DU-12F module with an RCD snubber circuit.

The ACIM is the simplest and most rugged electric motor. It consists of two basic electrical assemblies: the wound stator and the rotor assembly. The outer stationary "stator" consists of coils that are supplied with an AC current to produce a rotating magnetic field. The inner non-stationary "rotor" revolves as a result of the torque that is created by the rotating magnetic field. The induction AC motor derives its name from currents flowing in the rotor that are induced by alternating

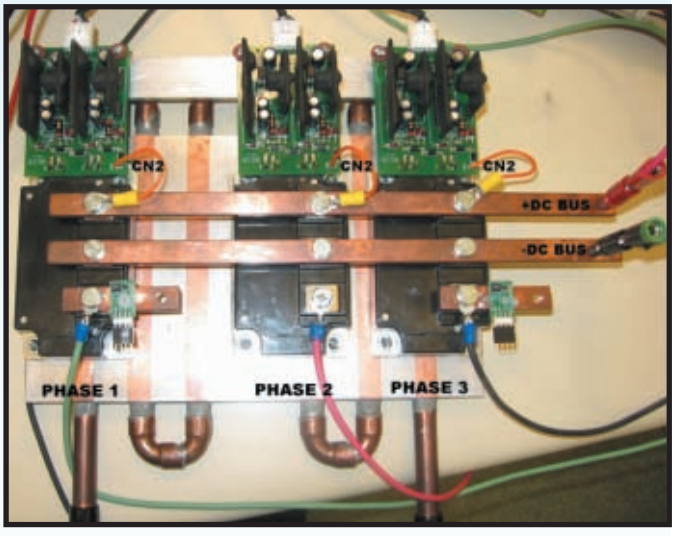


Photo 1—This photo shows the IGBT and gate driver assembly during the testing phase of development. Hall-effect current sensors are mounted on phase 1 and 3. The snubber board (not shown) is installed on top of this circuit.

currents flowing in the stator.

You can control the speed of an ACIM by varying the frequency and amplitude of the drive voltage. Early configurations of drives used SCRs fired at the appropriate times to create an unsophisticated sinusoidal input waveform. As the semiconductor industry evolved, SCRs were replaced with MOSFET or IGBT devices, which are more efficient than the SCR, and could be switched at higher frequencies to continuously generate variable-drive voltages and currents that closely resemble a sinusoidal waveform.

The AC current that is supplied to the stator, which produces the rotating magnetic field, can be controlled by a PWM algorithm that directly influences the speed of the rotating field and the output RPM of the rotor. We used a PWM technique known as space vector modulation.

POWER INVERTER

Powerex CM400DU-12F insulated gate bipolar transistor (IGBT) modules and their companion BG2B gate-driver circuits provide a relatively low-cost method for generating three-phase AC power from DC power.[1] The DC power from the EV battery pack is converted to three-phase AC to drive an AC induction motor. We configured each phase of the IGBT module design with RCD snubber protection (see Figure 1).[2] The gate driver circuit can be seen in the application note for the BG2B universal gate drive board from Powerex using two VLA106-15242 DC/DC converters and two VLA503-01 gate drivers.[3] The gate driver boards are recommended by Powerex for use with the dual-IGBT modules and provide 2,500 VRMS of control signal isolation via high-speed optocouplers and desaturation detection to prevent short-circuit conditions on the IGBTs. The CM400DU-12F modules are rated with a collector-emitter voltage of 600 V and a continuous emitter current of 400 A with a peak rating of 800 A. The modules are able to provide switching speeds of up to 30 kHz, we used 20 kHz. The frequency range promotes efficient operation of the spatial vector modulation (SVM) algorithms used to drive the IGBTs and bring the switching noise out of audible range.

Because the IGBT modules are capable of high switching speeds and operate at extreme power levels, transient voltage and current protection are important. An RCD snubber circuit was chosen for its ability to limit peak voltages and reduce total circuit losses, including switching and snubber losses. Low ESR and low self-inductance rated components are the heart of the protective design, eliminating the parasitic and residual inductances that can occur across the IGBT's switches and across the DC bus.

The inverter can generate approximately 100 kW, so it requires an effective cooling system. The initial cooling system chosen was based on a liquid coldplate from D6 Industries that costs $180. The cooling system can be configured as a stand-alone unit with its own pump and radiator or as part of the vehicle's heating and cooling system. The packaging of the IGBT modules enables all three to be mounted directly on a single coldplate for simple and effective thermal management. The gate driver circuits for the inverter are supported by an aluminum bar (see Photo 1). This cooling configuration is mounted to the inverter enclosure for a greater heat dissipation area.

GATE DRIVERS

Gate driver circuits are required to control the IGBTs. These circuits provide appropriate firing and off-time voltages to the gate-emitter connection. They also provide optical isolation of control signals from the high voltages that are being controlled. We chose the off-the-shelf gate driver assembly BG2B-5015 kit from Powerex.[3]

To produce the firing voltages, the circuit employs an isolated DC/DC converter module represented by the DC/DC converter module block diagram (see Figure 2).[4] The DC/DC converter enables the driver board to be run off of a single DC supply that takes 15 V and converts it to an isolated 24 VDC. This voltage is then placed across a resistor and Zener diode network that provides isolated 15.8 V at V_{CC}, pin 3, and –8.2 V at V_{EE}, pin 1, with respect to the zero reference point V_{E1}, pin 2. The transformer in the DC/DC converter also provides 2,500-VRMS protection to the gate driver assembly module.

The DC/DC converter provides an isolated power output to pins 4 and 6 on a hybrid IC IGBT gate driver represented by the gate driver assembly module block diagram (see Figure 2).[5] The gate driver assembly uses the control input at pin 13 and 14 to bias the gate-emitter connection in the gate driver module appropriately to turn the IGBT on or off. The 15.8-V signal at V_O, pin 5, ensures that the IGBT will be saturated in the on state. The –8.2-V signal ensures that the IGBT will not be switched on by noise or Miller capacitance between the collector and the gate. The output stage of this assembly provides high current to the gate so proper

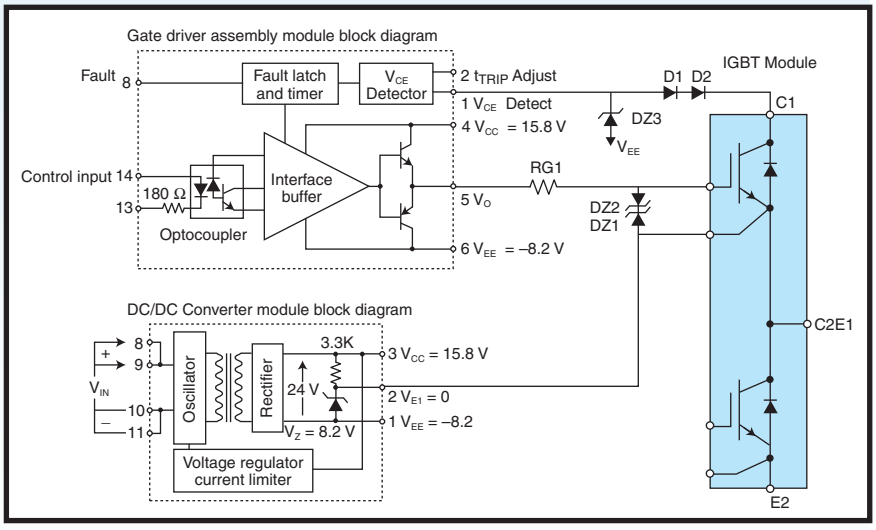


Figure 2—These are gate driver board circuit connections to one IGBT. Bottom transistor connections are removed for simplicity.

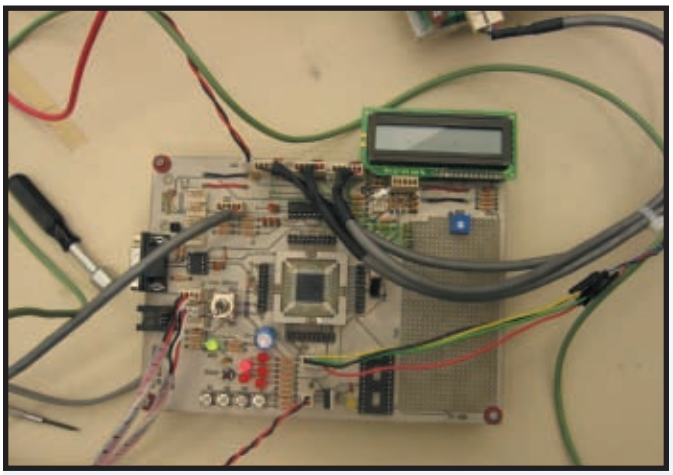


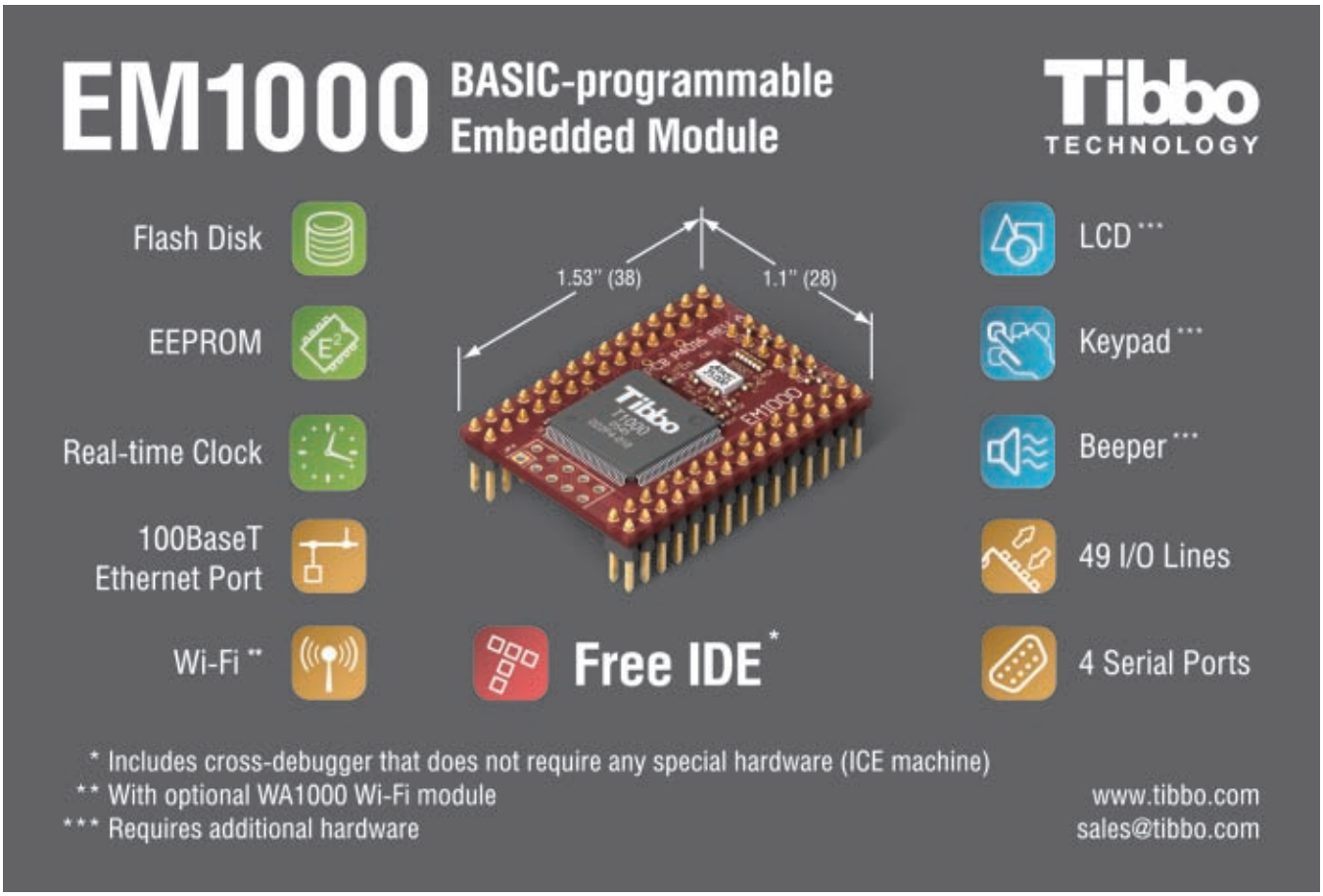
Photo 2—The motor control board is shown here. We based our design on a Microchip Technology dsPIC MC1 motor controller board.

control of load currents can be obtained. As the current passing through the IGBT increases, the amount of gate current must also increase to keep the IGBT saturated. The gate driver assemblies we used for our inverter project are capable of driving an IGBT that can switch up to 400 A. The gate current required to control this amount of current is approximately 5 A DC. The gate driver assembly also provides optical isolation of control signals via high-speed open-collector optocouplers that protect up to 2,500 VRMS. This ensures that the control signals, pins 13 and 14, and high-voltage signals are well insulated from one another, thus protecting the microcontroller that is producing the inverter drive logic.

CONTROL BOARD

The inverter's control board is primarily a stripped-down variant of the Microchip MC-1 development board (see Photo 2).[6] (An alternate version of Photo 2 with call-outs is posted on the *Circuit Cellar* FTP site.) We did this for code compatibility and to reduce actual development time. Several simple changes were made to the PCB design. The bias resistors on the gate drivers were moved to the output of the logic buffer and given the capacity to function as pull-up, pull-down, or float, depending on the placement of a jumper. Regulators were added to the board, providing stable 5- and 15-V sources. Additional decoupling capacitors were placed in the circuit to reduce the inevitable noise (motor noise and switching noise) that is inherent to the system. The complete control board is shown in Figure 3.

We revised the control board four times. The first board was tight, with limited room for additional prototyped circuitry. This would be excellent for a production model but a nuisance for development. Further revisions increased the board's size and spaced out the components. There is now more real estate for a larger prototype area, allowing for circuit changes if required. The final board includes all of the changes. The schematic is available on the *Circuit Cellar* FTP site.



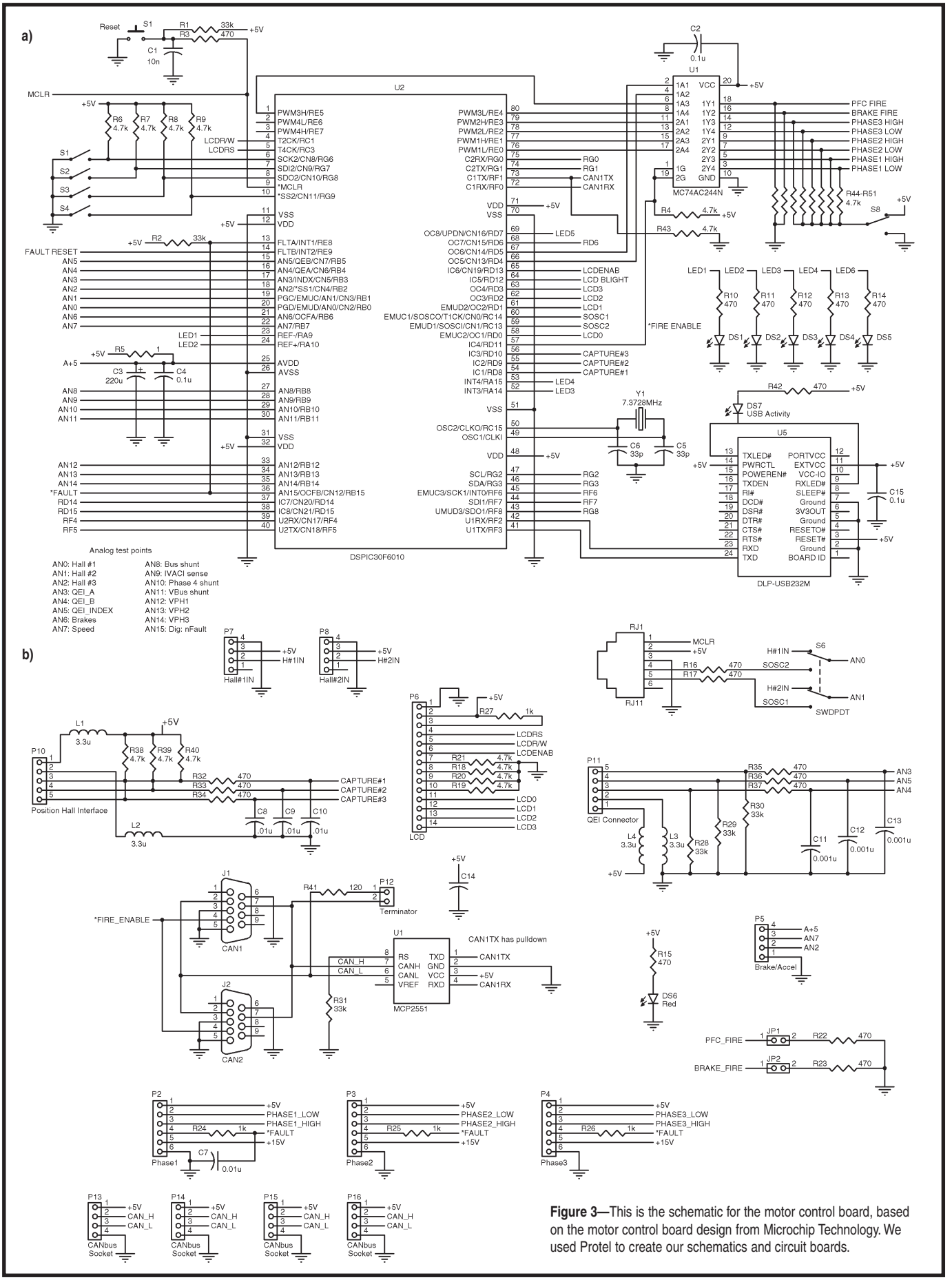


Figure 3—This is the schematic for the motor control board, based on the motor control board design from Microchip Technology. We used Protel to create our schematics and circuit boards.

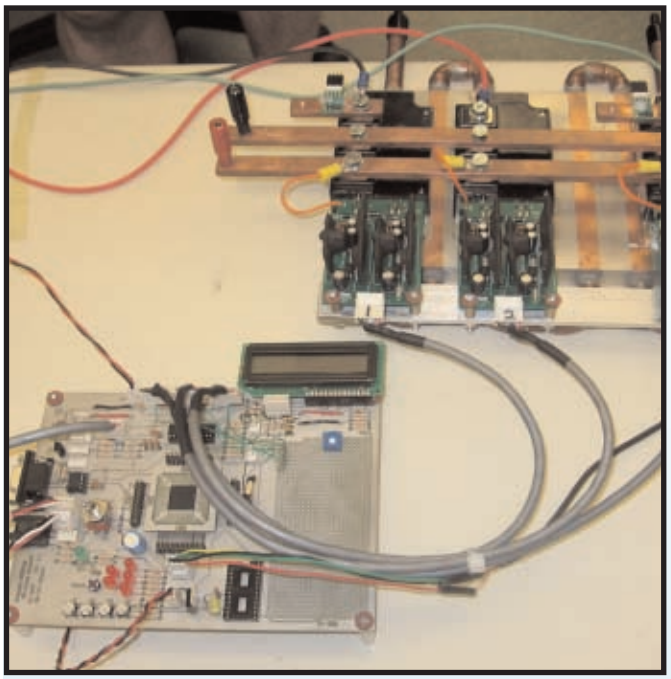


Photo 3—This shows the interconnections between the motor control board and the gate drivers on the inverter. Shielded cable was used to reduce EMI.

The most significant difficult aspect of the board design process involved trying to anticipate the needs of future users. The board was given more CAN bus connectors than the standard MC1 board. We included two DE-9 connectors instead of one, as well as four custom sockets for additional CAN bus modules. The final connection between the control board and the inverter is shown in Photo 3.

A daughter board was created to assist in debugging as well as provide an interface to the LabVIEW application used for motor tuning and instrumentation (see Photo 4). (An alternate version of Photo 4 with callouts is posted on the *Circuit Cellar* FTP site.) The daughter board acts as a bridge between the CAN bus and the USB with robust error checking between the two protocols. The daughter board microcontroller and the CAN bus controller were left with separate clocks due to the timing sensitivity of the CAN bus. With some minor modifications, the module could also control an optional LCD. The module connects to the main board with a four-pin connector that also provides power to the module. Four of these sockets are provided on the main board.

SPACE VECTOR MODULATION

AC induction motor (ACIM) control can be accomplished using many techniques. For our project, we chose a PWM technique known as space vector modulation (SVM). To help illustrate the advantages of SVM, a quick review of some of the other techniques for motor control is in order.

The simplest method of motor control is the square-wave, or six-step inverter. The method is not processor intensive. It uses simple circuitry but does not simulate a sine wave effectively. This waveform generates significant total harmonic distortion (THD) that makes the system power inefficient. Energy is lost as heat, vibration, and noise. A multistep inverter can be created that can generate more than six voltage steps in the waveform. This method simulates a sine wave more closely, but requires much more complicated circuitry, additional switches, and transformers. The resulting control is better than the six-step inverter, but still generates an enough harmonic distortion to make it inefficient. Next, there are PWM techniques. These techniques improve the simulation of sine waves and therefore reduce the THD. There are many PWM classifications. The two discussed here will be sinusoidal PWM, specifically the volts-hertz (V/Hz) method, and space vector modulation (SVM).

ACIM speed level and torque level can be controlled by varying the voltage and frequency supplied to the motor. The V/Hz method uses a constant, K, to relate the desired drive voltage to the choice of input frequency with the calculation:

$$K = \frac{V}{Hz}$$

Because the ACIM is inductive and if the input frequency is decreased, the stator currents will increase. The input voltage will have to decrease by a proportional amount to counteract the current increase. The result is in a relatively constant stator field:

$$\text{Voltage} = K \times \text{Frequency}$$

This is a simplistic representation of the V/Hz profile. In reality, the relationship between voltage and frequency does not have to be linear, and may be adjusted to provide optimal motor performance in some frequency ranges.

The V/Hz method also incorporates feedback from the motor to more efficiently generate drive demand using PID control. The actual frequency of the motor is measured using a tachometer that is frequently used to calculate an error signal that is used by the V/Hz profile to generate the appropriate voltage and frequency values. Finally, a PWM code will vary the duty cycle with respect to time to generate a simulated sinusoidal drive signal.

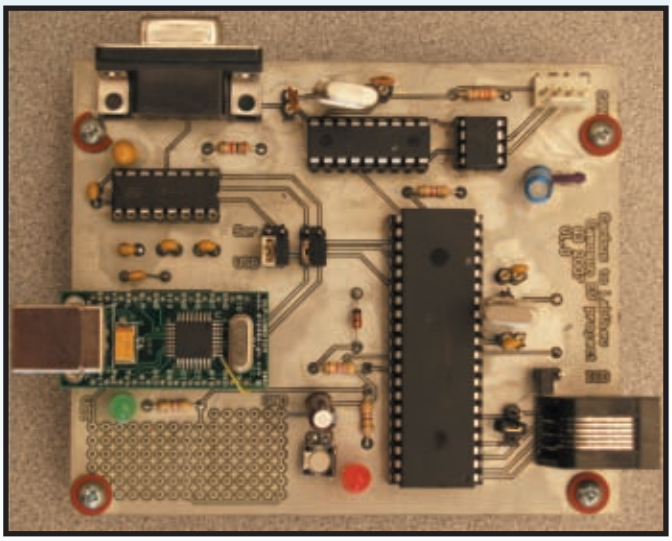


Photo 4—This daughter board was used to interface the HMI to the control board. The control board is connected via a CAN bus and the LabVIEW HMI via USB.

The drawback of the V/Hz control is that it does not handle fast dynamic changes in speed or load well. The reason for this is that control of the torque and rotor currents cannot be separated, and change in drive voltage affects both of these motor parameters. To be able to control the torque independently, control of the phase of the input voltage is required. This is realized with space vector modulation (SVM).[7]

SVM uses mathematical coordinate system transforms to simplify motor control by making the stator phase currents appear as DC values under steady state conditions. The two transforms used are the Clarke and Park Transforms. First, two of the phase currents are measured and used as inputs to the mathematical model. The third phase does not need to be known because the sum of the three phase currents should be zero. The currents can be represented as time-varying vectors in a three-axis, 120° separated coordinate system. To simplify this, the Clarke Transform represents the three phases as two time-varying vector components thus, reducing the coordinate system to a two-axis system. The Park Transform further simplifies by changing the stationary coordinate system with time-varying vectors to a rotating coordinate system with stationary vectors, thus, representing the AC currents as DC value with respect to the axes. The rotating reference plane is required to spin at the speed of the stator current, which is synchronous with the rotating magnetic field of the stator. These quantities are spinning slightly faster than the rotor. This is known as slip and is required to induce a magnetic field in the rotor. With the motor's time constant, usually given by the manufacturer, the slip frequency can be calculated. Then, when compared to motor velocity, the slip frequency will give an angle that is used to align the rotating plane with the stator's current value. The last part of the transform process is to invert the calculated transform parameters back to values that the switching circuitry can use. This generates the PWM code.

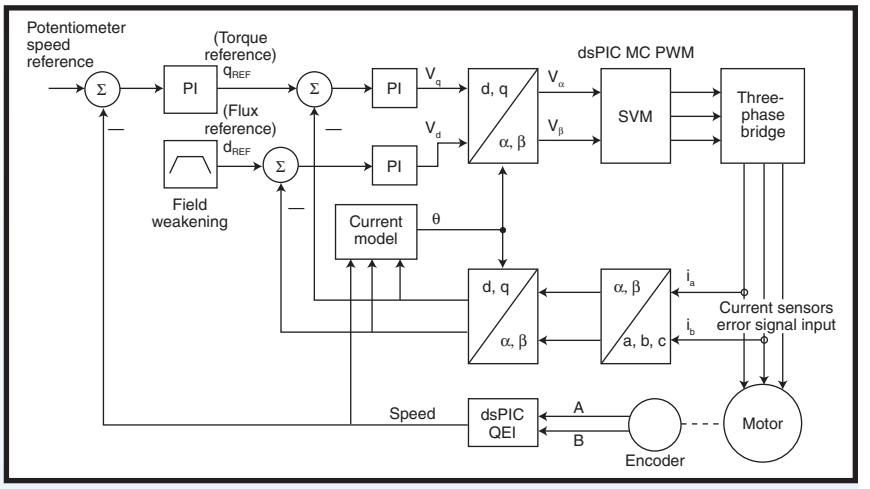


Figure 4—This is a block diagram of a PID control loop from Microchip Technology's application note AN908.

This is also the first advantage of SVM. The most critical part of these transformations is that the components that make up the transformed current vector determine the torque and rotor field. Now the torque can be independently manipulated to provide smoother, faster dynamic control of the motor.

Another advantage of SVM is the ability to reduce THD and switching losses by using an algorithm with the following switching rules: the trajectory of the rotating vector should be a circle; only one switching per state; no more than three switchings in one sample time; and the final state of one sample must be the initial state of the next sample.[8]

This is known as conventional SVM and the algorithm maintains symmetry in the switching waveforms. This symmetry is responsible for better performance due to reduced harmonics.

Lastly, SVM can provide a high modulation index, 0.907, whereas sine wave PWM can have a maximum modulation index of 0.79, both in the linear range. This results in a maximum line-to-line voltage that is almost equal to V_{DC}. This result is possibly the greatest advantage of SVM because the torque that is generated by the motor is higher than any other PWM technique. This provides supreme dynamic response of the motor.

PID TUNING

There are abundant sources of material available on the Internet and in libraries all around the world that discuss, in great detail, the methods for tuning proportional, integral, and derivative (PID) control loops. Therefore, it is not necessary to discuss such things here, but it may be helpful to present some details that are specific to this project.

Electric motors of all sizes and descriptions are built for specific applications. For our electric vehicle project, we decided to use a squirrel-cage AC induction motor. To use this motor for such an application, it is obvious that some tuning of the motor controller is required. To tune the motor, we developed a LabVIEW application that enables you to tune the PID variables. Then, after starting the motor, you can watch the motor response in real time.

Motor response is critical to the overall operation of the EV system. When a new motor is connected to the motor controller circuit board, the PID parameters must be properly aligned. If the motor does not respond well to input like stepping on the accelerator, there needs to be alterations to the PID control loops.

The proportional gain of the controller determines the maximum output level of the control loop. If the proportional gain is too low, the output of the control loop will never reach the set value of the input. If the proportional gain is too high, the output will oscillate and may become unstable. An ideal output from the proportional stage will closely follow the desired value without any oscillations or ringing when it reaches the steady state.

The integral stage is meant to reduce the steady state error, but integral gain can introduce ringing and overshoot. The derivative stage is meant to reduce the ringing and overshoot, but derivative gain can introduce steady state error. An ideal

balance between integral and derivative gain can be achieved through trial and tuning to give an optimal response.

The spatial vector modulation software that we downloaded from Microchip contains three PID loops.[9] One loop controls the flux field of the stator, the second loop controls the torque current in the stator, and the third loop, which contains the two previously mentioned loops, controls the angular velocity of the rotor. When you input a velocity demand or input a setpoint (acceleration/deceleration on the potentiometer), the torque and flux loops are passed a set value from the velocity loop and execute the appropriate controls. The feedback loop signals for the torque and flux loop are generated by the Hall-effect current sensors on i_a and i_b and the velocity demand feedback is given by the encoder (see Figure 4)

We elected not to use the velocity control loop for acceleration of the motor. The gas pedal of an internal combustion car sets acceleration but does not set the speed. Because the acceleration of a motor is directly proportional to the torque, we passed the input demand value from the gas pedal directly to the torque control loop. Luckily, SVM provided us with the isolated control over the torque that gives the end user a more realistic feeling gas pedal.

ALTERING MOTOR CONTROL PARAMETERS

It is important that you are familiar with the dsPIC assembly language so you can make changes to the motor control software, but expertise is not necessary. The SVM portion of the software is handled as a "black box." This means that you do not need to know exactly how it works. If you wish to alter the "black box," further documentation is available from Microchip.[9] The main motor control scheme is executed at a higher level in "PIC C" language in a file called ACIM.c posted on the *Circuit Cellar* FTP site.

From ACIM.c, you can manipulate almost every aspect of the motor behavior (see Listing 1). ACIM.c contains a function called `doControl()`, which is executed on a time-based interrupt vector. This is where you would make alterations to the code in order to make the motor perform differently. For our project, we needed to get the motor to respond to input the same way that a car would. Some of our colleagues needed to introduce regenerative braking into their software. This was all done within the `DoControl()` function. However, you cannot allow too many instructions to occur within this function; otherwise, the SVM period may become altered and impede proper operation.

HMI

The human machine interface (HMI) for this project is a LabVIEW 7.1 application built to communicate with the controller area network (CAN) bus module over "serial- over-USB" protocol by Future Technology Devices International. We used the DLP-USB232M-G integrated module featuring FTDI's FT232BL second-generation USB UART.[10] This module was an appropriate choice for development because it can be plugged into a standard 24-pin 0.6† wide DIP socket that is mounted on our daughter board and it could be moved easily to another board, if required. FTDI also developed its USB functions as LabVIEW Virtual Instruments (VIs) that are called from our HMI, which also made them a good choice for our application.[11] For message processing on the daughter board, we used a PIC18F4620[12] with an external CAN bus controller MCP2515[13] and transceiver MCP2551.[14] These CAN bus control ICs and the microcontroller are manufactured by Microchip.

PROTOCOL

Communication with the motor controller is through the CAN bus protocol. The CAN bus is a broadcast, differential serial bus. CAN was developed to be a robust communication protocol for particularly noisy environments; therefore, it is already commonly used in the automotive industry. Because CAN is a broadcast bus, there is no way to send a message to just one specific node. All nodes will invariably pick up all traffic. However, the CAN bus hardware provides local filtering so each node may react only to the relevant messages.[15] Because our development did not require a great deal of message traffic and uses only two nodes, we chose not to use the CAN bus filters and identifiers. We developed our own protocol using headers and identifiers contained

Listing 1—This code is from the motor control software from the ACIM.c file (`doControl()` function excerpt). It reads the speed input demand from a potentiometer on the control board, gets feedback from Hall-effect current sensors, and drives the motor with the option of open-loop or closed-loop control.

```
ReadSignedADC0( &ReadADCParm );

 // Set reference speed
 if(uGF.bit.ChangeSpeed)
    CtrlParm.qVelRef = ReadADCParm.qADValue/8;
 else
      CtrlParm.qVelRef = ReadADCParm.qADValue/16;

 if( uGF.bit.OpenLoop )
      {
      // OPENLOOP:  force rotating angle,Vd,Vq

      if( uGF.bit.ChangeMode )
          {
          // just changed to openloop
          uGF.bit.ChangeMode = 0;
          // synchronize angles
          OpenLoopParm.qAngFlux = CurModelParm.qAngFlux;

              // VqRef & VdRef not used
          CtrlParm.qVqRef = 0;
          CtrlParm.qVdRef = 0;
          }

    OpenLoopParm.qVelMech = CtrlParm.qVelRef;

    // calc rotational angle of rotor flux in 1.15 format
    // just for reference & sign needed by CorrectPhase
    CurModelParm.qVelMech = EncoderParm.qVelMech;
    CurModel();
```

Listing 2—This code is from the MCP2515.c file (`sliders()` code excerpt). The function gets the slider array from the LabVIEW interface over the CAN bus and sets the PID parameters for the motor control.

```
for(slider_count = 0; slider_count <= 8;)
{
   if((temp[count] == 0xFF) && (temp[count - 1] == 0xFF))
   //get each slider   value
   {
       temp1[1] = temp[count-2];
       temp1[2] = temp[count-3];

       check_sum = temp1[1] + temp1[2];
       //add id number
       temp1[0] = slider_count + 0x0B;//first one is 0x0B
       temp1[3] = check_sum + temp1[0];//new check sum
       //one shot mode for sending PID parameters
       CAN2510BitModify(0x0F,0x08,0x08); //one shot mode
       //  send data
       sent = CAN2510WriteStd(0x00, CAN2510_PRI_HIGHEST,
        4, temp1);//returns 0,1,2 to //indicate which
        buffer//-1 if no success
       Delay10KTCYx(5);//10ms
       CAN2510BitModify(0x2c, 0x1f, 0x00);//reset all tx
        and rx flags in canintf
       check_sum = 0;
       slider_count++;
   }//close if(temp[count]
   count++;
}//close slider loop
count = 0;//reset count
```

within the data portion of the CAN bus packet. This would be a disadvantage in more complex systems because all bus traffic is picked up by all nodes. To reduce microprocessor overhead in these designs, use the filters to receive only those messages required for processing by the specified node.

The CAN bus protocol also has error detection accomplished by the following methods: monitoring (transmitters compare the bits to be transmitted with the bits detected on the bus), cyclic redundancy check (CRC), bit stuffing, and message frame check.

Because the CAN bus protocol handles most of the error checking and timing issues, we did not need to introduce elaborate error-handling algorithms. We simply added headers and checksums to the data that is sent in a function called `sliders()` (see Listing 2). This function assembles and sends the PID tuning parameters and waits for an acknowledgment package, ACK or NAK, from the motor control microcontroller when the parameters are received before taking any further action.

FIRMWARE/SOFTWARE

The daughter board firmware is responsible for assembling CAN bus packets and sending and receiving messages. Currently, the only





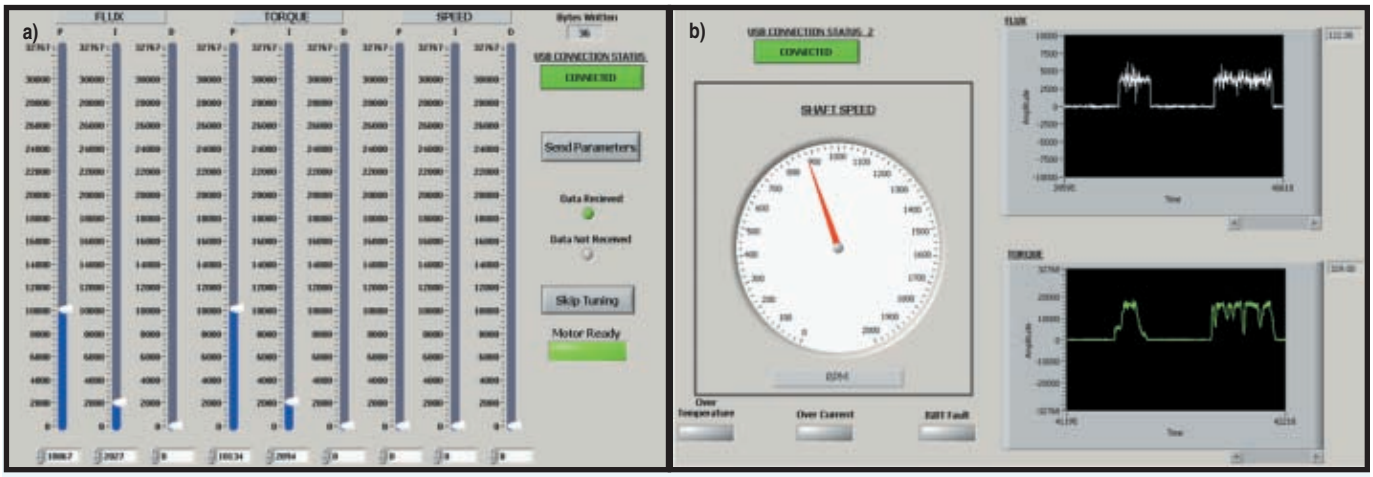


Photo 5a—This is the LabVIEW HMI that enables you to adjust the motor controller's PID tuning parameters. The sliders adjust the PID of the flux, torque, and speed and save the values in an array that is sent to the motor control board when "Send Parameters" is clicked and there is an indication that the data has been received. **b—**The real-time HMI enables you to observe feedback from the motor, shaft speed, and graphical representations of the torque and flux. Fault and alarm conditions from the motor and motor controller are also displayed.

messages sent by the daughter board are the PID tuning parameters. You can select the PID parameters using sliders in the LabVIEW GUI (see Photo 5a). The GUI software assembles and sends all of the PID parameters in one array, appending each individual parameter with 0xFFFF. This allows the firmware on the PIC18F4620 to parse the array and assemble each parameter into a CAN bus message. When receiving messages, the firmware extracts the data portion of the CAN bus message, puts it into a temporary buffer, and then puts the data onto the UART to be received by the GUI. Because we aimed to minimize the delay of data handling time and make the GUI as "real time" as possible, we processed all of our daughter board's incoming messages in the LabVIEW application. The HMI handles the incoming string of data using case statements to parse the array and determine the type of data arriving before sending it to the appropriate gauge or graph on the GUI (see Photo 5b).

As the electric vehicle becomes a more viable alternative for transportation, this project will be referenced as one way to help with the conversion of a gasoline engine to an electric motor. Currently, the main factors in conversion are safety and battery technology. As these concerns are addressed, an open-source project will be a welcome alternative for those who wish to take on an electric motor modification themselves.

Dan Hall is a recent Camosun College graduate with a diploma in Electronics Engineering Technology. You may contact him at dan.je.hall@gmail.com.

Tristan Kasmer is an engineering technologist from Camosun College. He currently lives in Victoria, BC, and is pursuing activities in alternative energy and control systems. You may contact him at tristankasmer@hotmail.com.

Doug Krahn is a recent Camosun College graduate with a diploma in Electronics Engineering Technology. He works for the Canadian Department of National Defense. You may contact Doug at dougkr@telus.net.

Adam McIntyre is a recent Camosun College graduate with a diploma in Electronics Engineering Technology. He is continuing his studies at the University of Victoria. You may contact Adam at a_mcintyre@shaw.ca.

Dena Ponech is a recent Camosun College graduate with a diploma in Electronics Engineering Technology. She works for the Canadian Department of National Defense in Marine Engineering Systems. You may contact Dena at denaponech@hotmail.com.

PROJECT FILES

To download code and additional files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

REFERENCES

[1] Powerex, Inc., "CM400DU-12F: ModuleTrench Gate Design Dual IGBTMOD," www.pwrx.com/pwrx/docs/cm400du-12f.pdf.

[2] R. Severns, "Design of Snubbers for Power Circuits," Cornell Dubilier Electronics, Inc., 2007, www.cde.com/tech/design.pdf.

[3] Powerex, Inc., "BG2B: Universal Gate Driver Prototype Board," 2005, www.pwrx.com/pwrx/app/bg2b_application_note.pdf.

[4] Powerex, Inc., "VLA106-15242: Isolated DC/DC Converter," 2007, www.pwrx.com/pwrx/docs/vla106_15242.pdf.

[5] Powerex, Inc., "VLA503: Hybrid IC IGBT Gate Driver," 2007, www.pwrx.com/pwrx/docs/vla503.pdf.

[6] Microchip Technology, Inc., "dsPICDEM MC1 Motor Control Development Board User's Guide," DS70098A, 2003, ww1.microchip.com/downloads/en/DeviceDoc/70098A.pdf.

.

[7] S. Bowling "How to Turn an AC Induction Motor Into a DC Motor (A Matter of Perspective)," Microchip Technology, Inc., www.newarkinone.thinkhost.com/brands/promos/leading_edge/Microchip_06_11man.pdf.

[8] R. Parekh, "AN955: VF Control of 3-Phase Induction Motor Using Space Vector Modulation," Microchip Technology, Inc., DS00955A, 2005, ww1.microchip.com/downloads/en/AppNotes/00955a.pdf.

[9] D. Ross, J. Theys, and S. Bowling, "AN908: Using the dsPIC30F for Vector Control of an ACIM," Microchip Technology, Inc., 2004, ww1.microchip.com/downloads/en/AppNotes/ACIM%20Vector%20Control%2000908a.pdf.

[10] DLP Design, "DLP-USB245M User Manual," 2002, www.ftdichip.com/Documents/DataSheets/DLP/dlp-usb245m13.pdf.

[11] Future Technology Devices International, "D2XX Programmer's Guide," 2005, ftp://ftp.efo.ru/pub/ftdichip/Documents/D2XXPG31.pdf.

[12] Microchip Technology, Inc., "PIC18F2525/2620/4525/4620 Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology," DS39626C, 2007, ww1.microchip.com/downloads/en/DeviceDoc/39626C.pdf.

[13] ———, "MCP2515: Stand-Alone CAN Controller with SPI Interface," DS21801D, 2005, ww1.microchip.com/downloads/en/DeviceDoc/21801d.pdf.

[14] ———, "MCP2551: High-Speed CAN Transceiver," DS21667E, 2007, ww1.microchip.com/downloads/en/DeviceDoc/21667E.pdf.

[15] Robert Bosch, "CAN Specification," Version 2.0, D-70442, 1991, www.semiconductors.bosch.de/pdf/can2spec.pdf.

RESOURCE

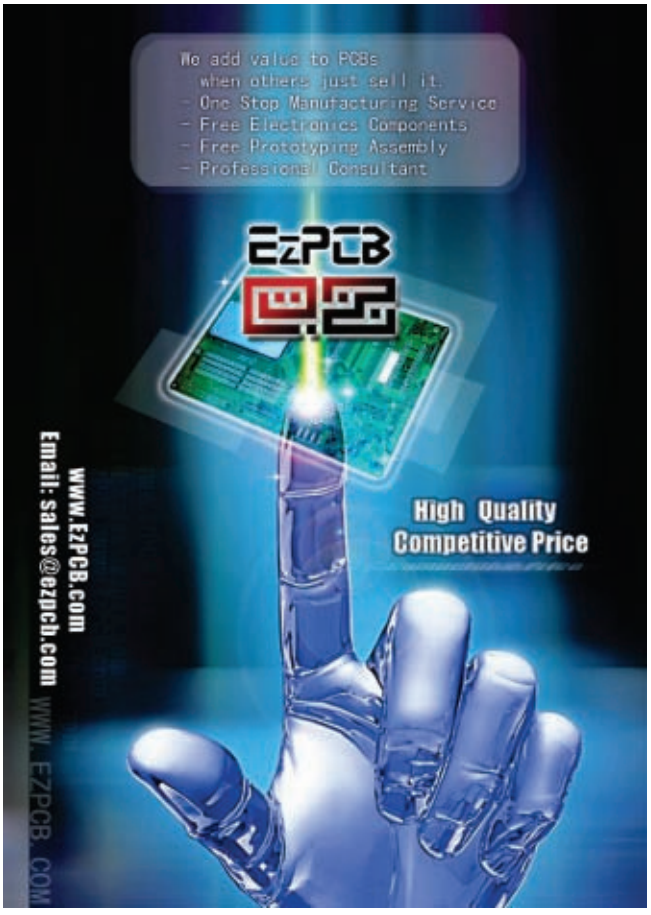
B. Bose, *Modern Power Electronics and AC Drives,* Prentice Hall, Upper Saddle River, NJ, 2002.

SOURCES

DLP-USB232M-G Integrated module and FT232BL USB UART
Future Technology Devices International
www.ftdichip.com

MC-1 Development board and PIC18F4620 microcontroller
Microchip Technology, Inc.
www.microchip.com

CM400DU-12F Transistor module, VLA106-15242 DC/DC converters, and VLA503-01 gate drivers
Powerex, Inc.
www.pwrx.com







CIRCUIT CELLAR

Designer's Notification Network

Circuit Cellar design contest entrants have received thousands of valuable development tools and product samples. Because of their contest participation, these engineers receive advance e-mail notice from Circuit Cellar as soon as new samples become available. Now you too can benefit from this early notification.

Welcome to the Designer's Notification Network. Print subscribers are invited to join the Network for advance notice about our new sample distribution programs.



Find out more at www.circuitcellar.com/network.

Microchip Technology

USB CONNECTIVITY

Sample Zone

If you need Full-Speed USB 2.0 Device, Embedded Host, Dual Role and On-The-Go solutions, you're in the right place. Circuit Cellar is pleased to launch the Microchip Technology USB connectivity sample zone—a special program created for readers of Circuit Cellar magazine.



Thousands of qualifying Circuit Cellar readers can receive 8-, 16-, and 32-bit MCUs with USB connectivity, providing easy migration with a single development environment. A select group will also receive complete USB Starter kits to better evaluate their design options.

Request these samples and discover other special offers from Microchip! Visit www.circuitcellar.com/USB.

CIRCUIT CELLAR®

MICROCHIP

Visit Microchip's USB Design Center for all current application notes, software downloads, development tool support and complete product information: www.microchip.com/usb

A Bootloader For Blackfin

David designed a two-stage bootloader that allows application firmware to be updated in the field to support bug fixes and additional features for specific end-user applications. It also adds capabilities to the native boot processing of the Blackfin chip. Although some details are specific to the Blackfin family of DSPs, some general features may be helpful on other CPUs.

Not long ago, I was working on an inertial measurement unit (IMU) that was based on the highly integrated ADIS16350 inertial sensor from Analog Devices that Tom Cantrell wrote about in his column in Issue 208 ("Thanks for the MEMS," 2007). This is a six-axis MEMS sensor (three axes of angular rate and three axes of acceleration) in a compact and rugged package. My client wanted to marry an Analog Devices Blackfin DSP chip to it in order to create a self-contained inertial measurement solution.

After a couple of design iterations, we came up with the board shown in Photo 1, which holds the Blackfin processor and various interface and power-supply components. A group of finished units in their boxes is shown in Photo 2. A key aspect of the implementation was that the firmware would need to be updated in the field, after the unit had left the controlled environment of the factory, in order to support both bug fixes to the basic functionality and additional features for specific end-user applications.

This article is about the two-stage bootloader that we developed that meets all our requirements and adds some capabilities to the native boot processing of the Blackfin chip. While much of this discussion will be specific to the Blackfin family of DSP chips, some aspects of it are more general and can be ported to other processors.



Photo 2—The sensor, PCB, and the power-I/O connector fit into a small custom box, which is 3 × 4 × 7.5 cm overall (not including the connector). The JTAG connector seen on the left side in Photo 1 is not populated in production units.

BOOTING A BLACKFIN

Booting the application code is a multistage process. When the Blackfin DSP chip receives a hardware reset, it begins executing code at address 0xEF000000, which is the beginning of its on-chip Boot ROM. The Boot ROM examines the external Boot Mode pins, which indicate the source of the application code, which can be an external 8-bit parallel PROM, an external SPI PROM (Blackfin is a SPI master), or another processor connected to the SPI (Blackfin is a SPI slave). In our application, the mode pins indicate that the boot device is an external SPI flash memory EEPROM, so the Boot ROM begins loading and processing blocks from that device. Figure 1 shows the key features of the Blackfin that come into play.



Photo 1—The Analog Devices ADIS16350 sensor is married to a small board carrying a Blackfin DSP to do post processing on the data. The CPU is the large chip in the center of the PCB, and the SPI flash PROM used for booting is the eight-pin device to its right.

The structure of a loader file consists of a series of blocks of various types. Each block contains a header, and the header contains a target address, a length, and several flags. The flags include IGNORE, FINAL, ZEROFILL, and INIT. The block will also contain data bytes following the header, as long as the length is nonzero and the ZEROFILL flag is not set.

The Boot ROM firmware processes blocks from the external SPI EEPROM one at a time, stopping only when it gets to a block that has its FINAL flag set. After processing that block, the Boot ROM jumps to the address contained in the Blackfin's reset event vector, which should point to the cold-start entry point of the newly loaded code. The Boot ROM initializes this register to a default value, but it is possible to modify this register (and other aspects of the state of the Blackfin) through the use of INIT blocks.

A block that has its IGNORE flag set may or may not contain data bytes (it usually does), but the Boot ROM skips over such blocks altogether—it simply adds the length of the block to its current SPI EEPROM address value and looks for the next block. Such blocks can be used to hold information for a second-stage bootloader program or for the application itself.

Any block that does not have the IGNORE flag set is either a data block or a ZEROFILL block. In the first case, the data associated with the block is copied to the specified target address for the specified length. In the second case, there is no data associated with the block in the EEPROM, but the specified target memory address is filled with bytes of zero for the specified length.

A data block that has its INIT flag set is called an "INIT block," and it must contain Blackfin code that begins execution at its load address. When the Boot ROM encounters such a block, it loads the data bytes into memory starting at the specified target address and then executes a subroutine call to that same address. The INIT block code must finish with a return instruction in order to allow the Boot ROM to continue processing blocks.

BUILDING THE SOFTWARE

Blackfin software is built in the usual way: compile/assemble, link, and load. The result of compiling or assembling source files is a set of object files. The linker is used to combine object files into a single executable image, which is stored in a file that has a .DXE extension. The loader converts one or more .DXE files into a single loader file (.LDR extension) that can be stored in a boot device, and contains the blocks that the Boot ROM will process. Among other things, the loader omits information in the .DXE file that isn't needed, such as debugging symbols and other metadata.

The loader includes the important feature that it can combine multiple .DXE files into a single loader file. It does this by inserting an IGNORE block at the beginning of each one. This IGNORE block includes a 4-byte data field that contains the total length of the entire set of blocks that represent that particular .DXE file. If a second-stage bootloader or other software wants to skip over the .DXE image, it can simply read the IGNORE block and then add its value to the EEPROM address pointer, which will then cause it to point to the next item in the EEPROM following the .DXE image.

Furthermore, the loader can optionally mark the code block of the first .DXE file as an INIT block. This means that both the first and second .DXE files in a multi-.DXE loader file will get loaded and executed in sequence by the Boot ROM.

SUPPORTING FIELD UPDATES

In order to support firmware updates in the field in a robust manner, it is necessary to have the capability of storing more than one copy of the application code in the SPI EERPOM—the one currently executing, and the newer one being installed. Until the install process is completed and verified, the DSP will execute the older version on any hardware reset. Furthermore, it is a requirement that the build process for the application is supported by the standard ADI VisualDSP++ development environment

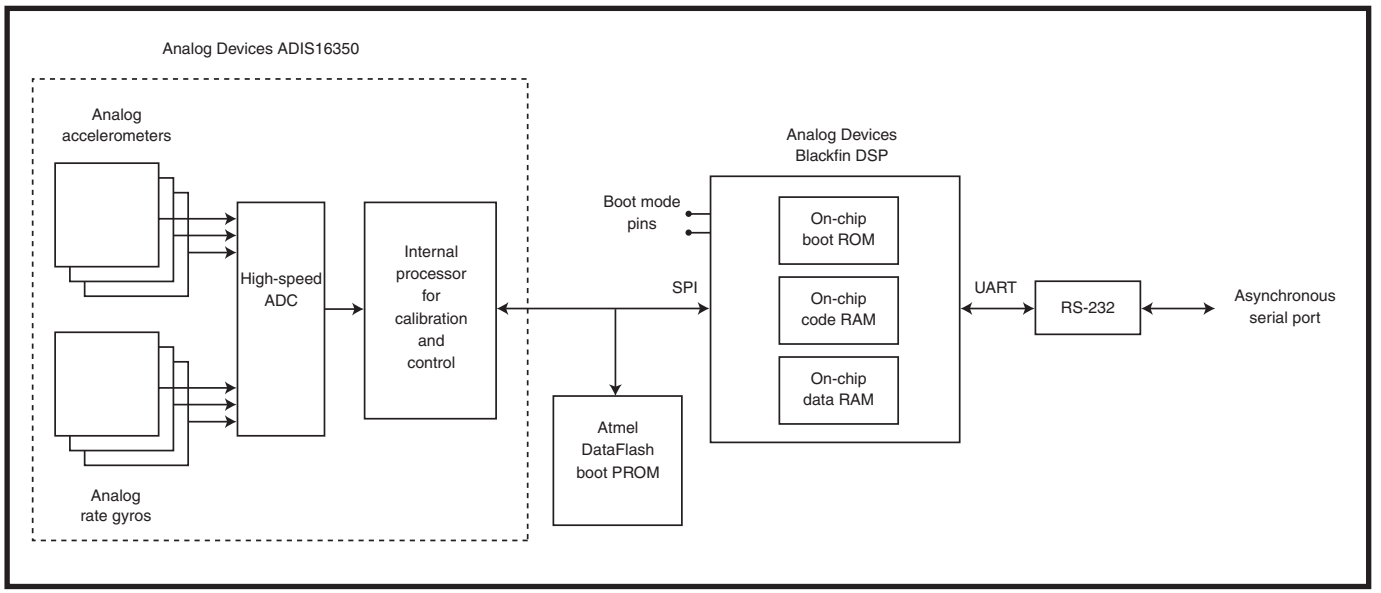


Figure 1—The Blackfin includes both RAM and ROM on-chip, with the RAM divided into separate areas for code and data. The on-chip SPI hardware interface can be configured for either master or slave operation, and can be used to boot the processor.

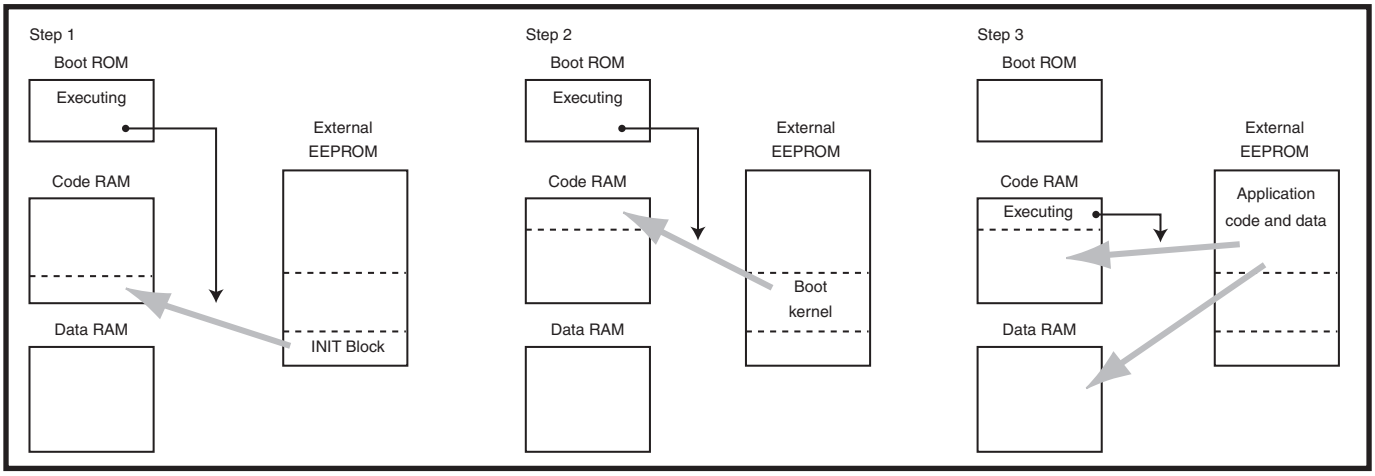


Figure 2—When the second-stage boot kernel is executing, it must be located in code memory in an area not required by the application code. This ends up being a three-stage process: the Boot ROM loads and executes the INIT block, which changes the reset vector; the Boot ROM loads and executes the second-stage boot kernel; and finally, the boot kernel loads and executes the application, restoring the original reset vector.

(the IDDE GUI) and that the build process for field upgrades be exactly the same as for factory-installed software. As I will show later, the capabilities of the Boot ROM and the build tools outlined previously are sufficient to achieve these requirements without resorting to custom tools in the build chain.

SUPPORTING NEWER EEPROM

There is a second issue related to the specific SPI EEPROM device used on our IMU, which is an Atmel AT45DB081D, a member of their DataFlash family. The Blackfin's on-chip Boot ROM supports older versions of the DataFlash family, but does not correctly support this newer member (the one with the –D suffix). The specific issue is that the –D device now implements the 0x03 "legacy read" command, while the older devices did not. This causes the Boot ROM to identify it as a generic 24-bit addressable SPI flash rather than a DataFlash, and the Boot ROM then assumes that the device has 256 bytes per page. As a result, the Boot ROM will work correctly only if the AT45DB081D is permanently set to its 256-byte Page mode.

In order to fully support the 264-byte Page mode, a second-stage boot kernel is required. This works as long as all of the block headers for the INIT block and the boot kernel itself fit into the first 256 bytes of the DataFlash device. As long as the boot kernel comprises a single code block, it can be arbitrarily long (well, up to 65,534 bytes) and extend beyond the first DataFlash page.

TWO-STAGE BOOT PROCESS

Therefore, we needed to implement a two-stage boot process. This consists of the boot ROM executing, which loads and then executes a second-stage boot kernel. The boot kernel, in turn, loads and executes the actual application firmware.

In order to support multiple copies of the application firmware in EEPROM at the same time, it is necessary to have some storage reserved in the EEPROM to indicate which copy is the "active" copy of the firmware at any given point in time. Within the context of the standard software development toolchain, the only way to accomplish this is to include an INIT block. The INIT block contains just a "return" instruction, along with four additional bytes of space that get initialized to all zeros. When the boot kernel gets control, it examines those 4 bytes to determine the base address of the "active" loader file in the EEPROM. It will then skip the first two .DXE images in this file (the INIT block and the boot kernel itself) and load the third .DXE image, which is the application firmware.

However, there is one additional twist. When the boot kernel is executing, it will be loading the application code, so the boot kernel itself must be located in code memory not needed by the application. There are two ways to achieve this—the boot ROM could load the boot kernel at the default address and then the boot kernel could relocate itself to a different address higher up in memory before continuing. But in many ways, it's simpler to just have the boot kernel load and run at the higher memory address to begin with. This adds the requirement that our INIT block must set the reset vector to that other address—so that the Boot ROM will jump to that address when it finishes loading the boot kernel—and that the boot kernel itself must restore the default reset vector value before it loads the application code. This still leaves open the possibility that the application can have its own INIT block that sets a non-default reset vector value. But constructing a loader file with two (or more) INIT blocks, while possible, would require a non-standard development toolchain.

As a result of all of this, we're going to build each application image (.LDR file) as a concatenation of three separate DSP executable programs (.DXE fields). The first is our INIT block that performs two functions: It provides a place in the EEPROM to store a pointer to the current application image, and it sets up the environment (a non-default reset vector value) in which the second-stage boot kernel runs (see Figure 2).

Figure 3 shows the details of the EEPROM layout. As a result of the space taken up by various data structures within the loader file, the code of our INIT block begins at address 0x00018. The first thing located here must be an executable instruction, so we place a jump instruction here (2 bytes) that jumps past the next 4 bytes, which is where we're going to store our application pointer. Therefore, the pointer starts at address 0x0001A (decimal 26). The code following the pointer sets the Blackfin reset vector to the code memory address where the boot kernel will load and run, and then returns to the Boot ROM.

The boot kernel is based on the source code for the Boot ROM, which is supplied by Analog Devices. Unnecessary features, such as support for devices other than the SPI EEPROM, were removed, while new features were added. The new features include support for Atmel DataFlash in either 256-byte Page mode or 264-byte Page mode, and the ability to select multiple loader files based on the pointer stored at offset 26 in the EEPROM. Otherwise, it performs the same basic

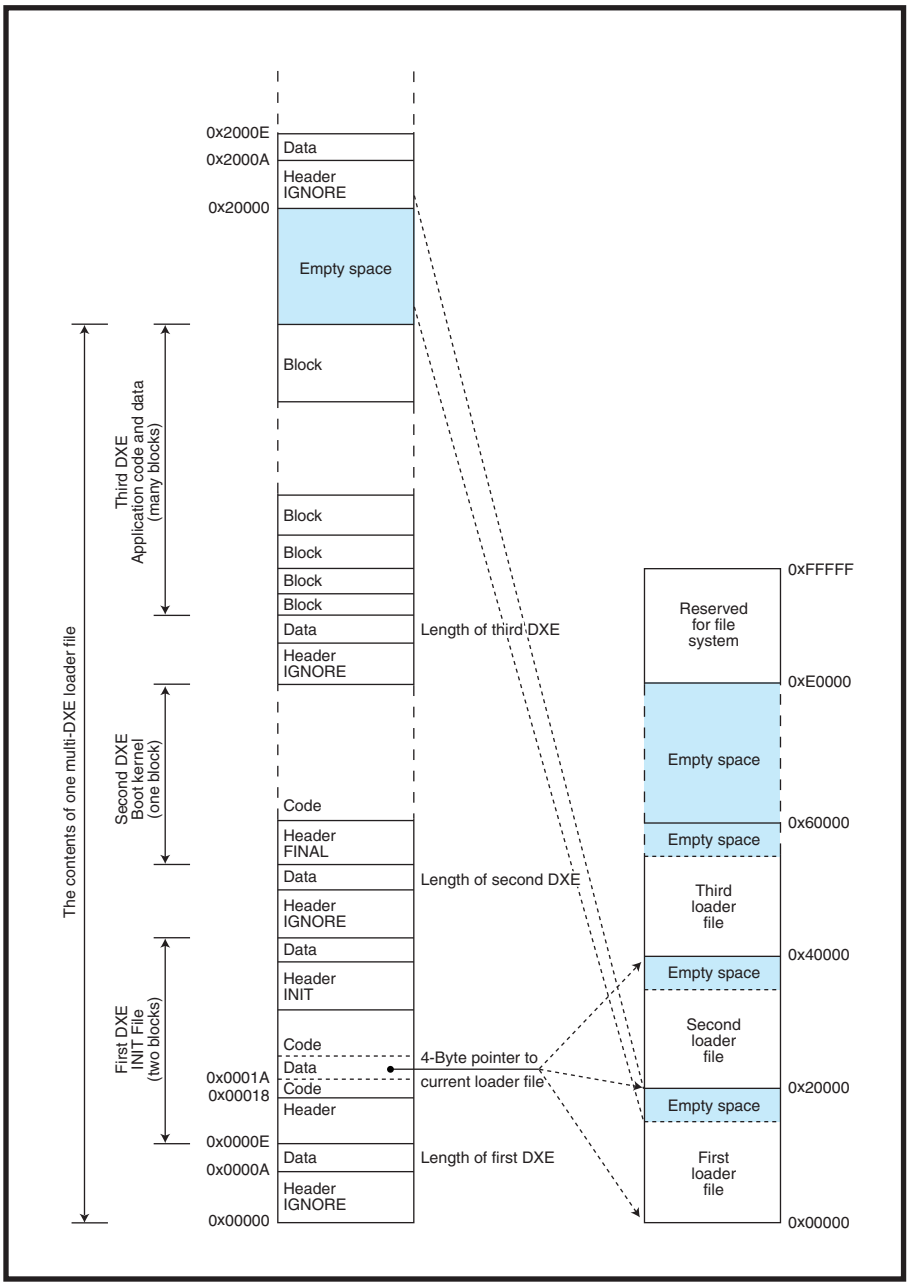


Figure 3—The right-hand side of this diagram shows the overall layout of areas within the EEPROM, while the left-hand side shows the internal structure of a single .LDR file.

PIC-SERVO
MOTION CONTROL
MOTION CONTROLLERS FOR BRUSH, BRUSHLESS AND STEPPER MOTORS.
• controller chips
• controller boards
www.picservo.com
JEFFREY KERR, LLC

THE SERIAL PORT LIVES!
Everything you need to know about COM ports, USB virtual COM ports, & asynchronous serial ports for embedded systems.
Hardware & software for RS-232 & RS-485. Wireless options and more.
Serial Port Complete
Second Edition
Jan Axelson
ISBN 978-1-931448-06-2 $39.95
Lakeview Research LLC www.Lvr.com
From the author of USB Complete

mach64
PROGRAMMABLE LOGIC STARTER KIT
Based on the Lattice ispMach 4064.
Includes 250 page lab manual.
Learn CPLDs the fun way with the MACH64! This complete kit comes with everything you need to take you from mystery to mastery with CPLDs and programmable logic.
Learn to turn software into hardware!
www.XGAMESTATION.com

block-loading functions as the Boot ROM itself.

THE FIRMWARE UPDATE PROCESS

So far, we have not addressed the question of how firmware updates actually get programmed into the EEPROM in the field. This could be implemented as a function within the boot kernel itself, but this would make the boot kernel significantly larger, and would "lock in" the implementation to loading methods that are known at the time the IMU is manufactured. Remember, although every copy of the application code contains a boot kernel, only the one loaded at the factory ever gets executed.

Therefore, we chose to place the firmware that implements the field-update function into the application code itself. The application uses a message-based interface via the Blackfin's UART port, and messages specific to the firmware update function have been added to that interface. While this gives us tremendous flexibility to add new functionality to the update function at any time, it also creates the onus that the update function be bug-free—if we ever "break" the field-update function by introducing a bug, then the unit must be disassembled so that the CPU's JTAG interface can be accessed to reprogram the EEPROM from scratch.

Any number of separate loader file images can be stored in the SPI EEPROM, but the present scheme uses just three, with predefined starting addresses of 0x00000, 0x20000, and 0x40000. The image at 0x00000 is always the first one loaded, usually when the IMU is manufactured. This can be accomplished by programming the EEPROM before it is installed on the board, or by using the Blackfin's JTAG interface and the loader utility built into the IDDE toolchain.

Updates applied in the field are loaded at 0x20000 or 0x40000, based on a simple "toggle" algorithm. Before beginning a firmware update, the application code examines the value stored at offset 26 (0x0001A). If this value is equal to 0x20000, the next update gets loaded at 0x40000. Otherwise (the current value is either 0x00000 or 0x40000), the next update gets loaded at 0x20000.

The firmware update message protocol includes several layers of error checking and recovery. Each data message is protected by a checksum, and once the entire loader file has been transmitted, a checksum over the entire image is separately calculated. If any errors are encountered, the entire process can be restarted without any risk to the system.

When a firmware update is completed, the last thing that happens is that locations 26–29 in the EEPROM are updated to point to the new image. This operation is the only step in the entire process during which a problem (e.g., a power interruption) could leave the IMU in a nonfunctional state. Once this step is complete, the next reboot of the IMU will load the new application code. Note also that regardless of the value of this pointer, the INIT block and the boot kernel that are used are always the ones starting at address 0x00000, because this is hard-coded into the Boot ROM. The only way to update the INIT block or the second-stage boot kernel is to use the JTAG interface to store a new loader file at offset 0x00000 in the EEPROM.

When the second-stage boot kernel finally executes, it examines locations 26–29 in the EEPROM in order to get a pointer to the "current" application image. This pointer contains the address of the beginning of the image, including the INIT block and the boot kernel itself, so at this point, the boot kernel must skip these items (using the information in the IGNORE blocks) before it can begin loading the application.

START YOUR DESIGN

The archive for this article on the *Circuit Cellar* FTP site includes the source code for the original Blackfin Boot ROM and our boot kernel for comparison. It also includes the INIT block. It does not include the application code for the firmware update function, but if you would like assistance in this area, feel free to get in touch with me.

We have found this system to work very well in practice, and an OEM who is using the IMU in a larger embedded system has successfully implemented the firmware update protocol in their host CPU. Whenever they need new functionality in the IMU, we can e-mail them an updated loader file and they can install it without disassembling the system in any way.

While many of the details discussed here are specific to the Blackfin boot-process, I hope that the more general concepts will be found useful on other CPUs as well. I want to thank Robert Pinto and Enpoint, LLC, for their support and the permission to tell you about this aspect of their product.

David Tweed (dtweed@acm.org) is a hardware and real-time firmware engineering consultant who has been working with embedded processors starting in 1976 with the Intel 8008. His system design experience includes computer design from supercomputers to workstations, digital telecommunications systems, and the application of embedded microcomputers and DSPs. He is also a Circuit Cellar *project editor and quiz master. When not playing with electronics and software, he pursues his hobby as an amateur musician, playing keyboards and low brass instruments in several community groups.*

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

RESOURCE

Enpoint, "GPS/Inertial Solutions," www.enpoint.com.

SOURCES

ADIS16350 Inertial sensor and Blackfin DSP
Analog Devices, Inc.
www.analog.com

AT45DB081D DataFlash
Atmel Corp.
www.atmel.com

RENESAS SuperH

SuperH Flash Microcontroller reaches speeds up to 160MHz

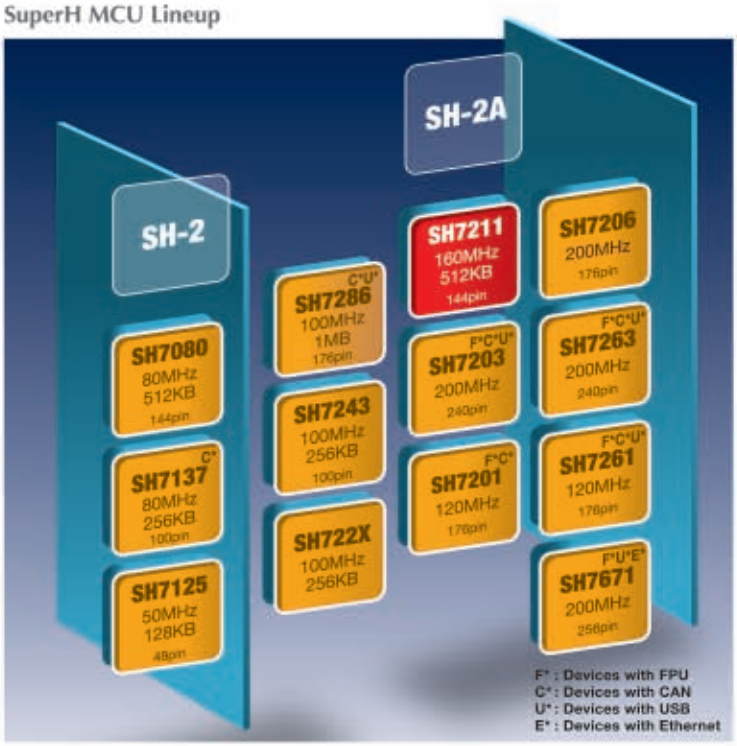
Superscalar performance, high-speed on-chip FLASH memory access, and much more

Renesas Technology

No. 1* supplier of microcontrollers in the world

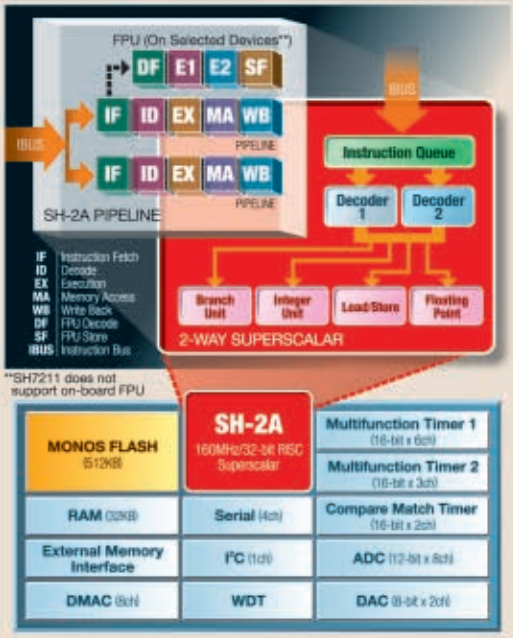
proudly presents the SuperH Family of devices. SuperH devices equipped with the SH-2A core offer superscalar performance at speeds of 160MHz, allowing high-speed access to on-chip FLASH memory and up to 200MHz CPU performance (ROMless devices). Enhanced features that include on board Floating Point Unit (FPU), Multiply Accumulate Unit (MAC), High-Speed Barrel Shifter and advanced addressing modes deliver DSP-like performance in RISC style architecture without the complicated programming associated with a DSP engine. The SuperH RISC engine and the SH-2A core are establishing new performance standards in the industry, and are ideal for systems that demand real-time, high-precision control and require a combination of high performance CPU with high-speed flash.

SuperH MCU Lineup



HOT Products **SH7211F**

(R5F72115D180FPV)



Top Reasons to Select SuperH

480 Dhrystone MIPS at **200** MHz

- **Superscalar Performance**
  - Two instructions are executed per cycle at 160MHz
  - World's fastest embedded FLASH with 12.5ns read access time.
- **Fast Real-Time Control**
  - Register Bank architecture (15 Banks) for context switching enables 37.5nsec (6 cycles) interrupt latency time.
- **High Integration**
  - 512KB On-Chip Flash / 32KB On-Chip RAM
  - Advanced 16-bit PWM timers to drive two motors simultaneously
  - 1.25μsec 12-bit A/D conversion with 3 sample & hold circuits

*Source: Gartner (March 2007) "2006 Worldwide Microcontroller Vendor Revenue" GJ07168

#1 MCU REACH FURTHER

Get Started Today -

Register to qualify for a SuperH MCU Kit

www.america.renesas.com/ReachSH/d



Starter Kit for SH7211F: R0K572115S000BE
USB Emulator: HS0005KCU11H

DEVCON Developers Conference
Oct. 13-15 San Diego
www.RenesasDevCon.com

Renesas Technology Corp.

Everywhere you imagine. RENESAS



IR Proximity Sensing

Ed takes a look at IR proximity sensing after seeing a colleague's homebrew chord keyboard. Even if you don't need a hand-mounted keyboard, knowing how a contactless switch works should come in handy.

A recent Mid-Hudson Valley Linux Users Group meeting featured Greg Priest-Dorman demonstrating his wearable Linux computer system. He's been using the homebrew chording keyboard shown in Photo 1 for the last decade, with a succession of computers that each ran afoul of Moore's Law.

The key clicking isn't objectionable and, indeed, seems quieter than most laptop keyboards. Perhaps that's because he removed several coils from each return spring to achieve faster and less tiring typing.

The only disadvantage seems to be the structure holding the thumb switches. I immediately thought of mounting IR proximity sensors over my fingers out to the first knuckle, so that extending, rather than flexing, my fingers would activate the switches.

However, making an optical sensor work under real-world conditions requires somewhat more circuitry than the datasheet might lead you to believe. In this column, I'll take a look at IR proximity sensing, which will be useful even if you're not interested in weird keyboards.

REFLECTIONS ON LIGHT

Optical distance measurement requires different techniques for different tasks. Applying the wrong method can cause serious headaches, so sorting through the options is a good first step for any design.

Triangulation makes use of the geometry between the emitter, receiver, and a remote object. Given a known distance between the emitter and receiver and the measured angles between the ends of that baseline and the object, you can solve the triangle and determine the object's distance. This works well for surveyors, even before GPS simplified the baseline measurements, and scales down to centimeter distances.

Time-of-flight measurements take advantage of light's 1 ns/ft speed. A laser beam modulated with a pseudo-random bit sequence bounces from the object to a photodiode and is then demodulated back to bits. A high-speed autocorrelator matches the incoming and outgoing bitstreams to find their time difference, which is directly proportional to distance. This also works for land-surveyor distances and scales well to a few meters.

Interferometry measurements split a single laser beam into two paths, bounce one from a mirror on the target, then combine them to form an interference pattern. The pattern varies from bright to dark to bright with distance, so measuring the intensity and counting peaks give a resolution in microns. This obviously works best for stationary objects and has a nasty sensitivity to vibration.

Measuring the intensity of light reflected from an object provides a less accurate and more compact distance measurement, because, all else being equal, a more distant object will be dimmer and a closer object will be brighter. Unfortunately, all else is rarely equal.

Intensity sensors generally use infrared light because IR's longer wavelength makes objects more reflective and, perhaps, for the simple reason that IR isn't visible. Most visible light sources also emit plenty of IR; all that ambient light reaching the sensor, whether reflected from the object or not, adds to the measured intensity and reduces the apparent distance.



Photo 1—Greg's chord keyboard has one modified push button switch at each fingertip and three for the thumb. He can reach below the key arch to get a more-or-less normal grip on objects.

The object's reflectivity sets the upper limit of light available at the sensor: a dull black object reflects far less light than a shiny white object at the same distance and will thus appear to be farther away. Not surprisingly, an object's visible-light reflectivity rarely matches its IR reflectivity.

The reflected light intensity at the

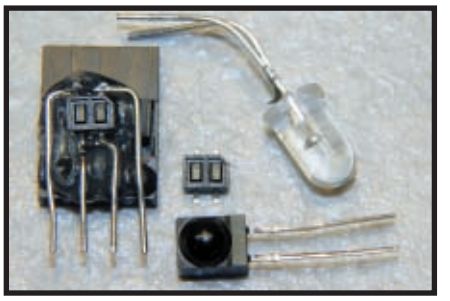


Photo 2—A through-hole IR LED and photodiode dwarf an SMD proximity sensor. I soldered wires to the sensor's pads and glued it to a plastic connector shell for the breadboard in Photo 3.

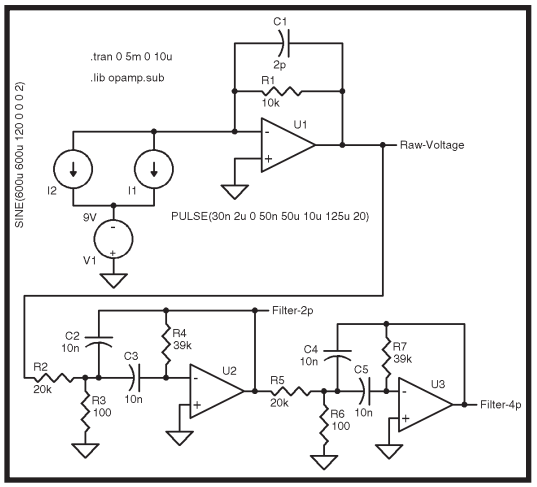


Figure 1—A Spice simulation model captures the essential elements of a modulated-IR proximity sensor. I1, an 8-kHz square-wave current source, simulates the reflected photocurrent. Current source I2 produces an offset 120-Hz sine wave mimicking a fluorescent lamp glaring on the photodiode. U1 converts those photocurrents into a DC-coupled voltage. U2 and U3 form a four-pole 800-Hz band-pass filter centered at 8 kHz to extract the modulated signal. V1 provides reverse bias for a real photodiode, but has no effect on the simulation.

sensor doesn't vary inversely as the fourth power of the object's distance as you might expect, because the source and reflection aren't ideal point sources and the object and receiver aren't isotropic. A laser pointer's spot on a diffuse surface forms one limiting case: the total power captured by the detector varies nearly as the square of the distance because the laser spot is effectively a point source in the sensor's field of view. A mirrored surface angled away from the sensor forms another limit: there's no reflection regardless of distance.

Despite those problems, a reflected-light sensor can provide reliable distance measurement in a well-controlled environment. For example, a sensor calibrated for a specific object with a constant reflectivity used with known ambient lighting can work quite well.

My fingers have a reasonably stable reflectivity and aren't at all mirror-like, but ambient illumination can vary from near darkness to full sunlight, with incandescent, LED, and fluorescent lighting thrown in for good measure. Obviously, depending on ideal behavior won't suffice, so let's see what will.

MEASURING REFLECTIONS

It turns out that semiconductor junctions are inherently photosensitive: early glass-cased diodes caused baffling problems on analog circuits exposed to fluorescent lights. Optimizing that effect produces a photodiode, the basic optical sensor, with a reverse current linearly related to the light energy incident on its PN junction. The Vishay Semiconductors BPV23NF infrared photodiode along the bottom of Photo 2 has an IR-transparent black epoxy case that blocks visible light, with a molded lens for a 120° field of view.

Although you can drive an ammeter directly from a reverse-biased photodiode and a battery, most circuits require a voltage proportional to the current. A circuit converting an input current to an output voltage has a gain with units of V/A, an impedance, and is known as a transimpedance amplifier. That sounds complicated, but an ordinary resistor has a "transimpedance gain" equal to its resistance.

Indeed, the resistor shown in many sensor datasheets may suffice, but the photodiode's reverse bias will vary with its photocurrent. Worse, the resistor forms an RC low-pass filter with the photodiode's junction capacitance, which can dramatically slow the voltage's rise and fall times.

For example, the BPV23NF's junction capacitance is about 20 pF with 2-V reverse bias. The 100-kΩ resistor required to get 1 V from a 10-µA photocurrent forms a low-pass filter with a 2-µs time constant:

$$\tau = 2\,\mu s = 100 \times 10^{3}\,\Omega \cdot 20 \times 10^{-12}\,F \quad [1]$$

That corresponds to a cutoff frequency of about:

$$80\ kHz = \frac{1}{2\pi \cdot \tau} = \frac{1}{2\pi \cdot 2\,\mu s} \quad [2]$$

An 80-kHz bandwidth may suffice for some purposes and be woefully inadequate for others. In the latter case, you must throw more hardware at the problem.

Op-amp U1 in Figure 1 acts as a linear transimpedance amplifier with a gain set by feedback resistor R1. Current source I1 simulates the photodiode, although it omits all the nonlinearities and real-world effects. Voltage source V1 isn't needed with an ideal current source, but it's a reminder that you must reverse-bias a real photodiode.

U1's inverting input has a high impedance and can't supply current to the photodiode. Therefore, the photocurrent must come through the feedback resistor and the amplifier's output voltage will drive that current through R1. Because U1's noninverting input is at 0 V and feedback maintains the inverting input at 0 V, the output voltage equals the photocurrent times R1: the transimpedance is exactly R1 volts per ampere.

C1 compensates the op-amp for the photodiode's capacitance. Finding the exact value requires some tweakage starting from the value predicted from the known feedback resistor (R1), the sum of the photodiode capacitance and

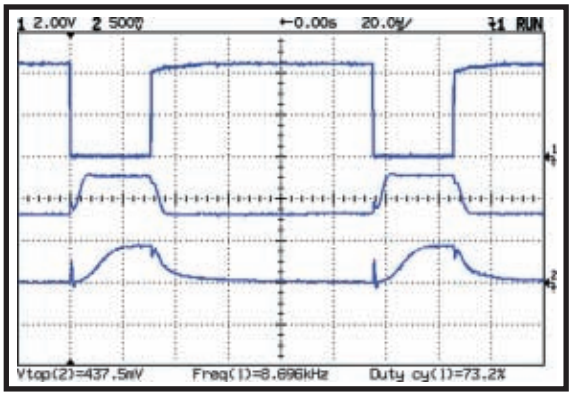


Photo 4—Simple DC threshold detection fails when confronted with ambient light. The two lower traces show the SMD sensor's response with and without an incandescent flashlight shining on the target. The baseline offset is larger than the signal. The upper trace shows the low-active IR LED drive.

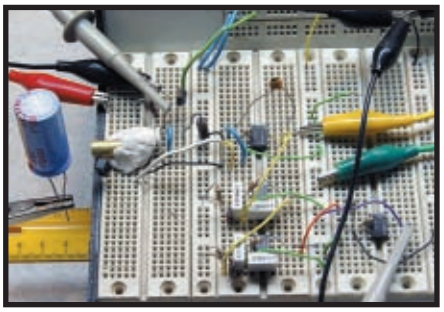


Photo 3—This ugly breadboard implements the transimpedance amplifier and filters shown in Figure 1. A signal generator pulses the IR LED inside the brass tube through a one-transistor buffer. The fourth op-amp on the far right provides voltage gain. The green and yellow clips lead to a pair of 9-V batteries that supply quiet DC power.

op-amp input capacitance (CT), and the op-amp's gain-bandwidth product (GBW):

$$C1 = \sqrt{\frac{CT}{2\pi \cdot R1 \cdot GBW}} \quad [3]$$

I used an LF411 with a 1-MHz GBW and an input capacitance of a few picofarads. The BPV23NF capacitance is about 10 pF at 9-V reverse bias, so C1 should be 15 pF: obviously, your results will vary.

Photo 3 shows a breadboard version of the circuit in Figure 1, with the IR LED hidden inside the brass tube and lump of modeling clay near the left edge. The photodiode stands beside the brass tube, facing a blue capacitor serving as a target.

A discrete LED and photodiode would be too large for hand-mounted sensors, so I also tried the surface-mount proximity sensor shown on the left of Photo 2. It's an anonymous surplus part similar (but unrelated) to an Avago Technologies HSDL-9100. I soldered wires to the SMD pads and glued the whole affair to a connector shell for use on the solderless breadboard.

Photo 4 shows the SMD sensor's response to a block of wood positioned 8 mm away, with ordinary fluorescent lamps overhead. The upper trace is the low-active LED drive. The bottom trace is U1's output voltage: 500 mV corresponds to 50 µA of photocurrent, with essentially no ambient-light interference. The SMD sensor is probably a phototransistor, as the rise and fall times are much slower than a photodiode.

You could feed this signal into a comparator set to 250 mV and get a decent logic signal, which is what many datasheets and app notes indicate.

Looks pretty good, doesn't it? The middle trace in Photo 4 hints at why this simple scheme might not work quite as well as you expect.

AMBIENT LIMIT

I aimed a small incandescent-bulb flashlight at the wood target and promptly raised the background voltage to 800 mV, far higher than the entire signal in the lower trace. A simple 250-mV comparator would be stuck active, even with the LED off.

If you measure the background level, perhaps with a microcontroller's ADC input, then turn the LED on and measure the level again, your program can compensate for background illumination. Alas, another trade-off might make that impossible.

I picked R1, the feedback resistor that sets the transimpedance gain, by taking the breadboard outdoors in direct sunlight. The BPV32 photodiode produced 600 µA of current and I picked R1 for 6 V at the output of U1, a reasonable limit for ±9-V supplies. Remember that no useful signal emerges from an op-amp jammed at its maximum output voltage.

In general, both the maximum illumination on the photodiode and the maximum allowable output voltage put an upper limit on the transimpedance gain. High ambient lighting and a low amplifier voltage range may reduce the maximum gain enough that the photocurrent from the reflected LED becomes immeasurable. Single-supply op-amps used in microcontroller applications exacerbate this problem.

For example, an op-amp capable of 3-V rail-to-rail output requires a smaller R1 to handle the same ambient light:

$$R1 = 5\,k\Omega = \frac{3\ V}{600\ \mu A} \quad [4]$$

The photocurrent that produced the 500-mV change shown in Photo 4 will now produce only 250 mV.

If the response varies inversely as the square of the distance, doubling the distance reduces the signal by a factor of four. At 16 mm, the signal drops to 60 mV.

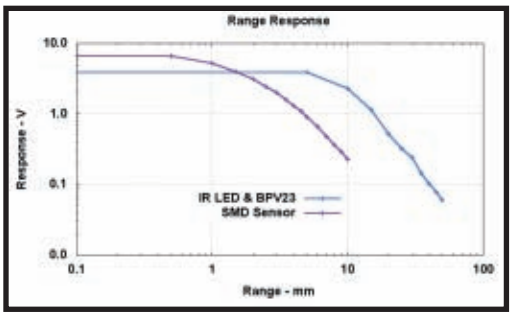


Figure 3—Plotting voltage-versus-range on log-log scales shows nice 1/rx responses, at least beyond a few millimeters. The voltage is the RMS output of the 8-kHz four-pole band-pass filter, boosted by an additional op-amp stage.

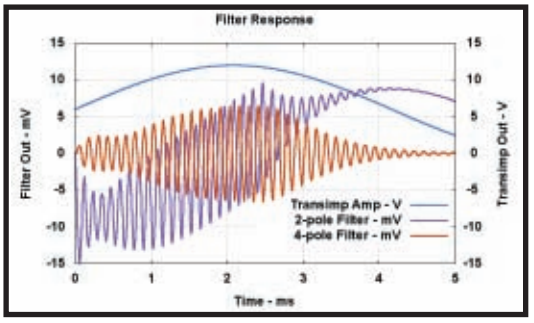


Figure 2—The blue trace represents the sum of the 120-Hz and 8-kHz signals, measured in volts on the right-hand axis. The brown trace shows the output of the four-pole filter, measured on the left-hand axis, with all traces of the interference removed. Note the 1000:1 ratio of the interference-to-signal voltages!

Suppose the signal feeds an 8-bit microcontroller ADC with a 3-V reference. The resolution will be:

$$1\ LSB = 11.7\ mV = \frac{3\ V}{256} \quad [5]$$

At that range, the signal amounts to just 5 LSB of the ADC's range, which is probably down in the noise level. More-distant targets will be invisible, but you can't increase the gain because there's no headroom for larger signals.

Obviously, we need more circuitry!

DEMODULATION AND DISTANCE

A photodiode has a nearly linear relationship between incident light power and photocurrent, which means that a given change in illumination produces the same change in photocurrent regardless of the overall light level. In practical terms, the photodiode produces the same current difference between LED-on and LED-off in sunlight as it does in darkness.

Turning the LED on and off at a known frequency produces an AC photocurrent that can be extracted from the ambient illumination, amplified as needed, then demodulated to produce distance information. The signal amplitude depends on the object's reflectivity and distance, just as before, but now the signal can be separated from the background illumination.

I chose an 8-kHz modulation frequency based on the relatively slow rise and fall time of the SMD sensor. I didn't need a high-speed transimpedance amplifier to achieve that bandwidth, but I left U1 in the circuit to simplify the discussion.

U2 and U3 along the bottom of

Figure 1 each implement a two-pole band-pass filter centered at 8 kHz, with Q = 10 for a bandwidth of about 800 Hz. The filter shape isn't critical; I used the Analog Devices Filter Designer web site to create 1-dB Chebyshev filters with a multiple feedback topology to reduce the parts count. The pair forms a four-pole filter with a net 500-Hz bandwidth.

Current source I2 in Figure 1 is an offset 600-µA sine wave at 120 Hz simulating a twice-as-bright-as-the-sun fluorescent lamp, shown by the smooth blue trace in Figure 2. You cannot see the 2-µA square-wave modulation from I1 that simulates the photodiode's response to the LED, but it's superimposed on that trace from t = 0 to t = 2.5 ms.

The purple trace shows that the output of U2, with only two poles of filtering, still has a significant 120-Hz component. The brown trace is the output of U3, with only the 8-kHz modulation remaining. Notice how the output requires about 1.5 ms to reach full amplitude, then tapers off after the modulating signal stops. Obviously, the firmware must take those delays into account when it samples the filter's output.

I used 5% resistors and 10% capacitors, so the as-built filters wound up at 8.7 kHz. In actual use, the firmware would measure the true center frequency and adjust the LED modulation to suit. I just tweaked the function generator frequency.

I measured the response of the two sensors as a function of distance and plotted them on log-log axes in Figure 3. The slope of a straight line with log-log axes equals the coefficient b of the exponent in the relationship:

$$y = ae^{bx} \quad [6]$$

An eyeballed straight-line fit to the right-hand asymptote of the SMD curve has a slope of:

$$b = -2 = \frac{\Delta Y}{\Delta X} = \frac{\log(10/0.1)}{\log(1.5/15)} \quad [7]$$

The PBV23 curve is slightly steeper, perhaps because a glint on the capacitor acts as a point source:

$$b = -2.6 = \frac{\log(10/0.1)}{\log(7/40)} \quad [8]$$

Varying background illumination still produces a relatively small effect that the firmware could periodically measure and fold into the calculation. Because I'm only interested in proximity, not actual distance, the calculation need not have any great linearity or precision.

Ambient illumination and voltage limits still set the upper bound for the transimpedance gain of U1, but additional gain after the band-pass filters can boost the AC signal to more tractable levels prior to an ADC or demodulator. In fact, that's exactly what the fourth op-amp in Photo 3 does.

In any event, you should now understand why commercial IR distance sensors use modulated IR and why the simple DC sensors shown in datasheets just don't work in the real world. Just a little more circuitry can dramatically increase reliability and accuracy: use the gain wisely!

CONTACT RELEASE

I'm still mulling over how to build my chord keyboard, but Spaceman Spiff's SpiffChorder USB keyboard interface mentioned in the Resources section will certainly form part of the solution. Greg's keyboard plugs directly into the USB port of his Sony ultraportable PC, with no need for special drivers or shim code.

You could, of course, use a two-button chord keyboard: a Morse code keyer with iambic dot and dash switches. You must send with metronomic precision for accurate machine decoding, but the geek points would be essentially infinite!

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@ieee.org with "Circuit Cellar" in the subject to avoid spam filters.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

RESOURCES

Analog Devices, Inc., "Active Filter Synthesis: A Simple Tool for Designing Active Filters Using Voltage-Feedback Opamps," www.analog.com/Analog_Root/static/techSupport/designTools/interactiveTools/filter/filter.html.

Background on Chord Keyboards and Keyers, "Chorded keyboard," Wikipedia, http://en.wikipedia.org/wiki/Chorded_keyboard.

P. C. D. Hobbs, "Photodiode Front Ends: The REAL Story," Optics & Photonics News, Optical Society of America, April 2001, http://users.bestweb.net/~hobbs/frontends/frontends.pdf.

M. Holm Olsen "SpiffChorder USB Keyboard Project," http://symlink.dk/projects/spiffchorder.

Morse Code Keyboards, www.makoa.org/jlubin/morsecode.htm and www.westest.com/darci/usbindex.html.

E. Nisley, "Foolish LED Tricks," *Circuit Cellar* 177, 2005.

———, "IR Sensing," *Circuit Cellar* 157, 2003.

———, "LED Optics: Lights! Diodes! Current! Math!," *Circuit Cellar* 183, 2005.

Op-amp and Analog Circuitry, "The Audio Pages," Elliott Sound Productions, www.sound.au.com.

B. Pease, "What's All This Transimpedance Amplifier Stuff, Anyhow? (Part 1)," *Electronic Design*, 2001, http://electronicdesign.com/Articles/Index.cfm?ArticleID=4346.

G. Priest-Dorman, "Chording Keyboard," http://chorder.cs.vassar.edu.

Vishay Semiconductors optical sensor information, www.vishay.com/optical-sensors/.

SOURCES

HSDL-9100 Analog output reflective sensor
Avago Technologies (née Agilent, née HP)
www.avagotech.com/products/parametric/ir_sensors/proximity_sensors/

SwitcherCAD Spice III simulator
Linear Technology Corp.
www.linear.com/designtools/software/switchercad.jsp

GP2Y0AH01K0F High-precision displacement sensor
Sharp Electronics Corp.
www.sharpsma.com

BPV23NF Infrared photodiode
Vishay Semiconductors
www.vishay.com

PSoC Design Techniques (Part 2)

Add DSP Effects, A User Interface, And More

Chris continues explaining PSoC design techniques by describing his eight-channel mixer project. This month, he covers how to enhance the single-chip design with DSP effects, an intercom mode, a user interface, speech, and permanent setting memory.

Last month, I demonstrated some advanced analog design techniques in the form of a PSoC eight-channel mixer with adjustment knobs. In this article, I will explain how to further enhance the single-chip design with digital signal processing (DSP) effects, an intercom mode, a user interface, speech, and permanent setting memory.

ADD DSP EFFECTS

Suppose you wanted to add some DSP to one of the mixer's audio channels. The PSoC is certainly capable of this feat. There are some digital and analog resources left over and many options to accomplish some basic DSP. In general, any DSP effect comprises three components: an input to create digital data, an output to represent the result as an analog signal, and the effect (algorithm) altering the samples in between.

ANALOG-TO-DIGITAL DATA

The input can be one of many types of ADCs available in the PSoC. Some are not suited to audio because their potential sampling rates are too low to properly represent audible sound. The choices involve trade-offs between sampling rates, sample resolutions, and required processing time. By migrating the design to a Cypress Semiconductor CY8C29466 PSoC, you end up with two times the ROM, eight times the RAM, and an advanced Type 2 decimator that allows for the new high-speed and low-overhead DelSig ADC.

The DelSig ADC module requires another two digital blocks and one analog block for a single-order decimator. Nine-bit resolution (128× decimation rate) was selected in favor of the next step down (7.5-bit resolution, 64× decimation rate) to better fill the output resolution, which will be 8 bits (discussed later). One arithmetic shift right of the incoming 9-bit value and you have a full 8-bit sample.

With a 128× decimation rate and a clock rate of 4 MHz, the resultant sample rate is approximately 7.8 kHz. Not high fidelity by any means, but with some external R/C filtering, it sounds pretty good. It also manages to use the 4-MHz clock already present at every SC block due to the requirements of the mixer, for minimal resource consumption.

As a bonus, the DelSig module uses virtually no CPU overhead other than what is used to check for, retrieve, and use its data. So, plenty of PSoC processing power is left over between sample cycles. The mixer (not counting the adjustment knob scanning) requires no processing time. It also works passively with the PSoC's hardware functions.

DIGITAL AUDIO SAMPLES

The DSP output is not a traditional DAC or even a filtered PWM (although it is similar in implementation). The PSoC has a neat little module called the pseudo-random sequence generator (PRS). The PRS module does an excellent job of generating an even distribution of random numbers over a set range. It also has a handy output labeled "compare out," which allows access to a signal that will be high or low based on whether the currently generated number is above or below the compare value.

At first glance, this feature might get passed over without a second thought. Let's take a closer look at the compare out signal of a PRS module. With an input clock of 12 MHz, the PRS module would generate a random compare output at the same rate. This output would look a lot like random noise. However, if you were to take a simple R/C filter and apply it to the PRS compare out signal, it would result in a stable voltage. Thus, a PRS module's compare output may be treated as a PWM. There is only one conceptual difference between the two. A PWM's output is defined by its duty cycle or pulse width, when filtered, this becomes voltage. A PRS module's compare output is defined by its probability value (to be above or below the compare value over time); when filtered, this becomes voltage as well.

To be more specific, once the PRS module has been properly initialized and "seeded" for the desired range, its seed register becomes the compare

Sampling frequency	0 pages RAM	One page RAM	Two pages RAM	Three pages RAM	Four pages RAM	Five pages RAM	Six pages RAM
15.625 kHz	170 µs	16 ms	32 ms	49 ms	65 ms	80 ms	97 ms
13.393 kHz	170 µs	18 ms	38 ms	56 ms	76 ms	95 ms	113 ms
11.719 kHz	170 µs	21 ms	44 ms	65 ms	86 ms	108 ms	128 ms
10.417 kHz	170 µs	25 ms	50 ms	74 ms	98 ms	121 ms	145 ms
9.375 kHz	170 µs	28 ms	55 ms	82 ms	109 ms	136 ms	163 ms
8.523 kHz	170 µs	30 ms	60 ms	90 ms	120 ms	150 ms	180 ms
7.813 kHz	170 µs	33 ms	66 ms	98 ms	130 ms	162 ms	195 ms
7.212 kHz	170 µs	36 ms	73 ms	107 ms	141 ms	178 ms	210 ms
6.696 kHz	170 µs	39 ms	78 ms	115 ms	152 ms	190 ms	231 ms
6.250 kHz	170 µs	41 ms	82 ms	124 ms	163 ms	205 ms	245 ms
5.859 kHz	170 µs	44 ms	88 ms	131 ms	174 ms	220 ms	262 ms

Table 1—These are typical PSoC ADC sampling rates and resulting reverb delays.

register during runtime. For this application, the PRS8 module is initialized to use its full 8-bit range (as demonstrated in the PSoC technical reference manual).

Then, as samples are pushed through the FIFO buffer and need to be output, they are written directly to the PRS's seed register. No special timing or interrupts are needed to perform this update. The seed register may be updated with a new sample at any time, and the output probability will "fade" seamlessly to the new value. With a 12-MHz clock, this all happens plenty fast enough for audio.

ADD EFFECTS TO SAMPLES

The extra RAM in the CY8C29466 is needed for the effect. For most DSP effects, some sort of audio buffer is required to store a window of samples for processing. An echo effect can be added to a stream of audio by sampling the audio and repeating it after a short delay. The echo stream is the original stream offset by a fixed amount of time.

Reverberation is similar to the echo effect, except that the offset audio also gets attenuated, sampled, and echoed along with the original stream. This requires a decaying feedback loop, which can be switched in or out of the circuit to enable you to select whether you want an echo or a reverb effect in real time.

To achieve either of these effects, you need a buffer to store all of the samples in before it's time for them to be output. A special kind of buffer is needed to do this in real time in a streaming fashion. Enter the FIFO.

The FIFO buffer is an array of data accessed by two moving pointers: the Write pointer and the Read pointer. Initially, the array is empty, and both pointers point to the beginning of the buffer (the first byte, in the case of 8-bit samples). When data is entered into the buffer, it occupies the space pointed to by the Write pointer. Immediately afterwards, the Write pointer is incremented to point to the next byte in the buffer. Similarly, when data is read from the buffer, it comes from the location pointed to by the Read buffer, which is immediately incremented to point to the next location.

When either the Write or the Read pointer reaches the end of the buffer and is incremented, it wraps around to the first location. The result is a buffer that contains, at any given point (after it's been filled), one complete buffer worth of contiguous audio samples. When the buffer is completely full and streaming, the Write pointer should be trailing the Read pointer by one sample. Changing the sample rate or the size of the buffer adjusts how much time is stored.

With six of the eight RAM pages being used as audio sample buffers, setting the reverb to maximum will result in approximately 200 ms of reverb delay. Slower sampling rates will fill this limited buffer slower, resulting in more delay. Faster sampling rates will result in less delay (see Table 1).

As a side note, using the 7.5-bit resolution/64× decimation rate setting results in a 15.6-kHz sample rate, and the frequency response increases accordingly. However, this results in roughly half the maximum reverb delay, which is barely noticeable and not nearly as fun to play with. So 7.8 kHz it is! With some external memory and perhaps a larger package PSoC (with more I/O pins), higher sampling rates, and longer reverb delays, this method becomes usable.

The PSoC is also filtering the ADC samples as they arrive through an optimized infinite impulse response (IIR) filter algorithm. The algorithm counters some sampling noise by not allowing sharp spikes through.[1] This requires no digital or analog blocks to implement whatsoever, and is preferable in this case over a standard low-pass or band-pass SC filter module.

You may be wondering why a 0 delay is listed in Table 1 as shifting the output 170 µs from the input. This is the minimum time it takes for a sample to be fed through the software-only filter and through the FIFO buffer.

The unused control knob (RP_{15}) is used to adjust the reverb delay time. The new schematic is shown in Figure 1. A flowchart of the reverb algorithm is shown in Figure 2.

ADD PITCH SHIFTING

As I was having fun with the adjustable reverb, I noticed that there was plenty of ROM left over in the project. The reverb buffers had consumed most of the extra RAM. But when the reverb was not in use, all six buffers were open for another purpose.

As it happens, I recently took a Z8 Encore!-based project from an earlier issue of *Circuit Cellar* and converted its pitch-shifting algorithm into a

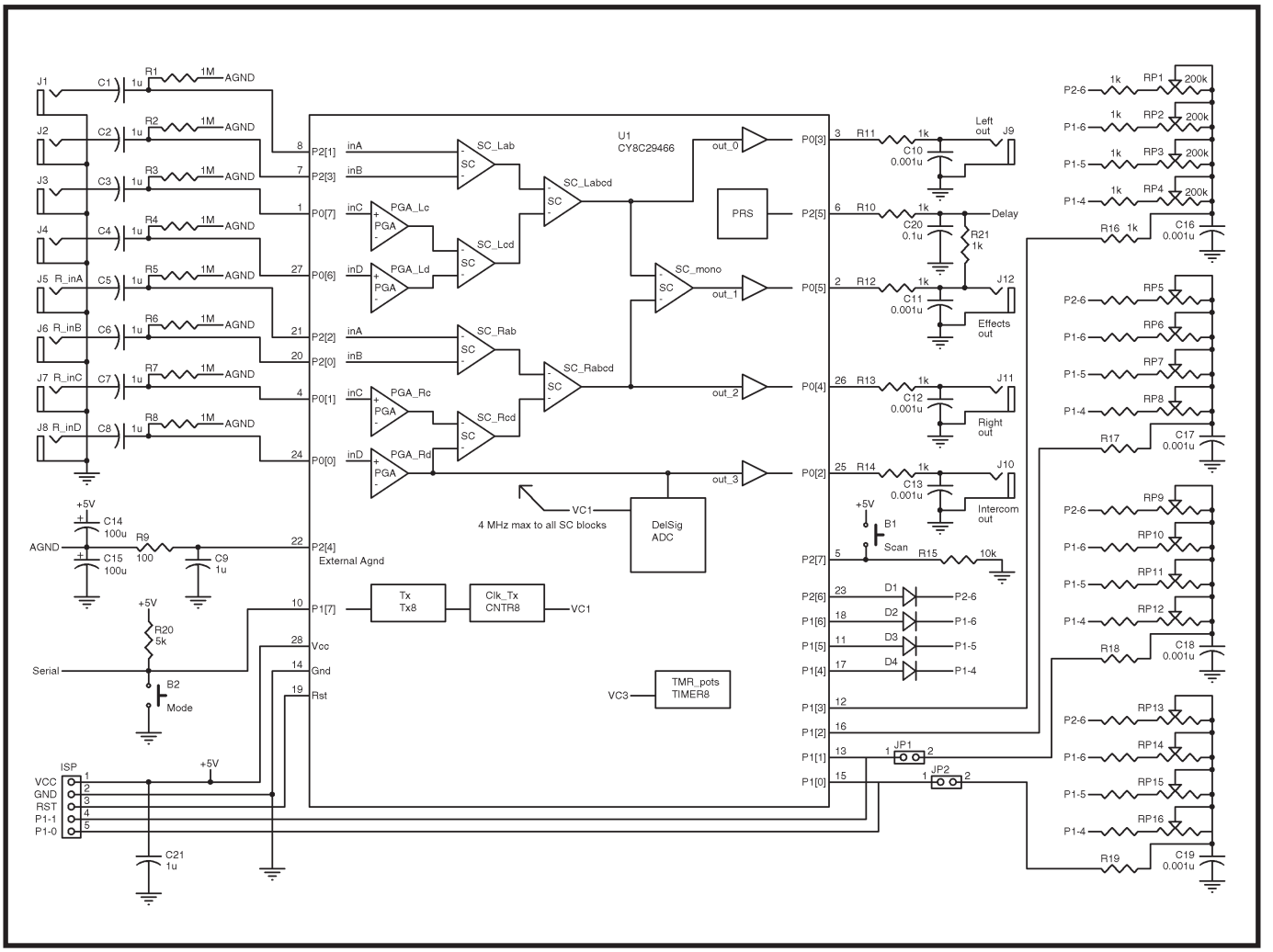


Figure 1—This is the revised mixer schematic. It now includes the additional circuitry to support new enhancements: DSP, intercom, WAV playback, and mode switching. Note that the component count of the project is still low, even after these enhancements.

PSoC project for fun.[2] It required only three pages of RAM to implement (two out-of-phase sample buffers and one triangle-wave buffer for cross-fading), and worked quite well with two buttons (one for up, the other for down) used to adjust the pitch offset.

It was no problem to integrate the pitch-shifting algorithm into the mixer/reverb project. To keep the intercom output pin (P0[2]) open, and because the two could feasibly coexist, the serial TX module shares its output pin with a mode button.

To accomplish this dual-purposing, an external pull-up resistor is required. This allows the TX module to output with an open-drain low drive instead of a strong drive, which does not affect an active-low button attached to the same line.

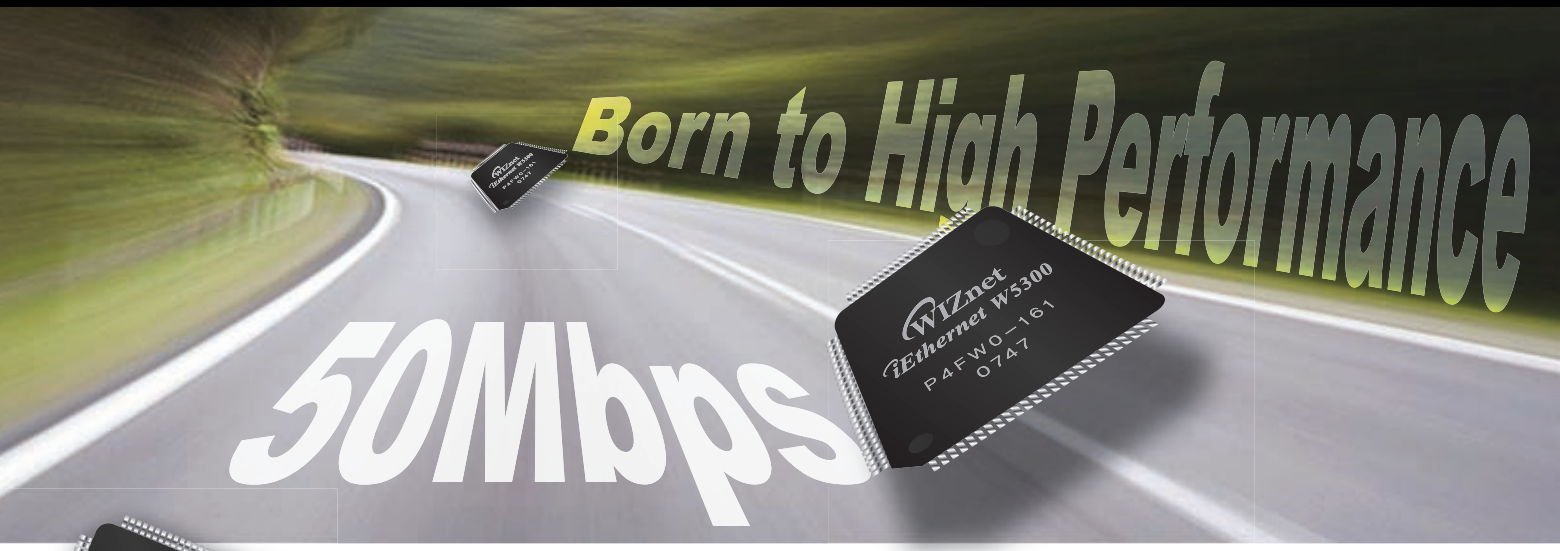
Basically, whenever the serial interface is not transmitting, the line is a standard High-Z digital input looking for a button press. When the serial interface needs to transmit, it connects the pin to the TX module and sets the Drive mode to Open Drain Low. It always waits for the button to be released before transmitting to avoid conflicts. The button presses will rarely, if ever, look like valid data to the serial terminal, so no erroneous characters appear.

When the button is pressed, the PSoC cycles through its modes: Reverb, Pitch Shifting, and Intercom. The latter was added because, well, it was possible. This mode disconnects the Rd signal from the mix so it can be accessed separately on P0[2], implementing a manual form of dynamic reconfiguration by altering the appropriate registers.

Reverb and Pitch Shifting modes each display a new title line on the serial terminal (because the last knob setting will change from Reverb Delay to Pitch Offset). Two new serial terminal example displays are shown in Figure 3. A pitch offset of 64 results in no pitch shift whatsoever. Settings from 65 to 128 result in an increase in pitch, and settings from 0 to 63 result in a decrease in pitch.

There is actually much finer resolution available to apply to the pitch-shifting algorithm. The input range from the knob timer is scaled to provide a usable pitch offset range to the algorithm. In the algorithm, pitch offsets of 0 to 255 result in lower pitch, 256 is for no shift, and 257 to 65,535 result in higher pitch (although it gets ridiculously and uselessly high after a couple thousand counts). However,

Exclusive Ethernet Highway





W5300

Fully Hardwired TCP/IP Ethernet Controller
designed for high-end embedded Internet system

Over **50Mbps!**
Dynamic Memory Allocation / 16 or 8 bit Data Bus Width

ioffload™ freeing CPU overhead and S/W TCP/IP bottleneck

• WIZ830MJ, Network Module with W5300



- Plug-in module
- Usable without H/W design for W5300, Transformer and RJ-45

• W5300E01-ARM



- ARM Based Evaluation Board for W5300's High-Performance

WIZnet 3rd Party e-market place - WIZWIKI.net

WIZnet

www.wizwiki.net
www.ewiznet.com
www.wiznet.co.kr

accessing all of these counts would require redesigning the knob timer to use a full 16 bits, as well as reselecting components to create an appropriate charge time to match the timer's new period. Or, two buttons to step up or down through the available pitch offsets. Both solutions seemed unnecessary for this project, which works out so nicely with the 0-to-128-scaled pitch offset range. Plus, it's a bit simpler to work with.

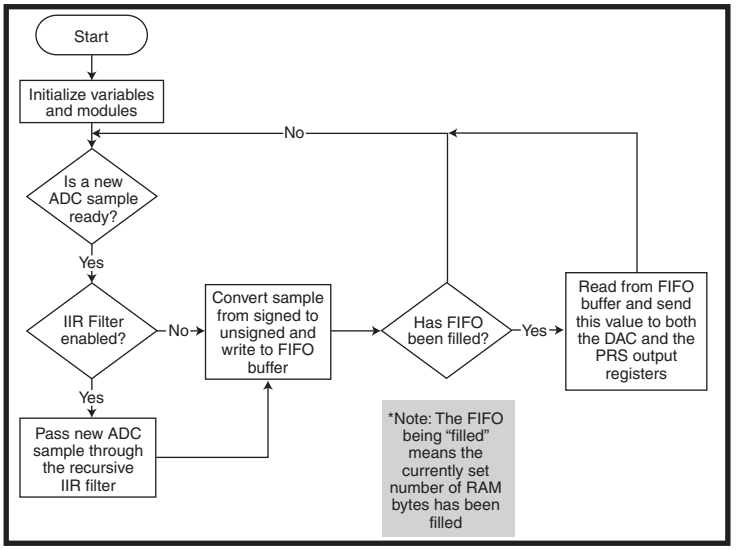


Figure 2—This flowchart illustrates the basic logic involved in creating a reverb DSP effect. The algorithm runs alongside the potentiometer scanning. It has been isolated in this flowchart for clarity.

ADD SPEECH

Suppose you don't have a serial terminal handy for your project, but you still want to know what mode is currently active. There is plenty of ROM left over after all of the aforementioned functionality has been implemented, and the required module (a PRS configured as a DAC) is present and available. This is sufficient to have certain events trigger WAV files to play through the reverb/pitch shift output channel.[3]

As fun as copyright infringement sounds, I could not bring myself to download and incorporate sound clips from my favorite TV shows into this project. Instead, I used my sound card to make my own recordings.

I started with the three modes the mixer would have available to it. I recorded the words "Reverb," "Pitch Shift," and "Intercom." Next, I time-compressed them to play back faster without changing pitch (a function of Sony's Sound Forge 8.0, which I used for the recording and manipulation) to save some space and make sure they still sounded like words. I then converted them down to 8 bits, and resampled digitally to 7.8 kHz to match the rate present inside the PSoC. Using this sampling rate enables the implementation of the WAV player without the need for a separate counter or timer module to generate the sample output interrupts. Again, this is not high fidelity by any means, but it is reasonably intelligible, as well as not being too tough on the ROM.

Using my Wav2H conversion application posted on the *Circuit Cellar* FTP site (and detailed in Cypress Semiconductor's application note AN13945), I created byte arrays to insert into the mixer project. I then piggybacked the playback loop on the ADC polling. I simply ignored the incoming samples and instead streamed from an array. Now, when the unit changes modes, it speaks.

RETAIN USER SETTINGS

Suppose you wanted the ability to store and retrieve sets of potentiometer positions from internal flash memory at will, thus allowing multiple users to save their preferred settings? This is possible within the project you have developed thus far.

You probably realize that you are completely out of I/O pins at this point on the PSoC. This is true. How then will you issue the commands to save and retrieve sets of potentiometer positions?



BitScope USB Mixed Signal Oscilloscope

Analog + Digital

Digital Storage Oscilloscope
✓ Dual Channel Digital Scope with industry standard probes or POD connected analog inputs. Fully opto-isolated.

Mixed Signal Oscilloscope
✓ Capture and display analog and logic signals together with sophisticated cross-triggers for precise analog/logic timing.

Multi-Band Spectrum Analyzer
✓ Display analog waveforms and their spectra simultaneously. Base-band or RF displays with variable bandwidth control.

Multi-Channel Logic Analyzer
✓ Eight logic/trigger channels with event capture to 25nS.

DSP Waveform Generator
✓ Optional flash programmable DSP based function generator. Operates concurrently with waveform and logic capture.

Mixed Signal Data Recorder
✓ Record to disk anything BitScope can capture. Supports on-screen waveform replay and export.

User Programmable Tools and Drivers
✓ Use supplied drivers and interfaces to build custom test and measurement and data acquisition solutions.

Inventing the future requires a lot of test gear...

...or a BitScope



BS100U Mixed Signal Storage Scope & Analyzer

Innovations in modern electronics engineering are leading the new wave of inventions that promise clean and energy efficient technologies that will change the way we live.

It's a sophisticated world mixing digital logic, complex analog signals and high speed events. To make sense of it all you need to see exactly what's going on in real-time.

BS100U combines analog and digital capture and analysis in one cost effective test and measurement package to give you the tools you need to navigate this exciting new frontier.



Standard 1M/20pF BNC inputs

Smart POD Connector



Opto-isolated USB 2.0

12VDC with low power modes

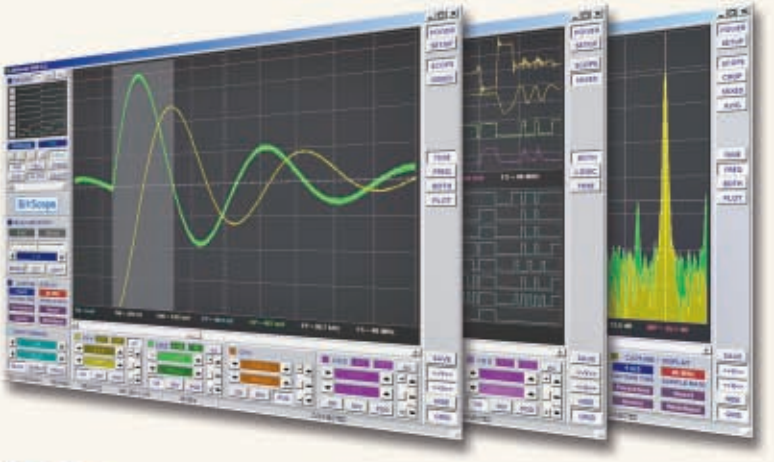
BitScope DSO Software for Windows and Linux

BS100U includes BitScope DSO the fast and intuitive multichannel test and measurement software for your PC or notebook.

Capture deep buffer one-shots, display waveforms and spectra real-time or capture mixed signal data to disk. Comprehensive integration means you can view analog and logic signals in many different ways all at the click of a button.

The software may also be used stand-alone to share data with colleagues, students or customers.

Waveforms may be exported as portable image files or live captures replayed on another PC as if a BS100U was locally connected.





www.bitscope.com

La	Lb	Lc	Ld	Ra	Rb	Rc	Rd	L	R	LR	MLc	MLd	MRc	MRd	Rvb
1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.14	1.14	1.14	1.14	064
La	Lb	Lc	Ld	Ra	Rb	Rc	Rd	L	R	LR	MLc	MLd	MRc	MRd	Pit
1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.13	1.23	1.23	1.23	1.23	042

Figure 3—These are new serial terminal output examples for reverb (top) and pitch shifting (bottom).

Well, you already have two buttons. They are being used independently, as well as without any type of tap/hold logic in place. How about this: A quick tap of the "Scan Pots" button will retrieve and cycle through the sets of potentiometer readings, while holding this button will result in normal adjustment operation with the potentiometers active. Then, if the "Mode" button is pressed while the potentiometers are active, the current potentiometer readings are written to the current Set. The resultant C main loop code that handles this overall logic is shown in Listing 1.

That logic handles all of the issues that arise with the suggesting of such a function. Initially, I was shooting for eight storage sets. At 16 bytes a piece, this would require only the last two blocks (64 bytes each) of flash memory to save. However, as I recorded the words "Writing," "Set," and the numbers "1" through "8," I realized I might be pushing the flash memory requirements.

Sure enough, even after all of these new sounds were time-compressed and converted to 8-bit/7.8-kHz format, the PSoC was still about 10 KB short. I needed to save some space. I quickened "Reverb" as much as possible without losing the pronunciation. I shortened "Pitch Shift" to "Pitch" and "Intercom" to "Com." I removed the word "Set," because it was unnecessary. I still did not have enough space. I was going to shorten "Writing" to "Save," but it would not have made much of a difference. At that point, I opted to drop down to four stored sets. Now, everything fits with room left over for the code necessary to implement it.

I created a constant array containing some default sets—set 1 has all unity gains with full reverb/max pitch shift, set 2 has slightly higher gains with lower reverb/pitch, and so on—to avoid undefined situations. I then ensured the compiler placed it in the last block of flash memory. Next, I wrote the logic necessary to store new settings in this location when the Mode button is pressed while the Scan Pots button is being held. The mixer speaks. For example, it will say "Writing 1" when storing Set 1. When retrieving Set 1, it will simply say, "1." The serial terminal reflects the retrieved settings.

HARDWARE/PCB

I used standard female RCA jacks as I/O because of their size, cost, and compatibility. I had ExpressPCB draw the schematics, lay out the PCB, and run netlist checks between the two. A view from the PCB editor for this project's circuit board is in the Printed Circuit Board Layout.doc file on the *Circuit Cellar* FTP site. If you are familiar with ExpressPCB, please note that this all fits into the inexpensive MiniBoard service.

I included an in-system programming (ISP) connector. It is standard practice with my designs, because

Listing 1—Here is the code used to handle the two button inputs. It determines whether the Mode button has been tapped or is held, as well as when both buttons are simultaneously pressed. This provides multiple user input functions with only two buttons.

```
while(1)
{
    DoEvents();
    if(CheckKnobEnableButton)
    {
        ButtHoldCount=0; ButtHeld=0;
//check for a tap or hold, here:
        while((CheckKnobEnableButton) && (ButtHoldCount<ButtHoldDelayCycles))
        {
            if(++ButtHoldCount>=ButtHoldDelayCycles) ButtHeld=1;
        }
        if(ButtHeld)
        {
            while(CheckKnobEnableButton)
            {
                ScanPots;
                if(GainChangeThisCycle)
                {
                        UpdateAllGains;
                        GainChangeThisCycle=0;
                        SendSerialNextChance=1;
                }
                if(++SerialOutputCounter>ReadingSetsBetweenSerialBursts)
                {
                        if(SendSerialNextChance)
                        {
                           TransmitSerial(); SendSerialNextChance=0;
                        }
                        SerialOutputCounter=0;
                }
//Now, here --- every cycle, check for the other button.  If both but-
tons pressed, its time to write!
                if(CheckModeButt)
                {
                        StoreSettings(SetIndex);
                        while(CheckModeButt);
                }
            }
        }
        else //tap
        {
             if(++SetIndex>4) SetIndex=1;
             LoadSet(SetIndex);
        }
    }
}
```

nobody wants to move the chip to test new firmware every reburn. Because this project utilized all I/O, two of the programming pins are used in the circuit (P1[0] and P1[1]). They are typically the last two pins you will use in a design, but because they are needed, they require two jumpers. The jumpers must be removed to burn the PSoC and installed to operate. (They connect two of the potentiometer input lines.) Jockeying two jumpers is still, in my opinion, preferable to jockeying a chip from socket to socket. The five-pin connector is designed for the Cypress Mini Programmer (CY3210-MiniProg1), which is an economical development PSoC burning tool.

EXAMPLE PROJECT NOTES

When the mixer first starts, it always loads Set 1 from memory and enters Reverb mode. (Thus, it speaks "1, Reverb.") When the Mode button is pressed, it cycles through the three available modes: Intercom (it says, "Com"), Pitch Shifting (it says, "Pitch"), and back to Reverb (it says, "Reverb").

When you tap the "Scan Pots" button briefly, the next Set is loaded from memory: "2," "3," and "4." After Set 4, it wraps around and loads Set 1 again.

When the "Scan Pots" button is held, the potentiometers are scanned and adjustments to the gains/reverb delay/pitch offset will be made accordingly. If the Mode button is pressed while the "Scan Pots" button is being held, the current readings from the potentiometers are stored in the current set in flash memory. If the most recently loaded set is 2, for example, this overwrites Set 2. (It would say, "Writing 2.") The next time Set 2 is retrieved, it contains these saved values.

FURTHER POSSIBILITIES

This was all accomplished using a 28-pin package. You might notice that this is not the largest package the Cypress CY8C29*xxx* family of PSoCs has to offer. By upgrading to a 44-pin or even 100-pin PSoC, all sorts of digital control and I/O functions become possible.

For example, while every analog block has been used, there are still some unused digital blocks. By simply cloning this project to a larger chip package (a handy feature of PSoC Designer), you gain the ability to add all sorts of communication schemes. The PSoC has user modules for standard serial communications, I²C, and SPI.

The pins may also be set up to bit bang custom or other communication schemes in or out of the chip. The main loop still has some processing time left over between handling either the reverb or pitch-shift algorithms and the potentiometer scanning, so much is possible.

With a communications scheme in place, you can control the mixer and reverb/pitch shift remotely from another console. Add a wireless PSoC to the design and you can have some wireless control over your mixer and DSP.

With a larger package, it would be relatively simple to add a standard LCD (with the LCD user module) to the project. The PSoC could control it directly. This would eliminate the need for a serial terminal to display the gains (although both could be simultaneously active).

It also goes without saying that you can add all sorts of buttons to a larger package. This would enable you to control more functions at will. You could use a separate Store Set button instead of the double-button press currently in place. Buttons to retrieve the next and previous sets would be good. Perhaps a separate button for retrieving each set would suit your particular design better. In such a configuration, holding said buttons could give the command to store to that particular set. Or, there could even be four separate Store Set buttons. The sky is the limit.

START YOUR PROJECT

The purpose of this series was to introduce you to some handy design techniques and teach you how to implement some common product features. You can now design an eight-channel mixer with 15 gain adjustments (all eight input gains, four microphone preamp gains, and three master gains), two selectable/adjustable DSP effects, four-set potentiometer position memory, an intercom, and voice menus. That's quite a mouthful. May this series provide a valuable starting point for your next (or first) PSoC project!

Author's note: Many of the application notes referenced in this article were recently submitted, so they may or may not be available on Cypress's web site at this time. For your convenience, the original, unedited versions of the relevant application notes are available on my web site, along with their project and related files. Go to www.chrispaiano.com for more information. In addition, I have a few homemade PSoC-based electronics kits and devices for sale on my site. The project described in this article is available as a kit or an assembled unit.

Chris Paiano has written more than 30 application notes for the Cypress PSoC chipset over the years, including such novelties as PongSoC and the Video RTA. You may contact him at chris.paiano@gmail.com.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

REFERENCES

[1] D.Van Ess, "AN2276: Algorithm-Binary-Weighted Single-Pole IIR Low-Pass Filters," Cypress Semiconductor Corp., 001-32902, 2005.

[2] B. Stewart, "Z8 Encore!-Based Audio Processor," *Circuit Cellar* 174, 2005.

[3] C. Paiano, "AN13945: Playing .WAV Files with a PSoC," Cypress Semiconductor Corp., 2007.

RESOURCE

Cypress Semiconductor Corp., "PSoC Mixed-Signal Array Technical Reference Manual," 001-14463, 2007.

SOURCES

CY3210-MiniProg1 Evaluation kit and CY8C29466 PSoC mixed-signal array
Cypress Semiconductor Corp.
www.cypress.com

Sound Forge 8.0
Sony Creative Software, Inc.
www.sonycreativesoftware.com



by Jeff Bachiochi

Water War Prevention

An MCU-Based Monitor For A Communal Well

Jeff's Microchip Technology PIC12F510-based system monitors the water filtration system in an underground well. The design indicates when the communal filtration system's brine tank is out of salt.

We are beginning to see some signs of public panic due to water issues. I have read about towns whose water sources (e.g., wells) are being threatened by pollution, both natural and man-made. I have also read about cases where upstream users are diverting increasingly large amounts of flow for their own (gluttonous) needs. Perhaps Mother Nature will choose to frustrate humanity by redistributing her life-giving precipitation. While solutions on a local level might be an initial necessity, regional, national, and even global resolutions might better promote unity. The Earth's resources do not respect our concocted territorial boundaries. Do we have the rights to water, coal, or any other resource under our feet? What about when a resource just passes through (e.g., a river)?

We must respect the planet and our neighbors, and we should be willing to compromise in order to avoid major conflicts. Unfortunately, some people don't think this way. As a result, it may not be long until a gallon of water costs as much as a gallon of oil. Only time will tell. I do know, however, that during the next few years engineers like us will be tasked with developing new systems to facilitate better water management, delivery, filtration, and monitoring. I recently used my engineering skills to tackle a local water issue. In this article, I'll describe how I built an MCU-based monitoring system for a water filtration tank.

WATER SUPPLY

Urban dwellers depend on municipal water works to provide an unending supply of fresh, albeit treated, water. As urban sprawl increases, more rural homesteaders are tied together by local community wells and even private single-home water systems. While a private system might be outside the jurisdiction of town regulations, I think it's safe to say that no one wants bad tasting, smelly, or toxic drinking water. As the landscape changes, we want to be assured that our water supply has not been compromised. Periodic water testing relieves these concerns.

Once we find that our underground aquifer is not being polluted, our attention shifts from supply to quality. Just because the water isn't toxic doesn't mean it is palatable. The rotten egg smell from sulfur, the laundry staining rust from iron, or the numerous other byproducts of natural but nontoxic minerals have a direct impact on the quality of the water we pump out of the ground. Extracting undesired material can be a fairly automatic process. Water filters/softeners mine minerals from the water by various means. Filtering is used to remove particulate. These can easily be seen and removed by simply straining the water, using a material with restricted openings. Filters are available down to 1 µm

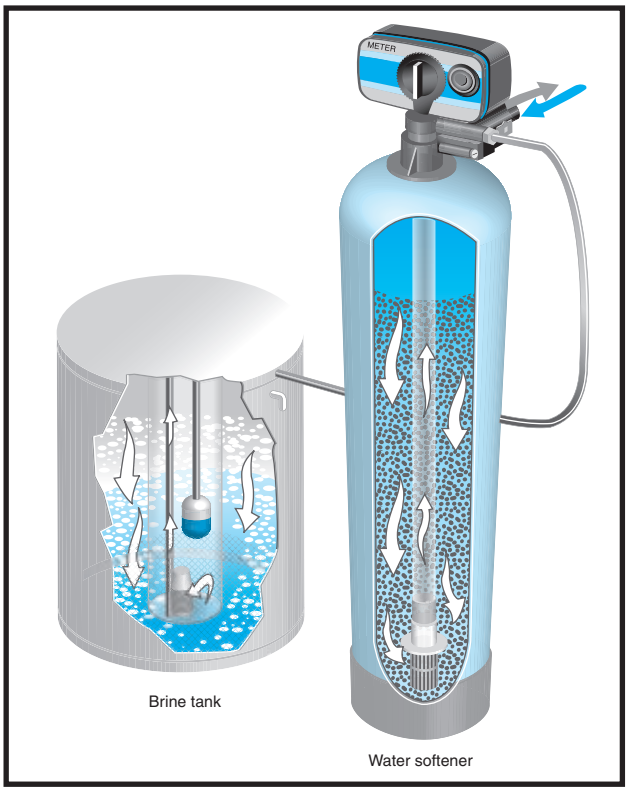


Figure 1—The basic automatic regeneration water softener system comprises a brine holding tank and a computer-controlled valve system atop a resin tank. The computer initiates a cleaning cycle based on water usage. A cam system opens and closes valves to route well water or brine (during cleaning) inputs through the resin tank to softened water or waste (during cleaning) outputs. (Source: Pure Water Products, www.pwgazette.com/howsoftenerswork.htm)

(0.01× the size of the human hair). The smaller the pores, the slower the flow through the filter, and the quicker it will become clogged (needing to be replaced). A filter of 10 or even 50 µm may be adequate unless you require protection against water parasites. Carbon is highly effective in removing chlorine, organic contaminants, chemicals, and undesirable tastes and odors. Like the filter cartridge, you must replace carbon periodically because it has a finite amount of absorption and will cease functioning once full.

As water travels through the Earth, minerals are absorbed and water becomes harder. The extent of hardness is measured in grains per gallon (GPG). A grain is a unit of weight (approximately 0.000143 lb, with 1 GPG = 17.1 ppm). Water with an excess of 5 to 10 GPG is considered very hard. The dissolved minerals can accumulate on surfaces in the form of a hard scale. The buildup will eventually clog pipes and may damage water-using appliances. These minerals also affect the ability of soap to clean surfaces, dishware, and laundry. While these are not generally harmful to the body, a water softener can remove them and protect your plumbing.

A water softener that will remove these is based on the exchange of ions. An ion is a molecule that has lost or gained one or more valence electrons, giving it a positive or negative electrical charge. My system has two basic parts, the brine tank and the resin tank with a controller (see Figure 1). The tall tank is filled with resin beads typically made from styrene or divinylbenzene. As well water passes through the tank, mineral ions are attracted to the resin, which in turn gives off salt ions. Like other filtering devices, once the surface area of the beads is covered with mineral ions, the exchange ceases. Unlike other filters, the system has the distinct advantage of being able to clean or regenerate itself via a periodic regeneration cycle. The tank's controller measures the amount of water used to determine when a cleaning cycle is necessary. When necessary, the controller initiates a cleaning cycle.

During a cleaning cycle, well water temporarily bypasses the filter while a brine solution—sodium chloride (NaCl) or potassium chloride (KCl) and water—flushes the mineral ions from the host resin. Thus, the resin is regenerated by the exchange of brine ions for mineral ions. The host resin itself does not need replacing. However, because the brine is removed and replaced with water each cleaning cycle, the NaCl or KCl in the brine tank needs replenishment. The brine tank can hold a few hundred pounds of NaCl or KCl pellets. When the brine is replaced with water, the pellets will be dissolved by the water until saturated, at which point the

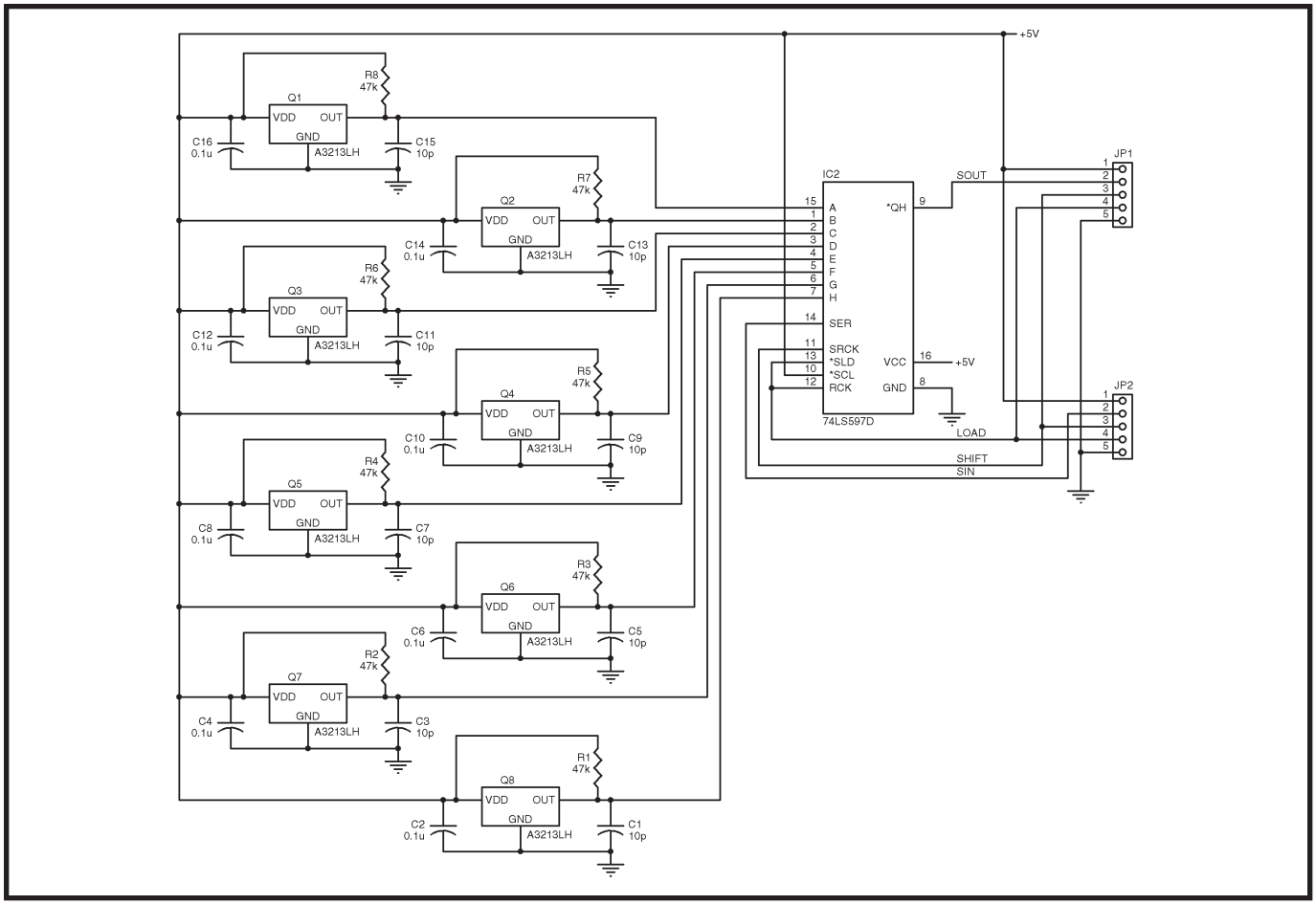


Figure 2—This is a modular shift register circuit that can be daisy chained to enable you to measure multiples of 8″. Eight Hall-effect sensors are equally spaced at 1″ intervals. The eight sensor outputs are latched and serially shifted via the sensor bus.

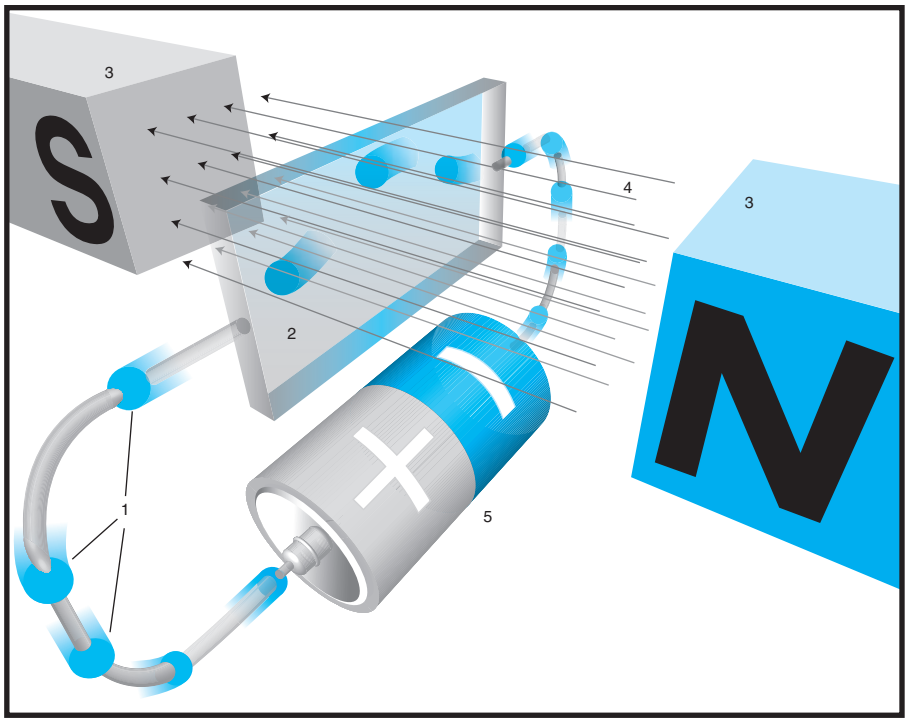


Figure 3—The Hall effect is a displacement of electron flow (1 and 5) within a conductor (2) due to exposure to an external magnetic field (3 and 4). A potential measured across the conductor is proportional to the magnetic field strength. (Source: Wikipedia, http://en.wikipedia.org/wiki/Hall_effect)

remaining pellets sit in the brine. Eventually, when all of the pellets have been dissolved and you've forgotten to add to the brine tank, the regeneration cycle becomes useless and the removal of minerals ceases.

My house and seven others in the neighborhood receive water through a single community well that just happens to be on my property. A perfectly working pump and water softener are key to keeping the neighborhood one happy family. Guess who gets the call when something acts up?

The pump, storage tank, and water softener are housed in a below-ground well house. Generally, this all worked without much intervention. Nevertheless, I came up with an idea several months ago. I wanted the ability to know—without having to periodically enter the dungeon pit—when the brine tank is empty and when the system fails.

TWOFER

Knowing the depth of the pellets left in the brine tank without having to venture underground is a big improvement. Because the water level in the brine tank isn't constant, weighing the tank to determine the potassium level won't work unless I know how much water is in the tank and can subtract its weight. Using electronics inside the brine tank isn't safe (due to corrosion), so I keep any electronics on the outside. I can sense the level of the water from the outside with a magnet floating on the water's surface. Sensors measure the water level. I use the same sensors to measure the position of a magnet that rests atop the pellets.

The system in Figure 2 is what measures the pellet and water levels. The brine tank has a float switch associated with the water refill cycle that prevents the water level from rising too high and overflowing the brine tank. This float is mounted inside a 4″ diameter pipe inside the tank. To measure the water level, I use a foam float with a magnet in the brine tank. To measure the pellet level, I use a piece of 0.5″ plastic egg crate with a magnet in the brine tank.

HALL MONITOR

In a previous column on electromagnetics, I discussed how a magnetic field is produced around a conductor that has a current passing through it ("Electric Motor Technology: Theory, Construction, And Requirements," *Circuit Cellar* 216, 2008). When this conductor is within an external magnetic field, the two magnetic fields create a force that tries to move the conductor. This force also has an effect (discovered by Edwin Hall) on the relative position of the current within the conductor (see Figure 3). If current flows through a conductor (x-axis) and you measure across the conductor (y-axis), you would measure no potential. As an external magnetic field increases through the conductor (z-axis), the current through the conductor would be forced off center and a potential would be measured across it. A Hall sensor is designed to measure

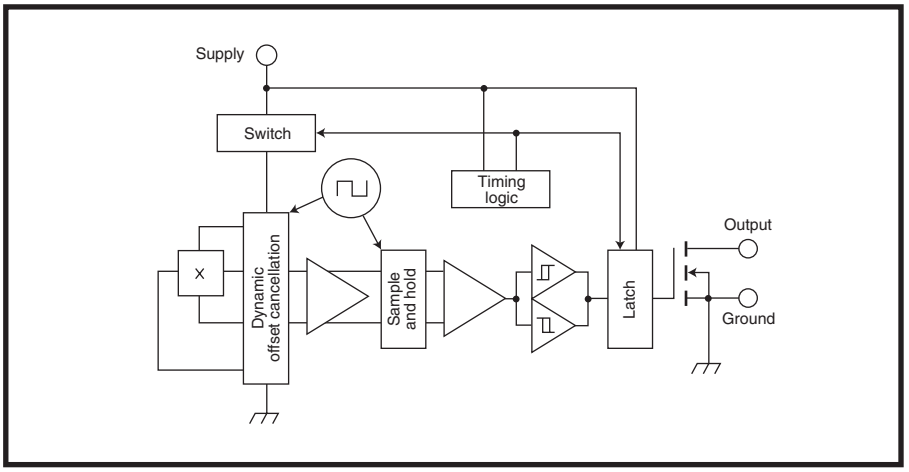


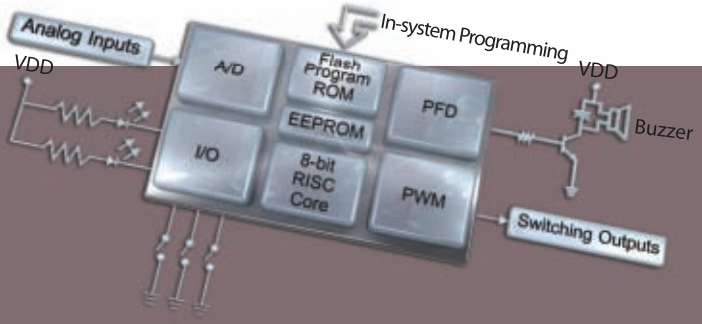
Figure 4—This Hall-effect device uses a bridge sensor and associated circuitry to indicate the presence of a magnetic field greater than 40 G. This is independent of the polarity. (Source: Allegro MicroSystems, Inc., www.allegromicro.com/en/Products/Part_Numbers/3213/3213.pdf)



HOLTEK
www.holtek.com

Imaginative Solutions

Flexible, large capacity and easy upgradeable intelligent controllers



Holtek's HT48FxxE and HT46F4xE series with their internal 8-bit RISC core, now provide customers with a wide and flexible range of devices for a huge variety of applications. These devices, with their large memory capacities and comprehensive functions can be used in all kinds of home appliances. Additionally, using Holtek's unique ISP, easy re-programming is provided for, without device removal, allowing for easy product upgrades.

- Wide operating voltage 2.2V ~ 5.5V
- Industrial specification -40 ~ +85℃
- Flash Memory can be re-programmed up to 100,000 times
- Data retention > 10 Years
- Excellent noise immunity and ESD protection
- ISP(In-System Programming) interface

Part No.	VDD	Clock	Program	RAM	EEPROM	I/O	Timer	PFD	Package
HT48F06E	2.2~5.5V	400K~12MHz	1K x 14	64 x 8	128 x 8	13	8-bit x 1	V	16NSOP, 18DIP/SOP, 20SSOP
HT48F10E	2.2~5.5V	400K~12MHz	1K x 14	64 x 8	128 x 8	19	8-bit x 1	V	24SKDIP/SOP/SSOP
HT48F30E	2.2~5.5V	400K~12MHz	2K x 14	96 x 8	128 x 8	23	8-bit x 1	V	24SKDIP/SOP/SSOP, 28SKDIP/SOP/SSOP
HT48F50E	2.2~5.5V	400K~12MHz	4K x 15	160 x 8	256 x 8	33	8-bit x 1 16-bit x 1	V	28SKDIP/SOP/SSOP, 48SSOP
HT48F70E	2.2~5.5V	400K~12MHz	8K x 16	224 x 8	256 x 8	56	16-bit x 2	V	48SSOP, 64LQFP

Part No.	VDD	Clock	Program	RAM	EEPROM	I/O	Timer	A/D	PWM	PFD	Package
HT46F46E	2.2~5.5V	400K~12MHz	1K x 14	64 x 8	128 x 8	13	8-bit x 1	8-bit x 4	8-bit x 1	V	16NSOP, 18DIP/SOP
HT46F47E	2.2~5.5V	400K~12MHz	2K x 14	64 x 8	128 x 8	13	8-bit x 1	9-bit x 4	8-bit x 1	V	16NSOP, 18DIP/SOP 20SSOP
HT46F48E	2.2~5.5V	400K~12MHz	2K x 14	88 x 8	128 x 8	19	8-bit x 1	9-bit x 4	8-bit x 1	V	24SKDIP/SOP/SSOP
HT46F49E	2.2~5.5V	400K~12MHz	4K x 15	128 x 8	256 x 8	23	8-bit x 1	9-bit x 4	8-bit x 2	V	24SKDIP/SOP/SSOP 28SKDIP/SOP/SSOP

HOLTEK SEMICONDUCTOR INC.

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

this effect using a thin current-carrying element in a bridge configuration. With a constant current flowing across the element in one axis (x), the other axis (y) will be balanced while no magnetic field is present. Any imbalance is proportional to the strength of the magnetic field passing through the element perpendicular to its other axis (z).

Additional circuitry within the Hall-effect sensor configures its function (see Figure 4). Hall sensors are available to measure the magnetic strength, to sense the presence of a magnetic field, or to flip-flop with a magnetic field's polarity reversal. Another import use for the Hall-effect sensor is to indirectly measure the current in a wire by measuring the magnetic field produced by the current in that wire. In this project, I want to sense the presence of a magnetic field from a magnet over a minimum distance of approximately 1″ to 2″.

For this operation, I chose the Allegro MicroSystems A3113, a micropower, ultra-sensitive, omni-polar Hall-effect switch. Typical switching characteristics (with hysteresis) are 40 gauss (G) on and 32 G off. Note that gauss is the unit of magnetic flux density (B). The Earth's magnetic field is approximately 0.5 G. The local hardware store had a small display of various magnets and I picked up a pair of ceramic magnets about the size of a domino. I connected one of the Hall-effect devices to a 5-V supply and monitored the device's open-collector output (with a 47-kΩ pull-up). I found the magnet triggered the device right around 2″ from it. Perfect. This verified that the devices were acceptable for my design criteria.

To prevent any contamination or reaction from the ceramic magnets, I dipped each into Plasti Dip, a liquid plastic coating used on many tool handles. This turned out to be more difficult than I had anticipated because of the magnet's strong attraction to the can's metal side. I taped a thread to the magnet so I could immerse it and hang it to drain and dry. The pull of these magnets simply ripped off the thread as it jumped to the can's side, creating the mess I was trying to avoid! I should have poured some of the liquid into a plastic cup first.



Photo 1—You can see the foam float I shaped inside the brine tank of our water softener system. The domino-sized magnet is glued along the right edge of the float. The tank's interior 4″ vertical pipe prevents the float from rotating out of alignment with the Hall sensor PCB mounted on the tank's exterior.

EGG CRATE AND FOAM

On my last trip to the hardware store, I seized the opportunity to search for the items I needed for this project. In the insulation aisle, I found some 1″ high-density foam. This became my brine float. This foam doesn't crumble like the block foam that comes as most packing material. In the lighting aisle, I found some 2′ × 4′ plastic egg crate grills for fluorescent fixtures. They became my pellet float.

Using the plastic brine tank's lid as a pattern, I cut a disk out of the foam and the egg crate. I also cut out a 4.5″ hole along one edge, which keeps the disks from spinning within the tank while floating atop the brine and riding atop the salt. After a couple of trial fittings in the tank, I attached magnets to the foam disk and the egg crate disk (see Photo 1). The magnets are vertically aligned to one another on each of the disks so when they are in the tank they pass directly behind the Hall-effect PCB strips mounted on the tank's exterior. The brine can easily pass through the holes in the plastic egg crate; the pellets cannot. Therefore, the egg crate rests atop all of the undisolved pellets (see Photo 2). While a cleaning cycle is active, all of the brine may be withdrawn from the tank. But it is quickly replaced by the end of the cycle. To prevent the two disks from attracting each other, I placed spacers on the egg crate to keep the two disks at a minimum of 3″ apart. The normal level for the water is quite high.

SENSOR STRIPS

To keep the maximum length of this narrow PCB reasonable, I used a length of 8.1″. This allows sensors to be evenly spaced at every inch and each PCB to report in a single byte (one sensor/bit/inch). Because the circuit allows daisy chaining, you can use a number of 8″ sections, as long as you tell the software how many

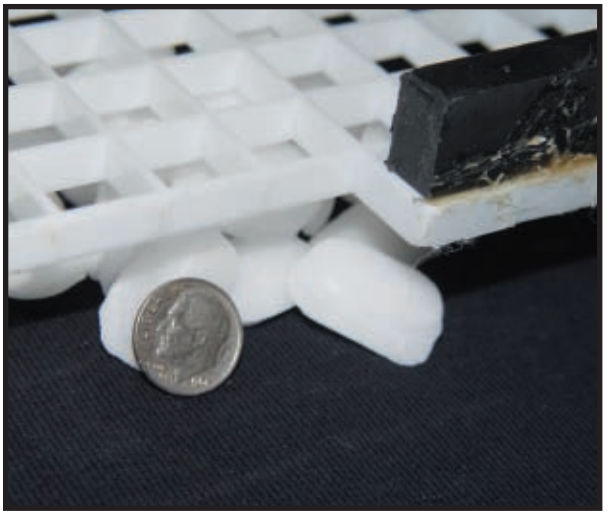


Photo 2—The undisolved NaCl or KCl pellets that remain in the brine are too large to fit through the plastic egg crate. This enables the egg crate to rest atop the pellets. It indicates the height of the remaining pellets.

The Newest Embedded Technologies



Flexis™ DemoQE128 Demo Board
www.mouser.com/freescale/a

New Products from:



VDIP1 Module
www.mouser.com/ftdi/a

FTDI Chip



RCM4000
RabbitCore® Modules
www.mouser.com/rabbitsemi/a

RABBIT Semiconductor



MatchPort™ b/g
Embedded Device Server
www.mouser.com/lantronix/a

LANTRONIX®

The ONLY New Catalog Every 90 Days

Experience Mouser's time-to-market advantage with no minimums and same-day shipping of the newest products from more than 366 leading suppliers.

MOUSER ELECTRONICS®

a tti company

The Newest Products For Your Newest Designs



www.mouser.com

Over A Million Products Online

(800) 346-6873

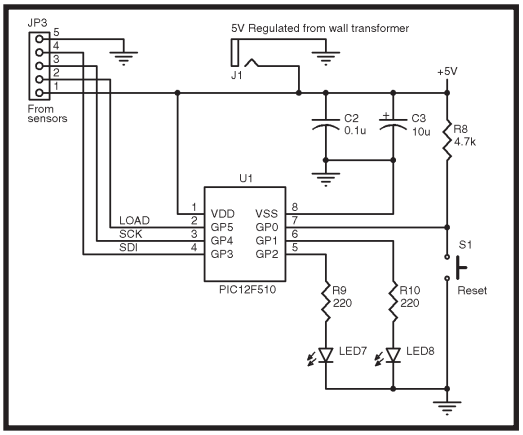


Figure 5—This application requires the simplest of microcontrollers. Activated sensor positions are indicated by blinking a code on the LEDs. A two-digit decimal blink code indicates the level of the pellet height (red LED) and the level of the water height (green LED).

boards (bytes) are in the chain.

Each PCB has its own parallel-to-serial shift register. The serial shift register bus contains five signals: 5 V, SDA, SCK, load, and ground. All eight parallel input bits are sampled using the load control line. The inputs can then be shifted out using a SPI (or in this case, the lines are bit banged). All Hall-effect outputs are high (1) unless they are sensing a magnetic field. Then the out is pulled low (0). The data is read most significant bit to least significant bit and refers to sensors positioned at 31″ down to 0″ (using four sensor boards and requiring 4 bytes).

A TINY APP FOR A TINY MICRO

This tiny application is written for a tiny microcontroller. At less than a buck, these eight-pin (and six-pin) devices can be used for many interesting projects. However, because they are inexpensive, they generally don't have many of the peripherals that larger devices have. The eight-pin Microchip Technology PIC12F510 microcontroller I used for this project does not have SPI support, so the serial shift routine must be software-driven (see Figure 5). The clock is toggled and data is read by assembly instructions as opposed to using a hardware peripheral that could do this in the background automatically.

Limited memory can be problematic for two reasons. The obvious problem: you have limited space in which to cram your application. Writing in assembler is sometimes the only way your application can fit because a higher-level language might require overhead that can put you over the space available. Another problem arises because they generally have few or no hardware peripherals. This means you can chew up precious memory just supporting that function, which then leaves even less for the application. This application requires only approximately 30% of the 1-KB program space available to the PIC12F510. That includes the software SPI and integer division routines.

The application periodically samples the Hall-effect sensors (by reading in 4 bytes of data) and stores the

No other PCB-design tool gives you more value per dollar

Boards designed under EAGLE are developed in one-man businesses or in large industrial companies. Most of the top companies are our customers. The crucial reason for selecting EAGLE is not usually the low price, but rather the high-end functionality along with the ease of use. And EAGLE users appreciate the outstanding level of support, which at CadSoft is always free of charge, and is available without restriction to every customer.

These are the real cost killers!

Version 5.0 is even easier to use, especially for beginners, due to an enhanced user interface.



New Version

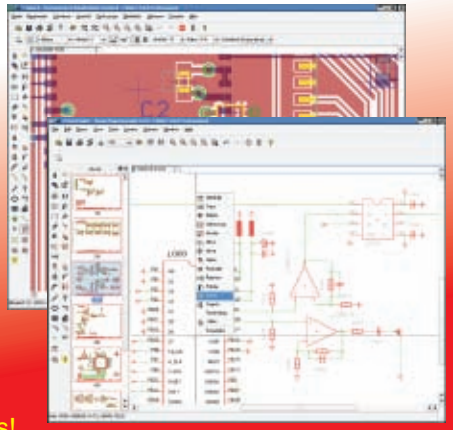
EAGLE 5.0

Schematic Capture • Board Layout
Autorouter

for Windows® Linux® Mac®

Version 5.0 Highlights

- Stand-alone schematic editor available.
- Automatic signal/contact cross references using frame coordinates.
- Right-mouse click for more consistent Windows UI.
- User-definable attributes for parts.
- Schematic sheet management.
- Hiding approved DRC and ERC errors.
- PRINT preview and text-searchable PDF output.
- Improved search engine for help.
- And much, much more.



Pick the level that is right for you — pay only the difference for upgrades!

	Light	Standard	Professional
Max. number of schematic sheets	1	99	999
Max. board size	4x3.2 inch	6.4x4 inch	64x64 inch
Max. # of signal layers	2	4	16
Layout or Schematic Editor		$249	$498
Layout and Schematic Editor		$498	$996
Layout Editor and Autorouter		$498	$996
Layout Editor and Schematic Editor and Autorouter	$49	$747	$1494

Standard and Light Editions have full functionality except for the limitations mentioned in the table.

You can use EAGLE Light for evaluation and non-commercial applications without charge. Download it from our web site.

www.cadsoftusa.com
800-858-8355

CadSoft Computer, Inc., 19620 Pines Blvd., Suite 217, Pembroke Pines, FL 33029
Hotline (954) 237 0932, Fax (954) 237 0968, E-Mail: info@cadsoftusa.com
Windows / Linux / Mac are registered trademarks of Microsoft Corp. / Linus Torvalds / Apple Computer, Inc.

data to a small table (see Figure 5). Two variables, `salt` and `water`, are used to indicate float levels. They are initialized to 32, one more than the maximum level of any sensor (in this case 32 sensors, 0″ to 31″). The table is scanned LSB (0) through MSB (31) to find a low. If a low is found, indicating the sensor is seeing a magnetic field, then the bit counter value is stored in either the `salt` or `water` variable. The `salt` variable is tested to see if it has been set. If any value other than 32 is found, the second variable `water` gets the counter value; otherwise, `salt` gets the counter value. This gives `salt` the first counter value when the first low is found (lower level) and `water` the last counter value when the second low is found (upper level). If a float magnet is seen by two adjacent sensors, the lowest sensor is used for the variable `salt` and the highest is used for the variable `water`.

The counts in `salt` are displayed by the red LED. The LED blinks once per second for each ten's digit and then again for each unit's digit. A digit of zero leaves the LED on for a longer period of time. The green LED then repeats the display of level count for the `water` variable (see Figure 6). The LEDs are mounted under the roof overhang outside of the locked well house so they can be viewed by looking out of my living room window. Now I can keep track of the system without having to go below ground.

SIMPLIFYING THE SIMPLISTIC

The basic premise of the project is to indicate when the water filtration's brine tank is out of salt. I could have done this with a single sensor properly positioned at the bottom of the tank, but this project gave me the opportunity to demonstrate the use of Hall-effect devices to do measurement and not just indication. While the actual level of water in the brine tank isn't important, measuring more than one level using the same strip of sensors is a good application. If the filtration system breaks down, the level of water in the brine tank will be an indicator of potential trouble. Because I know that a cleaning cycle normally takes place every "x" days, if the level of the brine hasn't changed in, say, 2x, then I know there must be a problem.

As it is, I use a regulated 5-V wall-wart supply to power this circuit (plugged into a GFI). If I control 5-V power to the Hall-effect sensors (leave them unpowered when not being sampled), the circuit current will be low enough to run with batteries. I could have used Allegro's A3214 for this project. Its less frequent sampling period (60 ms instead of 240 µs for the A3213) translates to a much lower operating current. But it wasn't available at the time I bought these parts.

I will let this circuit run for a while. I will watch its performance. I already have some thoughts on tying this information to other data I might collect in the well house. "Collect" being the key word here. I can think of a few technologies I want to investigate before taking a second look into the pit.

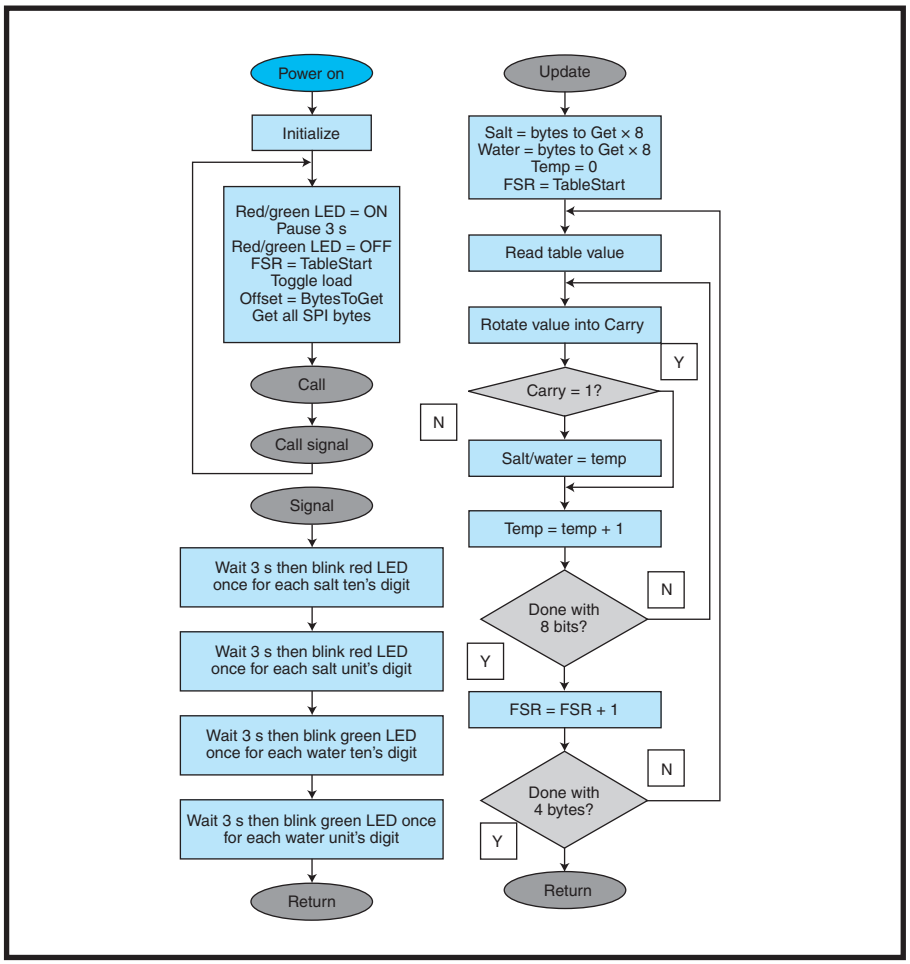


Figure 6—This application samples and reads the 32 Hall-effect sensor inputs via a software SPI routine. Two LEDs display pellet and water float heights by blinking a two-digit decimal code (1 = blink per inch).

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. He may be reached through the magazine (jeff.bachiochi@circuitcellar.com) or his web site (www.imaginethatnow.com).*

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

SOURCES

A3113 and A3213 Hall effect switches
Allegro MicroSystems, Inc.
www.allegromicro.com

PIC12F510 Microcontroller
Microchip Technology, Inc.
www.microchip.com



Direct Digital Synthesis 101

Need a refresher on direct digital synthesis? Robert brings you up to speed by covering DDS theory, a few chip-based solutions, and some firmware implementations.

Welcome to the Darker Side. If you are a regular reader, you probably remember my December 2007 article about using a phase-locked loop (PLL) to generate precise and stable frequencies ("Are You Locked?: A PLL Primer," *Circuit Cellar* 209). I also briefly introduced another interesting concept: the direct digital synthesizer, or DDS for short. At the time, I promised to dig into DDS techniques in a future column. This month, I'll make good on my promise.

You may also remember that I already presented a DDS project back in 2001, but this time I will go further than just describing a project ("DDS-GEN," *Circuit Cellar* 129 and 130). My aim is to help you understand how DDS techniques can help you in future projects. To do so, I will describe the pros and cons of using them. So, come with me on a journey to DDS world.

DDS BASICS

The simplest form of a digital waveform synthesizer is a table look-up generator (see Figure 1). Just program a period of the desired waveform in a digital memory (Why not an EPROM for old timers?), connect a binary counter to the address lines of the memory, connect a DAC to the memory data lines, keep the memory in Read mode, clock the counter with a fixed-frequency oscillator F_{CLOCK}, and voilà, you've got a waveform on the DAC output. Don't forget to add a low-pass filter to clean the output signal, with, as you know, a cut-off frequency a little less than $F_{CLOCK}/2$ to please Mr. Nyquist. This design works, but it is not too flexible. If you want to change the output frequency, you need to change the clock frequency, which is not easy to do, especially if you need a fine resolution.

The DDS architecture is an improvement on this original design (see Figure 2). Rather than add one to the table look-up address counter at each clock pulse like the counter did in the previous example, a DDS uses an N-bit long-phase register and adds a fixed-phase increment (W) at each clock pulse to this register. N can be quite high (e.g., 32 or 48 bits), so only the most significant bits of the phase register are used to select a value from the phase-to-amplitude look-up table, which is usually nothing more than a ROM preprogrammed with a sine waveform. Assume that you are using the P most significant bits as an address. Then the output of the look-up table is routed to a DAC. And, of course, the analog signal finally goes through a low-pass filter, which is called a "reconstruction filter." You will understand why in a minute.

How does it work? If the phase increment W is set to one, you will need 2^N clock pulses to go through all of the values of the look-up table. One sine period will be generated on the F_{OUT} output each 2^N clock pulses, exactly like the aforementioned counter-based architecture. If W is 2, it will be twice as fast and the output

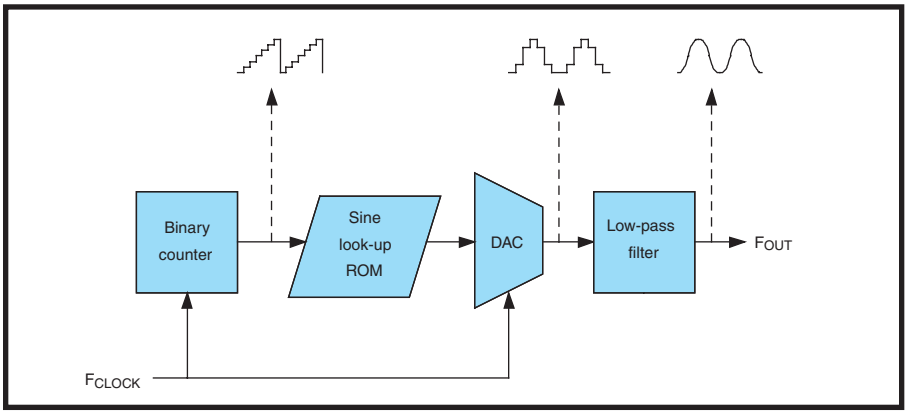


Figure 1—The most basic digital signal generator is built with a simple binary counter. Its output sequentially addresses the rows of a memory, which holds the successive points of the output signal. It is then converted to an analog signal and filtered.

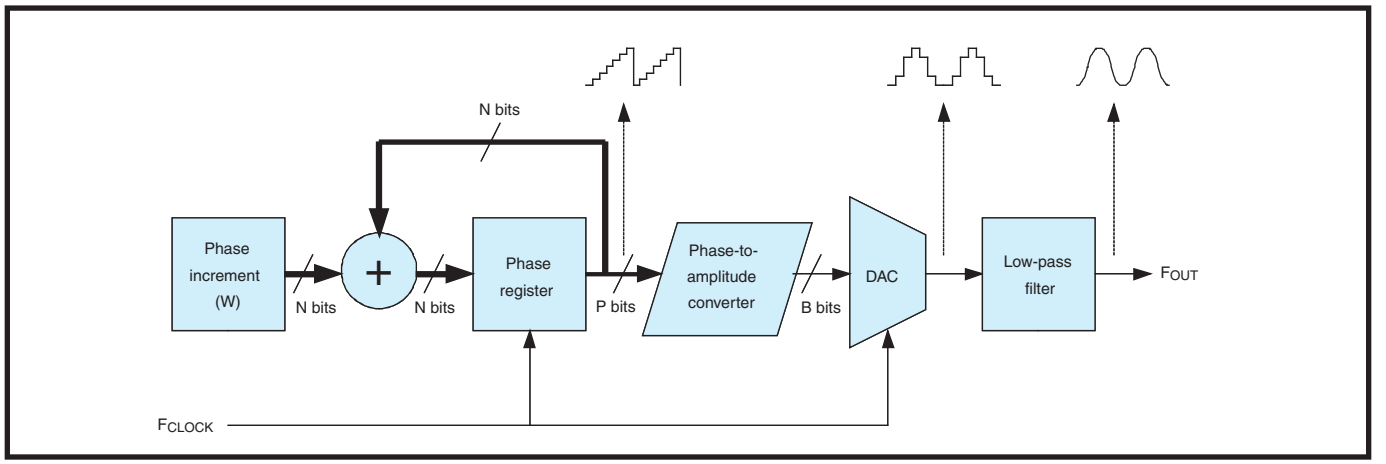


Figure 2—The basic architecture of a DDS is a variant of the counter-based digital generator, but it allows a fine frequency resolution thanks to a phase register and a binary adder. The key point is that the increment is not necessarily a divider of the phase register maximum value.

frequency will be doubled. As you know, you need a little more than two samples per period to be able to reconstruct a sine signal, so the maximum value of W is $2^{N-1} - 1$. The formula giving the output frequency based on the phase increment is then:

$$F_{OUT} = W \times \frac{F_{CLOCK}}{2^N}$$

Don't be confused. It is not a simple programmable divider because the phase register doesn't loop back to the same value after each generated period. The table in Figure 3 may help you understand it. What make a DDS a fantastic building block are the numeric examples. Just take a standard, low-performance DDS with a phase register of N = 32 bits and a reference clock F_{CLOCK} = 20 MHz. Your DDS can then generate any frequency from DC to nearly 10 MHz with a resolution of the following:

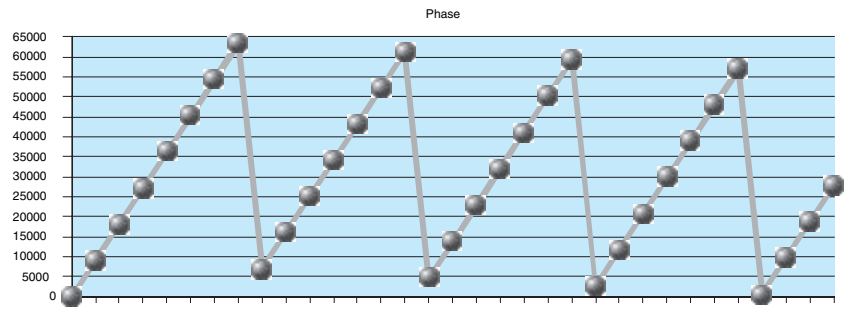
$$1 \times \frac{20\ \text{MHz}}{2^{32}} = 0.0046\ \text{Hz}$$

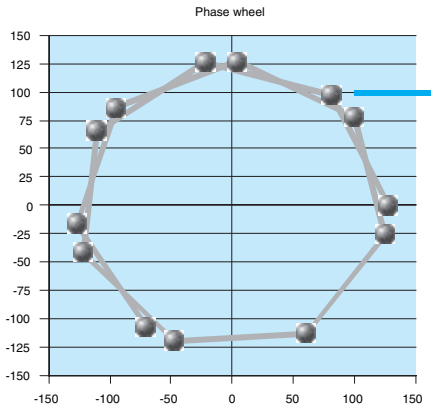
Not bad. In fact, the maximum frequency will be a little lower due to constraints on the low-pass filter, as you will see later.

DDS FLEXIBILITY

Another great advantage of a DDS generator is that you can use it for any

DDS Simulation	
Phase register length	16 bits
Phase maximum value	65,536
Phase increment	9,058
Reference clock frequency	1.00×10^6 Hz
Reference clock period	1.00×10^{-6} s
DAC Width	8 bits
F_{OUT}	138,214.11 Hz





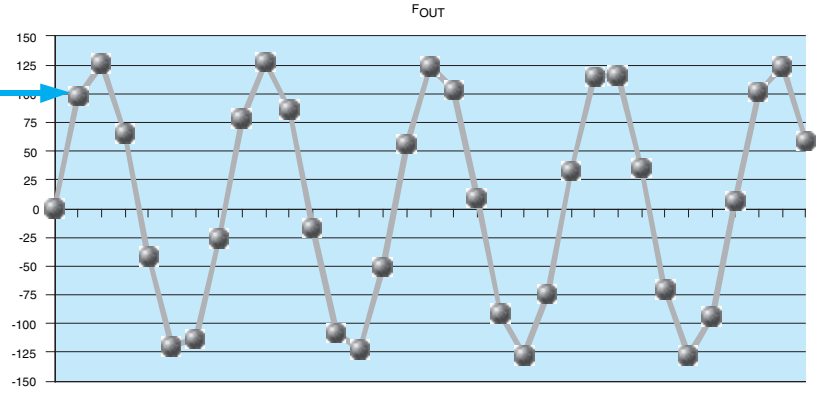


Figure 3—This spreadsheet simulation shows the "phase wheel" concept. A fixed angle is added to the phase register at each clock pulse. Note that each period of the output signal is not identical to the previous ones because the phase doesn't go back to the same value after a full turn.

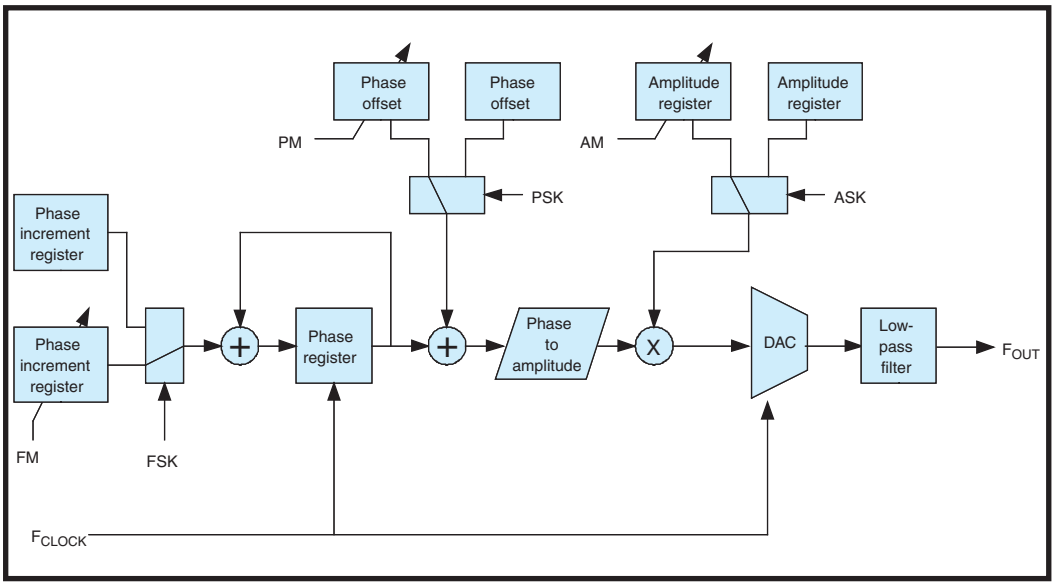


Figure 4—A DDS generator can be easily improved to add full digital modulation features, including either frequency, phase, or amplitude modulations.

kind of modulation, still fully in the digital domain. Refer to Figure 4, which shows a little enhanced DDS architecture. With a DDS, you can easily change the output frequency on the fly without any delay or phase shift just by loading a new value in the phase register or switching between different phase registers for FSK-like transmissions. You can also add a fixed value to the phase register independently from the DDS itself, which is ideal for phase modulation or PSK. You can add a digital multiplier before the DAC to implement software-controlled amplitude or AM modulation.

You can generate waveforms other than sine just by loading a period of your designed signal in the look-up table. But in that case, be careful, because you will be drastically limited in terms of maximum frequency. Due to the mandatory output low-pass filter, all harmonics above the Nyquist limit of $F_{CLOCK}/2$ will be filtered out. So, for non-sine signals, you will be limited to output frequencies low enough to ensure that all harmonics required for a good generation of your signal are significantly below this limit, which usually means frequencies not above $F_{CLOCK}/20$ to $F_{CLOCK}/100$. For example, look at the datasheet for a lab-class arbitrary signal generator like the Agilent Technologies 33220A, which is 50 Msps. It states maximum sine frequency = 20 MHz and maximum triangle frequency = 200 kHz.[1] Now you know why. If you need to generate a square signal, you will not have these limitations because you can generate a sine and add a simple comparator to extract a square signal with the same frequency.

There are a lot of other possibilities thanks to the digital structure of a DDS, and silicon makers are imaginative in these areas. You will see

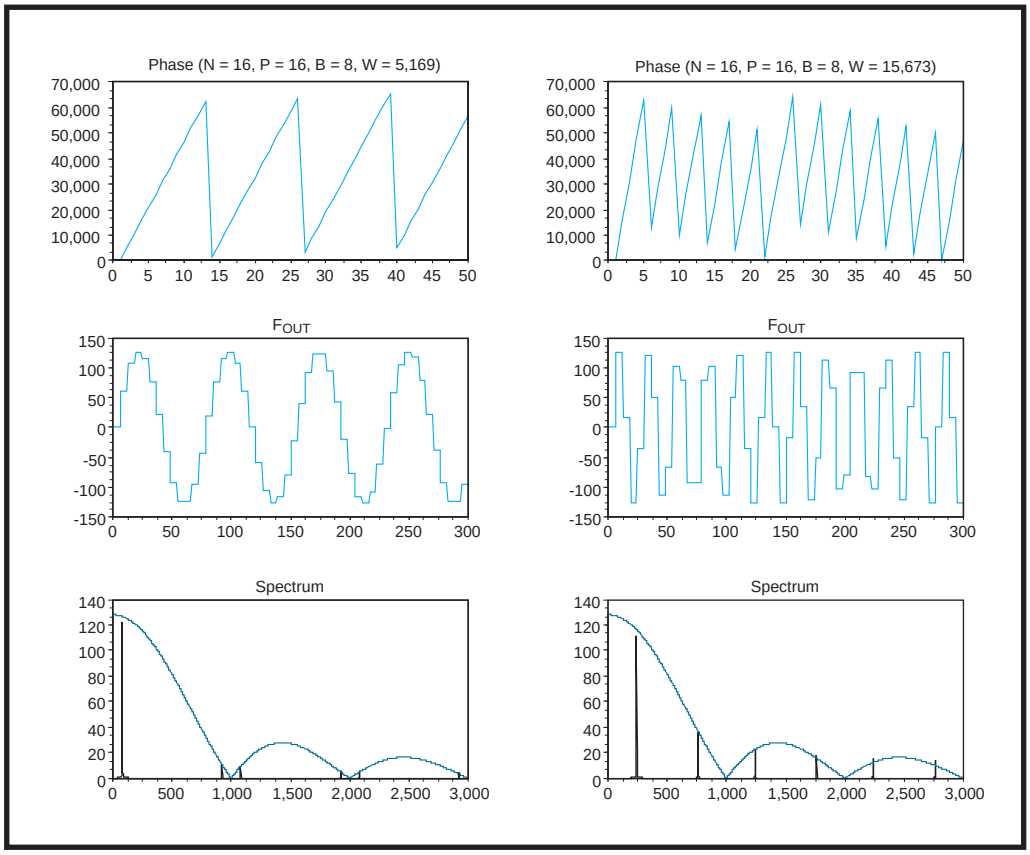


Figure 5—This Scilab simulation shows the phase, output signal, and output spectrum of a 16-bit DDS clocked at 1 GHz, with two different tuning words (F_{OUT} = 78 MHz on the left and 239 MHz on the right). The amplitude of the fundamental frequency gets lower when the frequency increases—following a sin(x)/x curve (dark blue)—and image frequencies get more powerful and unfortunately closer to the desired frequency.

some examples later on.

WHO SAID sin(x)/x?

You have now discovered all of the key advantages of a DDS architecture, but what are the difficulties that you may encounter? First, you have to look at the low-pass reconstruction filter again. Why do you need it? Because the output of the DAC is not a sine signal but a succession of steps that match a sine curve only at the clock-edge events, even if you assume that there are no other sources of error elsewhere. In the frequency domain, this means that the spectrum of the output signal will not be a simple fundamental F_{OUT}, but a more complex signal. I used SciLab, a Matlab-like open-source tool, for a simulation (see Figure 5).

There are image frequencies in the output. You get not only the frequency F_{OUT}, but also $F_{CLOCK} - F_{OUT}$ and $F_{CLOCK} + F_{OUT}$, and even $2\ F_{CLOCK} - F_{OUT}$ and $2\ F_{CLOCK} + F_{OUT}$, and more. The respective amplitudes of these image frequencies follow a curve mathematically defined as sin(x)/x, which happens to be the Fourier transform of a single step of width $1/F_{CLOCK}$.

But there is another problem. When your output frequency goes higher and higher, the power of the image frequencies gets higher too. Power needs to be found somewhere. This implies that the power of your desired F_{OUT} signal becomes lower and follows the same sin(x)/x curve shown in Figure 5.

This leads to two problems. One, you need to know (and compensate for if necessary) the reduction of signal amplitude when the frequency goes closer and closer to the Nyquist limit, at which point the theoretical power reduction is 3.92 dB. Two, when you come close to this limit, the first image frequency, which you need to cancel out with the low-pass filter, comes closer to your desired frequency and, worse, at a similar amplitude. Because the required low-pass filter would be impossible to build, you can't actually generate a signal arbitrarily close to the $F_{CLOCK}/2$ limit (see Figure 6). The usual reasonable limit is around 40% of F_{CLOCK} even with sharp filters.

However, nothing prevents you from using one of these image frequencies instead of the fundamental. Just replace the low-pass filter with a band-pass filter and you can use a DDS to generate a frequency higher than the Nyquist limit, far in the UHF area. The amplitude will be lower, but it will work as long as your filter is well designed.

ANY OTHER PROBLEM?

Once you have managed to filter out any image frequencies, will you get a perfectly clean sine signal? You will, but only if you have a perfect DDS with an infinite number of bits and infinite precision everywhere. Unfortunately, you are not that rich. One of your enemies will be DAC resolution. Because the resolution B of the DAC is not so high, there will be a quantization error, which will translate into quantization noise in the output spectrum. Once again, I have a small SciLab simulation with two different DAC resolutions (see Figure 7). The



theory says that the signal to total quantization noise power ratio is 1.76 + 6.02B dB, with B as the resolution in bits of the DAC. For example, with an 8-bit DAC, you can expect a 50-dB (i.e., 1.76 + 6.02 × 8) signal-to-noise power ratio. But that's just an average. However, there is a trick if the quantization noise is a problem. Because the noise is somehow spread from DC to the Nyquist limit, you can limit it just with a band-pass filter around your frequency of interest. If you reject all frequencies except a 10% pass-band around F_{OUT}, then the quantization noise will be divided by 10. Another solution is oversampling. If you increase F_{CLOCK} without increasing the low-pass filter corner frequency, the quantization noise will be lower in the filter passband too.

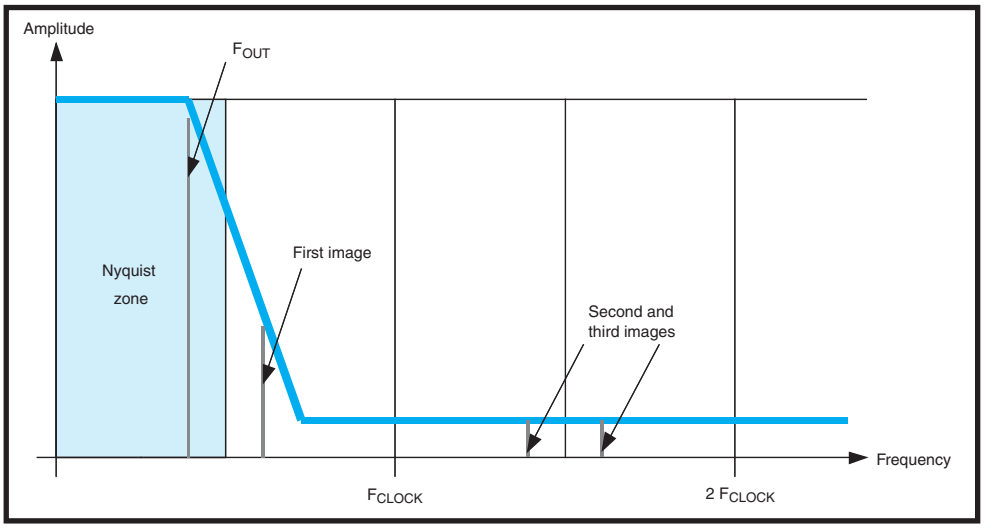


Figure 6—The role of the low-pass filter is fundamental. It must keep the fundamental frequency nearly untouched but provide a high attenuation on all image frequencies. That's why straight filters are usually required. It is also why the maximum frequency is usually 40% and not 50% of F_{CLOCK}.

DDS also has another issue that's often more crucial than quantization errors: phase accumulator truncation. The number of bits in the phase accumulator register is not infinite; furthermore, the number P of input bits in the look-up table is not infinite. This will give another error on the output. Contrary to DAC quantization, this error will not generate broadband noise but discrete spurious frequencies on the output spectrum. You may think of it as a miniature unwanted DDS generator working on the unused bits and unfortunately added to the output. Once again, the theory helps. It says that the relative

Cellular and GPS capable
Data Mover


• Flash file System
• 4 Chan 12-Bit A/D
• 1MB SRAM
• 512KB FLASH
• 4 Isolated Inputs
• 4 Hi Current Outputs
• 2 External RS-232
• 2 Internal RS-232
• Bat Backed Clk/Cal
• Cell Modem Option
• Internal GPS Option
• Metal Case Option
• 1ma Standby Option
It's easy and cost-effective to do mobile or solar-powered data collection and asset monitoring with the JK*micro's* **Data Mover**.
With the ability to integrate a Cellular Modem, GPS and DOS-based embedded controller in a single rugged enclosure, you can capture and transmit your data quickly, easily and at low cost. Inexpensive development kits including Borland C/C++ and PowerBasic are available now. Call or email us for more details.
Call **530-297-6073** Email **sales@jkmicro.com**
On the web at **www.jkmicro.com**
JK microsystems

WIRELESS RS-232
WCSC (Willies Computer Software Co)

* Extremely easy to install and use
* Distances up to 100 meters (329 feet), 500 meters with extended antenna (1640 feet)
* Bluetooth Serial Profile
* One to One virtual cable or selectively connect one to many.
* Communicates with Bluetooth enabled PDAs, Laptops, Smartphones & Computer
* No special software required!
Other Products
- Professional serial communication libraries & development tools.
- PCI, PCMCIA, Universal PCI, & ISA multiport RS232, Rs422, & Rs485 cards
- Bluetooth/USB Dongles
- USB/Serial Adapters
http://CircuitCellar.wcscnet.com
sales@wcscnet.com 281.360.4232

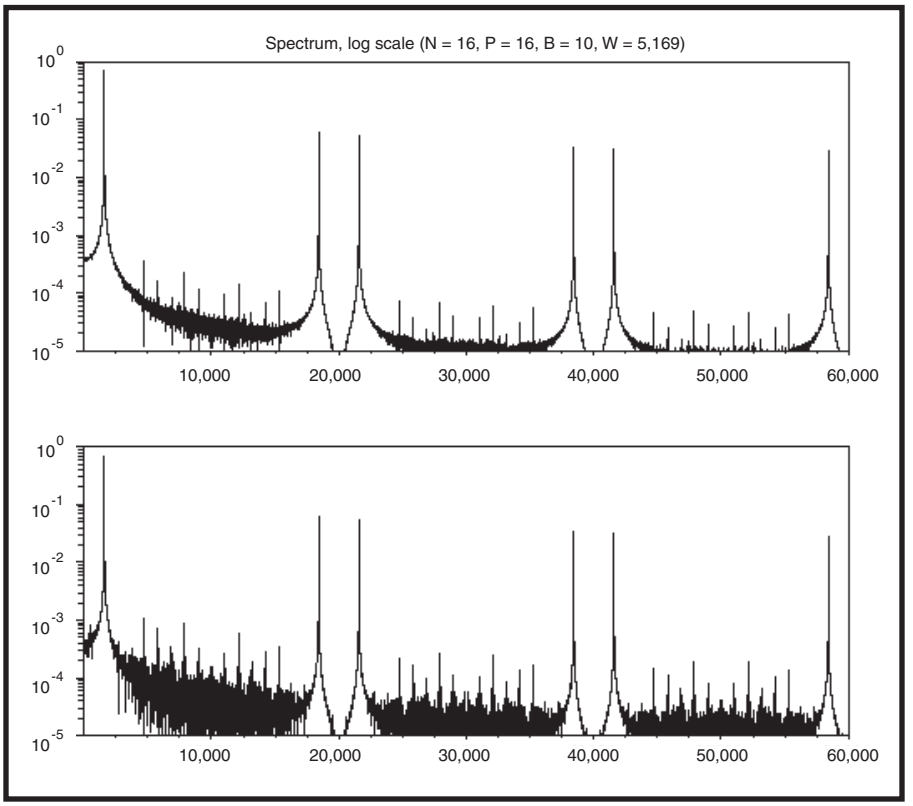
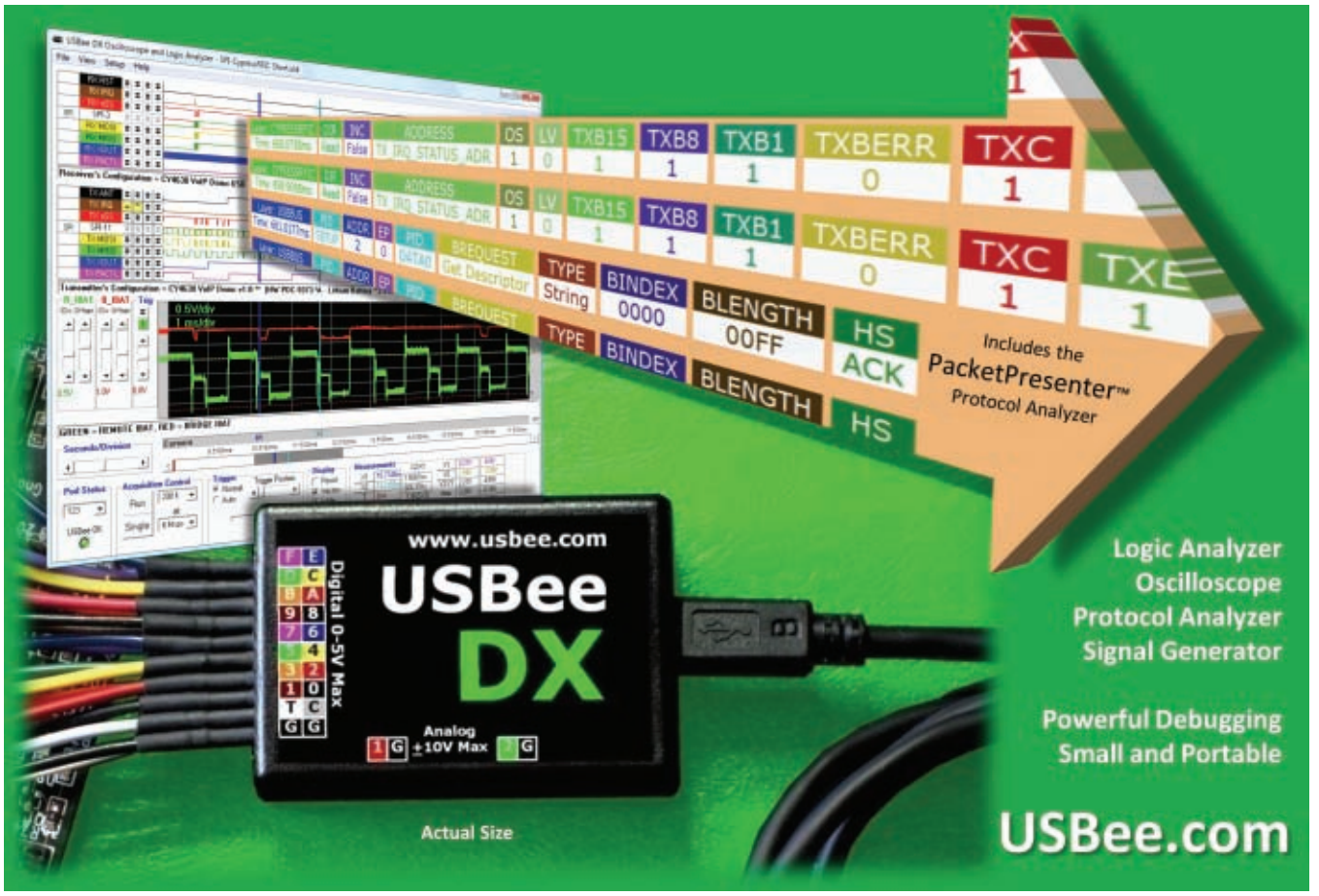


Figure 7—This simulation, still done with Scilab, shows DAC quantization effect. These are simulated output spectra of a 1-GHz 16-bit DDS tuned to provide a 78-MHz output. The top curve is with a 10-bit DAC. The bottom curve is with an 8-bit DAC. The vertical scale is logarithmic.

power of the largest spur is around –6.02 P dBc, with P being the number of phase bits after truncation. The difficulty is that the count, frequencies, and amplitudes of these spurious signals are dependent on both P and the phase increment value W. If you change the selected DDS frequency a little, the spurious frequencies will be drastically different (see Figure 8, p. 67). This behavior makes life for a DDS designer a little more complicated, but it is also a potential friend. If you have some flexibility in the DDS parameters, for example, and if you can have a slightly different F_{CLOCK} or F_{OUT}, then you may find another combination that gives fewer spurious frequencies (or at least less fewer spurs in a given frequency band). The good news is that the behavior of the DDS is predictable and some good simulation tools are available from chip manufacturers.

There are other sources of noise in DDS (clock jitter, DAC nonlinearity, clock feed-through, and more), but image frequencies, DAC quantization,



and phase-register truncation are usually the main contributors. However, don't conclude that DDS generates only noisy signals. These problems exist, but a good, well-designed DDS can have signal-to-noise ratios well above 70 dBc, large enough for the vast majority of applications. By the way, you will find two different figures in the specifications: the signal-to-noise ratio and the spurious-free dynamic range. They are correlated but not equivalent. The former is the ratio of signal power to the sum of all noises. The latter is the ratio of signal power to the strongest spurious frequency.

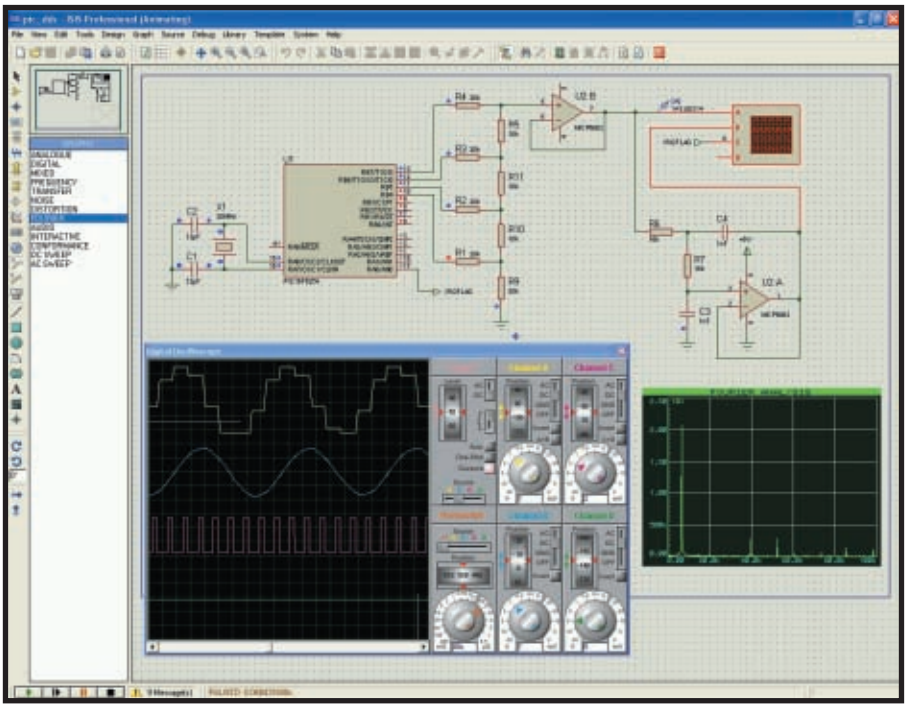


Photo 1—This is the Proteus VSM in action. The schematic shows you that I have used a PIC microcontroller, a small R/2R 4-bit DAC, and an active filter. Just virtually load the associated firmware in the microcontroller, connect a virtual oscilloscope or spectrum analyzer to the output, click "run," and you have the simulated output on the display. Impressive.

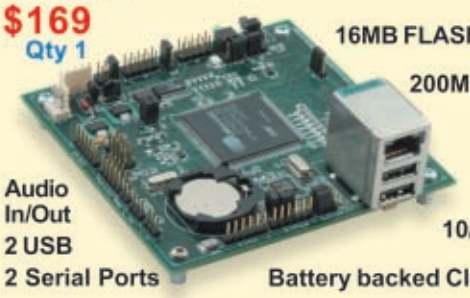
SOFTWARE IMPLEMENTATION

Enough theory. It's time to demonstrate how to build an actual DDS generator. There are some impressive dedicated integrated circuits around. I will examine them later. For now, let's start with a firmware-based implementation because this month's theme is Embedded Development.

Programming a DDS in a general-purpose microcontroller or DSP is often an effective solution for signal generation. Imagine that you are using a small microcontroller (i.e., a

Easy Embedded Linux

$169
Qty 1



16MB FLASH / 32MB RAM
200Mhz Arm9 CPU
16 Digital I/O
Watchdog
10/100 Ethernet
Battery backed Clock/Calendar
Audio In/Out
2 USB
2 Serial Ports

*We brought you the world's easiest to use DOS controllers and now we've done it again with Linux. The **OmniFlash** controller comes preloaded with Linux and our development kit includes all the tools you need to get your project up and running fast.*

*Out-of-the-box kernel support for USB mass storage and 802.11b wireless, along with a fully integrated Clock/Calendar puts the **OmniFlash** ahead of the competition.*

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

JK microsystems

CIRCUIT CELLAR®
back issues available as
Searchable Archives on CD-ROM

CD-ROM #12 2007 Issues 198-209
CD-ROM #11 2006 Issues 186-197
CD-ROM #10 2005 Issues 174-185
CD-ROM #9 2004 Issues 162-173

Order Online:
www.circuitcellar.com
or call 860.875.2199

Microchip Technology PIC16F629A clocked at 20 MHz) and you need to generate a 7,117-Hz signal, either sine or square. This is just an example, of course, but real-life applications can include DTMF generation, data rate generation, and similar problems.

The first idea that you, a firmware developer, will have will be to use the on-chip timer. Just configure a timer to count processor cycles (5 MHz maximum on this PIC variant) and toggle the output each N cycles. Calculate N for the output to be as close as possible to the required 7,117 Hz. Here you can have either 7,122 Hz (i.e., 5 MHz/702) or 7,112 Hz (i.e., 5 MHz/703). That isn't too bad, but it's quite far from the target, and you can't program a "fractional" count on a timer.

This is where DDS helps. Imagine another approach: configure the on-chip timer for an interrupt at any frequency but significantly above 2 × 7,117 Hz (e.g., 50 kHz). At each interrupt, add a fixed amount W to a 16-bit phase register, convert it to a sine using an 8-bit ROM-based look-up table, and send the value to a DAC. Then, filter it with a 10-kHz low-pass filter. Refer to the schematic in Photo 1, in which I have just used a simple 4-bit passive R-2R network as a DAC and a pair of Microchip MCP6002 op-amps as a buffer and low-pass filter. If you need a square signal, you can simply route the filtered signal back to the comparator available inside the PIC. The associated source code, fully coded in C using the free Hi-Tech Software PICC-Lite compiler, is available on the *Circuit Cellar* FTP site and is no longer than one page. You have built an actual DDS, and you can generate any frequency calculated as:

$$W \times \frac{50\ \text{kHz}}{65{,}536}$$

Thus, any frequency from 0.76 Hz to close to 20 kHz with a frequency step of 0.76 Hz! For example, just choose W = 9,328 and you get a frequency of 7,116.69 Hz. That's far closer to the 7,117-Hz target, isn't it? The magical trick comes from the fact that a DDS allows drastically finer frequency steps because the phase increment is not necessarily a sub-multiple of the period.

At this point, I can't resist telling you about a great simulation tool for mixed-signal designs. Labcenter Electronics's Proteus tool suite includes tools for schematic entry, Spice simulation, and PCB design. It also provides an impressive simulator named virtual system modeling (VSM) as an option. With VSM, you can simulate the code running on a microcontroller, like any firmware simulator, and the electronic circuits, like any Spice-like simulator, but you can simulate both simultaneously. Take another look at Photo 1. A virtual scope enabled me to verify the DDS signals

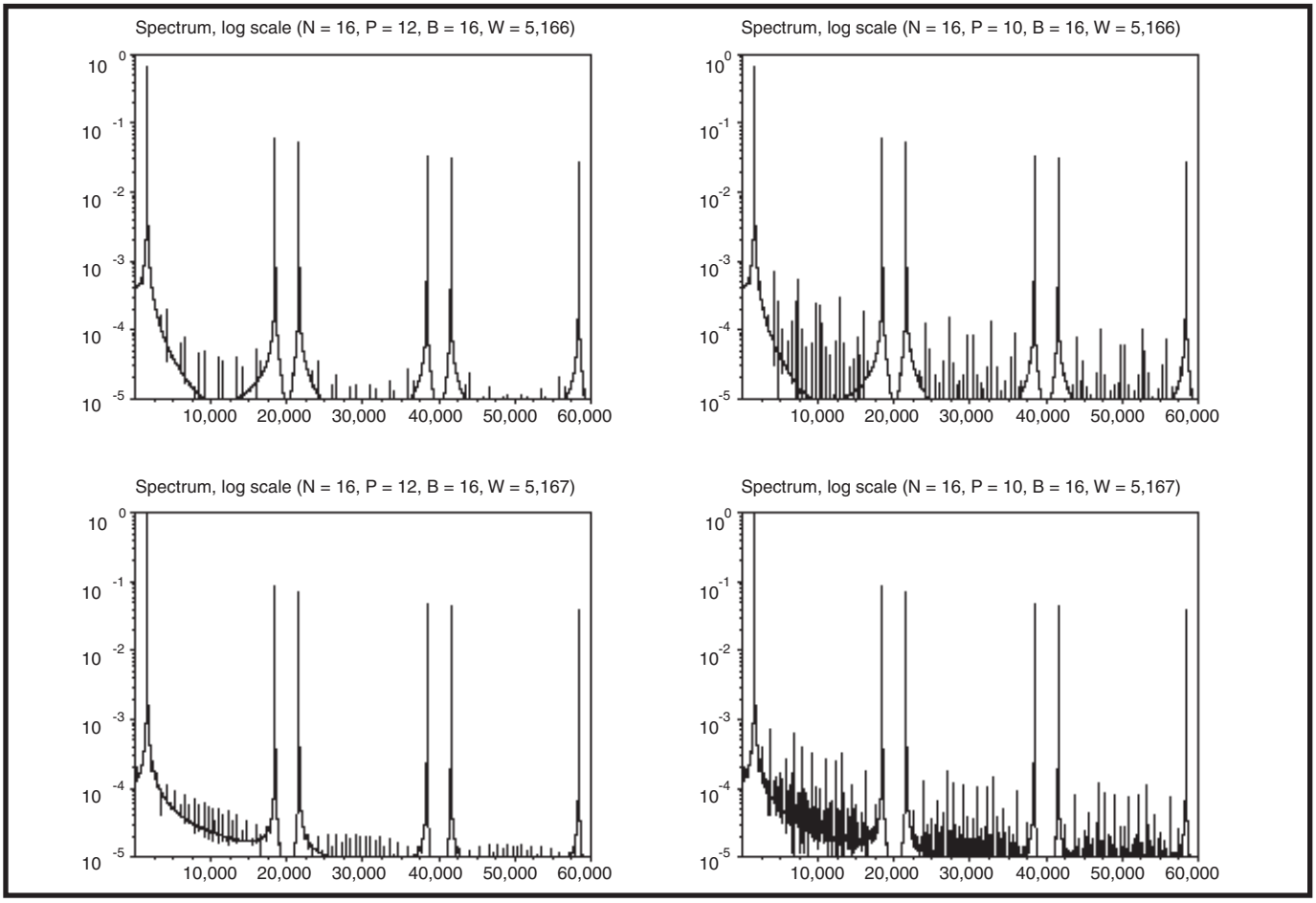


Figure 8—Here I'm illustrating spurious phase truncation. The left column is a DDS with 12 bits effectively used as a look-up table address. The right is with only 10 bits. The spurious spectrum is far more numerous and powerful in the latter. Finally, the bottom line shows what happens in the same condition with just a small change in the tuning word value (5,167 vs. 5,166). The spectrum of spurs is different.

generated by the PIC and filtered by the MCP6002, all without having to switch on the soldering iron! With the advanced simulation option, Proteus VSM can even calculate the spectrum of the filtered signal, as shown. Close to expectations, isn't it?

SILICON SOLUTIONS

Even if software-based DDS is possible, there are plenty of impressive silicon versions, particularly from Analog Devices. An example is an easy-to-use chip such as the AD9833, a $4 (qty. 1,000) low-power chip fitted in a 10-pin 3 mm × 3 mm MSOP package that enables you to generate frequencies up to nearly 12.5 MHz with 0.1-Hz resolution. The chip is driven by a standard SPI port, which enables you to connect it to any microcontroller. Since energy consumption is such an important consideration, note that this chip eats no more than 5.5 mA at 3 V, which is impressive for a 25-MHz chip. Technically speaking, it has a 28-bit phase register, a 12-bit look-up table, and a 10-bit DAC, which provides around 60 dB of signal-to-noise ratio. Now you will easily understand such a datasheet.

Now let's focus on a current, top-of-the-line DDS chip. The Analog Devices AD9910 depicted in Figure 9 is nearly 10 times more expensive than the AD9833. It costs around $35 (qty. 1,000) as I write this. But what a piece of silicon! First, its clock can be as high as 1,000 MHz, providing a useful output range up to 400 MHz. Providing a 1-GHz clock may be difficult, but these guys had the good idea to include an on-chip PLL to allow more reasonable external clock sources. Its 32-bit phase accumulator provides sub-hertz resolution, and it is equipped with a high-speed 14-bit DAC, enabling a spurious-free dynamic range up to 65 to 70 dBc, and still around –55 dBc at 400 MHz. But that was for the DDS core alone, and this chip has plenty of other blocks.

First, it has an auxiliary DAC to define the full-range amplitude without compromising the quantization noise. It can also automatically compensate for the sin(x)/x amplitude rolloff I discussed earlier, with a digital filter that has an inverse sin(x)/x response placed between the look-up table and the DAC. You can program eight different settings for frequency, phase, and amplitude, and then switch among them in nanoseconds via three external pins. If necessary, it can also automatically manage linear frequency, phase, or amplitude sweeps. In addition, it has a built-in 1,024 × 32 RAM that enables you to predefine custom frequency/phase/amplitude profiles and execute them at high speeds, which is perfect for generating complex

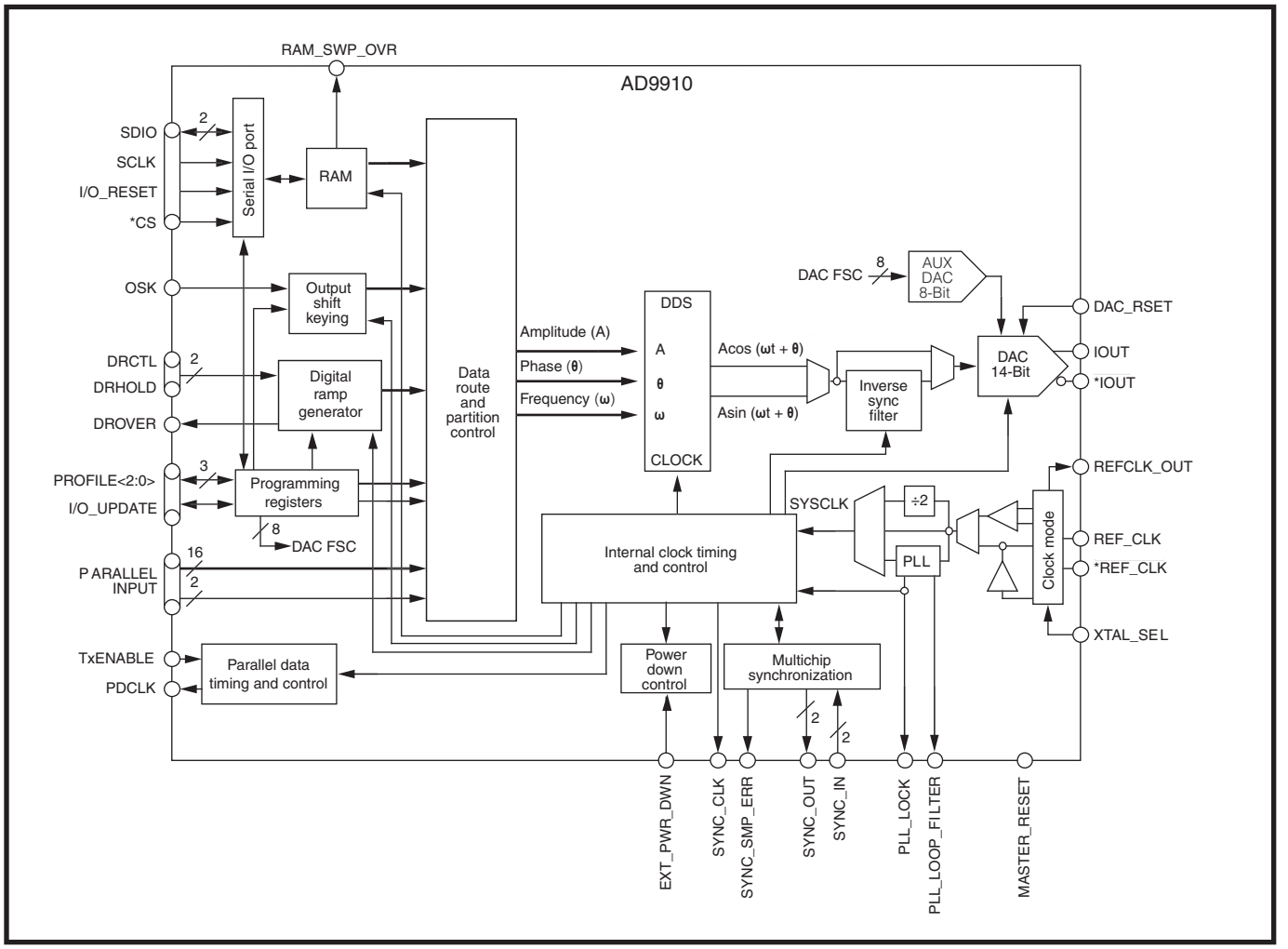


Figure 9—This is the internal architecture of the AD9910 high-end DDS chip (courtesy of Analog Devices). As you see, the DDS core is just a small part of the chip. It is surrounded by a zillion advanced high-speed digital modulation and control blocks, as well as a 14-bit Gsps DAC and a reference clock PLL multiplier.

modulated waveforms.

What else? Oh yes, it can be synchronized with other chips if its features are not enough for your application. It is managed through a SPI, but it also has a high-speed parallel bus for time-critical applications. OK, it is not a low-power chip (800 mW, 1.8 V, and 3.3 V), and you will have to solder its 100 pins and read its 64-page datasheet if you decide to use it, but that's the price you pay for such a list of features.

Another interesting chip is the Analog Devices AD9912. It is a 1-Gsps chip and has less modulation options, but it provides a 48-bit phase register and a 4-µHz resolution up to 400 MHz. (I'm not sure if such a resolution is useful. You should double-check the stability of your reference oscillator too.) Note that the AD9912 has an interesting new feature: SpurKiller channels. Theoretically, this feature will enable you to cancel any given pair of spurious signals. It is based on nothing less than two independent mini-DDS generators that can be tuned to generate a signal at the same frequency as the spurious one you want to kill, but in opposition of phase. The circuit then adds these signals on the output, all entirely in the digital domain, prior to the DAC. This feature seems to require delicate tuning. But a typical spurious reduction of around 6 to 8 dB is announced, with specific configurations providing up to 30-dB attenuation. Something to be bench-tested someday for sure!

WRAPPING UP

Here you are. You should now have a better idea about the pros and cons of direct digital synthesis. But let me summarize for good measure.

A direct digital synthesizer (DDS) will provide you with a marvelous sub-hertz frequency resolution, immediate frequency hopping, and efficient full-digital modulation features. However, its frequency range will be limited to around 40% of the clock source, except if you try to use image frequencies, and you may suffer from some nasty spurious signals on the output spectrum.

It is interesting to compare these characteristics with a synthesizer based on a PLL with programmable dividers. A integer PLL with its single divider can't have simultaneously fast tuning and a fine frequency resolution, which are always in opposition. Even with fractional PLLs that have two dividers, you will usually get only kilohertz-range frequency steps, and tuning to a new frequency will take tens or hundreds of microseconds. However, a PLL can be used to generate an output signal far above its reference frequency, and its output is usually clean, except for when it's close to the center output frequency or its harmonics. There aren't any "digital spurious frequencies" like with a DDS.

Based on this comparison, you will deduce that a PLL/VCO combination is usually more suited to local oscillators, where high frequencies and clean signals are a must. A DDS finds its key applications as a modulation source where agility is most important. However, DDS chips have gotten cleaner and cleaner over the years, and nobody would have imagined seeing a 1-GHz DDS chip for tens of dollars a couple of years ago.

For the best of both worlds, there are chips with both PLL and DDS cores. So, stay tuned, things can change quickly!

Now it's your turn. You should be ready to put a DDS in your next design, either as a piece of silicon or some lines of firmware. DDS is no longer on the darker side for you! ■

Author's note: I want to thank Labcenter Electronics and its U.S. and French distributors, R4 Systems and Multipower, respectively, who were kind enough to provide me with a Proteus VSM license for this project. Also, thank you to Jeff Keip and Eric Benoist at Analog Devices who were more than helpful (as usual)!

Robert Lacoste lives near Paris, France. He has 18 years of experience working on embedded systems, analog designs, and wireless telecommunications. He has won prizes in more than 15 international design contests. In 2003, Robert started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. You can reach him at rlacoste@alciom.com. Don't forget to write "Darker Side" in the subject line to bypass his spam filters.

PROJECT FILES

To download code and additional files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/217.

REFERENCE

[1] Agilent Technologies, Inc., "Agilent 33220A 20 MHz Function/Arbitrary Waveform Generator Data Sheet," 2005.

RESOURCES

Analog Devices, Inc., "A Technical Tutorial on Digital Signal Synthesis," 1999, www.analog.com/UploadedFiles/Tutorials/450968421DDS_Tutorial_rev12-2-99.pdf.

K. Gentile, D. Brandon, and T. Harris, "DDS Primer," Analog Devices, Inc., 2003, www.ieee.li/pdf/viewgraphs_dds.pdf.

SOURCES

33220A DDS-Based lab generator
Agilent Technologies, Inc.
www.agilent.com

AD9833 Waveform generator, AD9910 direct digital synthesizer, and AD9912 direct digital synthesizer
Analog Devices, Inc.
www.analog.com

PICC-Lite Compiler
Hi-Tech Software
www.htsoft.com

Proteus VSM Mixed-signal simulator
Labcenter Electronics
www.labcenter-electronics.com

MCP6002 op-amp and PIC16F629A microcontroller
Microchip Technology, Inc.
www.microchip.com

Scilab Simulator
Scilab
www.scilab.org

Create A USB Virtual COM Port

Physical COM ports are disappearing from PCs, but the right firmware can make a USB device appear as a virtual COM port. As Jan explains, you can use almost any microcontroller with an embedded or external full- or high-speed USB controller to design and program your own USB virtual COM port device.

COM ports have long provided a convenient way for PCs and embedded systems to exchange information. The traditional COM port on a PC is an RS-232 serial port on a motherboard or expansion card. Recent PCs often skip RS-232 in favor of USB. But with the right firmware, a USB device can appear as a virtual COM port that applications can access using .NET's `SerialPort` class or other COM port APIs or libraries.

In this article, I will describe how to design and program a USB virtual COM port device with a general-purpose microcontroller with an embedded or external USB controller. The device uses standard USB class drivers included with Windows and other operating systems.

OPTIONS FOR DEVICES

The COM port software interface provides a way for PCs to exchange data with devices for any purpose. A classic example is a modem that enables a PC to send and receive data over phone lines and responds to AT (Hayes modem) commands from the PC. Other COM port devices support vendor-specific command sets for data acquisition, motor control, or other uses.

To communicate with a COM port, an application first reserves the resource by opening the port. For many devices, an application can also get and set port parameters such as the bit rate, the number of data bits per word, and the flow-control method. The application reads and writes to the port as needed. When finished communicating, it closes the port to allow other applications access to it.

A USB virtual COM port is a software interface that allows applications to access a USB device as if it were a

USB CRASH COURSE

Every USB communication is between a host and a device. The host manages traffic on the bus, and the device responds to communications from the host. USB 2.0 supports three bus speeds: 480 Mbps (high speed), 12 Mbps (full speed), and 1.5 Mbps (low speed).

A device endpoint is a buffer that stores received data or data to transmit. Each endpoint has a number, a direction, and a wMaxPacketSize value, which is the maximum number of data bytes the endpoint can send or receive in a transaction.

Each USB transfer consists of one or more transactions that can carry data to or from an endpoint. A transaction begins with a token packet that specifies an endpoint number and direction. An IN token packet requests a data packet from the endpoint. An OUT token packet precedes a data packet from the host. In addition to data, each data packet contains error-checking bits and a packet ID (PID) with a data-sequencing value. Many transactions also have a handshake packet where the receiver of the data reports success or failure of the transaction.

USB supports four transfer types: control, bulk, interrupt, and isochronous. In a control transfer, the host sends a defined request to the device. The host uses control transfers after detecting an attached device to request a series of data structures called descriptors from the device. The descriptors provide information about the device's capabilities and help the host decide what driver to assign to the device. A class specification or vendor can also define requests.

Control transfers have up to three stages: Setup, Data (optional), and Status. The Setup stage contains the request. The Data stage contains data from the host or device, depending on the request. The Status stage contains information about the success of the transfer.

The other transfer types don't have defined stages. Instead, higher-level software defines how to use the data. Bulk transfers have the highest throughput on an otherwise idle bus but have no guaranteed timing. Printers and USB virtual COM port data use bulk transfers. Interrupt transfers have guaranteed maximum latency, or time between transaction attempts. Mice, keyboards, and virtual COM port status notifications use interrupt transfers. Isochronous transfers have guaranteed timing but no error correcting. Streaming audio and video use isochronous transfers.

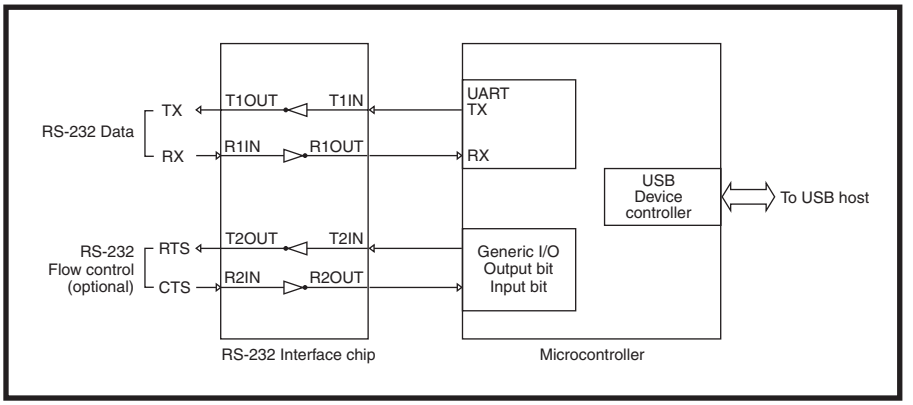


Figure 1—A microcontroller with a UART and an embedded USB device controller can function as a USB/RS-232 bridge. The RS-232 port in this example includes two data lines and two lines for flow control.

built-in serial port. Many USB virtual COM port devices function as bridges that convert between USB and RS-232 or other asynchronous serial interfaces. But a virtual COM port doesn't have to have a serial interface at all. Some virtual COM port devices convert between USB and a parallel interface. Or a device might just read and store sensor data from an on-chip analog port and send the data to a PC via USB.

One way to create a virtual COM port device is to use a dedicated chip, such as a Future Technology Devices International FT232R USB UART. The chip handles all USB-specific communications in hardware and has an asynchronous serial port that can interface to a port on a microcontroller. FTDI provides drivers for Windows and other operating systems. A similar chip is FTDI's FT245R USB FIFO, which has a parallel interface instead

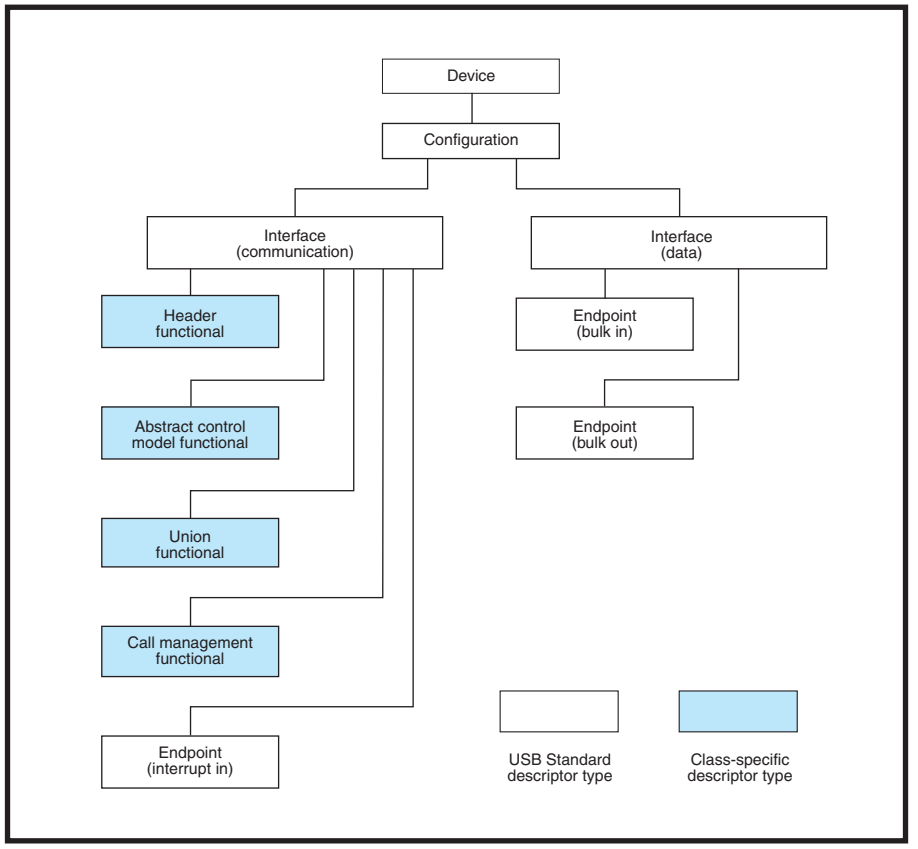


Figure 2—A USB virtual COM port device can use the USB CDC class drivers provided by Windows and other operating systems. A device that exchanges vendor-defined data can use these descriptors.

TOTAL PHASE

Beagle USB 480 Protocol Analyzer

$1200



High-speed USB monitor with real-time capture and filtering

Aardvark I2C/SPI Host Adapter

$250



I2C, SPI, & GPIO interface to USB emulates master and slave devices

Beagle I2C/SPI/MDIO Protocol Analyzer

$300



Non-intrusive I2C, SPI, and MDIO bus monitor with 20 ns resolution

All products include: Windows and Linux support, free software, free API with language bindings for C, C++, .NET, Python, VB6, and more.

Special Offer for Circuit Cellar Readers

Visit the link below for a special offer on Total Phase tools

www.totalphase.com/offer/CC174393

www.totalphase.com
(408) 850-6500 sales@totalphase.com

of a serial port.

These chips are a quick way to add a USB port to a design. Many existing devices with asynchronous serial ports can use a USB UART to convert to USB with no firmware changes. Other companies with UART bridge chips include MosChip Semiconductor, Prolific Technology, and Silicon Laboratories.

The device I'll describe takes a different approach. The design doesn't require a specific vendor's USB controller or driver. The device can use a general-purpose microcontroller with an embedded USB controller or a CPU that interfaces to an external USB controller. The USB port can be full- or high-speed. Device firmware manages USB communications and whatever other tasks the device is responsible for. Instead of a vendor-specific driver, the PC uses the USB communication devices class (CDC) driver included with Windows and other operating systems. For Windows, an INF file matches the driver to the device.

Several microcontroller vendors provide example firmware for USB virtual COM ports. The chips include Atmel's AT89C5131, Microchip Technology's PIC18F4550, and NXP Semiconductors's LPX214*x*. These examples are good starting points for projects.

If you don't have CDC example code for your CPU, you can base your firmware on other example code that transfers data using bulk or interrupt transfers. Any complete example firmware includes code for returning descriptors and responding to other control transfers and events on the bus. At the device, bulk and interrupt transfers are identical. The only difference is how the host schedules the transfers.

Listing 1—The device descriptor names the communication devices class (CDC).

```
// Device descriptor

rom USB_DEV_DSC device_dsc=
{
    0x12,               // Descriptor size in bytes
    0x01,               // DEVICE descriptor type
    0x0200,             // USB version, BCD (2.0)
    0x02                // Class: CDC
    0x00,               // Subclass: none
    0x00,               // Protocol: none
    0x08,               // Max. packet size, Endpoint 0
    0x0925,             // USB Vendor ID
    0x9060,             // USB Product ID
    0x0100,             // Device release, BCD (1.0)
    0x00,               // Manufacturer string index
    0x00,               // Product string index
    0x01,               // Serial number string index
    0x01                // Number of configurations
};
```



DUTIES OF FIRMWARE

Refer to the "USB Crash Course" sidebar for a quick review of USB basics. USB CDC firmware for a generic COM port device performs several tasks. During enumeration, the firmware responds to requests for descriptors that identify the device's CDC function.

The device receives COM port data following OUT token packets addressed to the bulk OUT endpoint and sends COM port data or NAK in response to IN token packets addressed to the bulk IN endpoint. To send status information, the device returns notification data in response to IN token packets on the interrupt IN endpoint. A device with no information to send returns a NAK. Most devices also respond to class-specific control requests that set and get serial-port parameters.

Figure 1 shows a CDC device that functions as a USB-to-RS-232 bridge. The microcontroller's asynchronous serial port interfaces to a Maxim Integrated

Conference: September 14-19 | Exhibition: September 16-17
Santa Clara Marriott | Santa Clara, CA
pcbwest.com

- Two-day exhibition
- Free events and sessions
- Networking opportunities
- Over 35 conference courses

REGISTER ONLINE FOR YOUR FREE EXHIBITS PASS!

GROW YOUR KNOWLEDGE . . .



IN THE HEART OF
SILICON VALLEY

PCB WEST 2008
Conference & Exhibition

UP Media Group

Monitoring Applications Just Got Easier



Smarter Embedded Design

The RCM4300 RabbitCore® module gives you more processing power and memory for your demanding embedded applications that require web interfaces, graphics or encryption. Data logging applications will also take advantage of the miniSD™ memory cards with up to 1GB of storage. It's easy to add the RCM4300 – or any RabbitCore module – to your pin-compatible design.



From **$80** qty. 1000

Superior Performance and Mass Storage for:

- Web interfaces and graphics
- Data logging
- Security and encryption

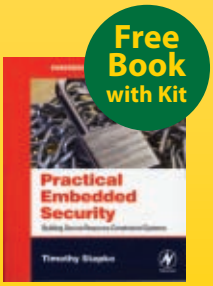
RCM4300 RabbitCore®

- **Rabbit® 4000 @ 58.98 MHz**
- **10/100Base-T Ethernet**
- **Up to 1GB of miniSD™**
- **8 channel analog input, 12-bit**

Get Your Development Kit Today for $259

Development kits include everything to start your development, Dynamic C® software, RCM4300 RabbitCore module, 512MB miniSD™ card, development board, programming cables and accessories.

www.rabbitkits.com



RABBIT™

2900 Spafford Street, Davis, CA 95618 Tel **530.757.8400**

08081

Listing 2—The communication interface provides an interrupt endpoint for sending notifications to the USB host.

```
// Communication interface descriptor

0x09,               // Descriptor size in bytes
0x04,               // INTERFACE descriptor type
0x00,               // Interface number
0x00,               // Alternate setting number
0x01,               // Number of endpoints
0x02,               // Class: CDC communication
0x02,               // Subclass: abstract control model
0x02,               // Protocol: V.25ter (AT commands)
0x00,               // Interface string index
```

Listing 3—The data interface provides two endpoints for sending and receiving COM port data.

```
// Data interface descriptor

0x09,               // Descriptor size in bytes
0x04,               // INTERFACE descriptor type
0x01,               // Interface number
0x00,               // Alternate setting number
0x02,               // Number of endpoints
0x0a,               // Class: CDC data
0x00,               // Subclass: none
0x00,               // Protocol: none
0x00,               // Interface string index
```

Products MAX232 or similar RS-232 converter. Microcontrollers generally don't have dedicated port bits for RS-232's status and control signals, but a device can use any spare port bits for needed signals. Typical RS-232 signals used for flow control are RTS and CTS.

The bridge performs the function of an RS-232 port on the host PC. The TX and RTS signals are outputs, and RX and CTS are inputs. In RS-232 lingo, the port is configured as a DTE.

The RS-232 interface on a PC can connect to a serial port on a microcontroller, to another component with a serial interface, or via a null-modem cable, to another PC. For cables of up to 4,000', use a full-duplex RS-422 interface chip such as a Maxim MAX3087. For a serial network, use an RS-485 transceiver in place of the MAX232. If unneeded for modem control, the RS-232 signals DTR, DSR, RI, and CD can serve as general-purpose I/O bits or remain unused.

CUSTOMIZING THE DESCRIPTORS

The USB 2.0 specification defines the content and format of standard USB descriptors. The CDC specification defines additional class-specific descriptors.

Figure 2 shows the descriptors in a typical CDC device that functions as a generic virtual COM port. I'll focus on device-specific items you may need to change from values in example firmware.

Every USB device has a device descriptor (see Listing 1). The class code specifies CDC as the device's class. The vendor ID and product ID values identify the specific device. Every device with the same vendor ID/product ID pair should use the same driver on the host.

The serial-number string index identifies a descriptor that contains a serial number. A serial number prevents unwanted "COM port proliferation." A device with a serial number retains its COM port number if it is moved to a different USB port on a Windows PC. A device that doesn't contain a serial number gets a new port number on each attachment to a different port on a PC.

Besides a device descriptor, a typical CDC virtual COM port device has one configuration descriptor and two interface descriptors. Each interface descriptor has subordinate descriptors. The configuration descriptor specifies power requirements and the number of interfaces in the configuration. The interface descriptors tell the host how the device implements its communication functions.

The communication interface descriptor names a CDC subclass and protocol (see Listing 2). Generic COM port devices and some modems belong to the abstract control model subclass. The protocol is V.25ter, which documents common AT commands. For compatibility with standard host drivers, a generic virtual COM port device should specify the V.25ter protocol even if the device doesn't use AT commands.

The communication interface has four class-specific descriptors and an endpoint descriptor. The header functional descriptor names the version of the CDC specification the interface complies with. The abstract control model descriptor specifies what class-specific requests and notifications the device supports. (More on those later.) The union functional descriptor identifies the interfaces that belong to the CDC function, which are typically the communication interface plus a data interface. The call management functional descriptor tells how the device manages calls. Because a generic COM port device has no calls to handle, the descriptor says the device doesn't handle call management.

An interrupt endpoint sends status notifications to the host. The endpoint descriptor provides the endpoint's number, direction, and wMaxPacketSize.

The data interface is responsible for sending and receiving the COM port data. The interface descriptor tells the host that the interface has two bulk endpoints, one to carry data to the host and one to carry data from the host (Listing 3). Each endpoint has an endpoint descriptor.

After retrieving the descriptors from the device and assigning the CDC driver, the host polls the bulk IN endpoint for COM port data and polls the interrupt IN endpoint for notification data. An endpoint with no data to send returns NAK in response to received IN token packets.

When sending COM port data to the host, a device indicates the end of a transfer by sending a short packet,

which is a data packet containing less than wMaxPacketSize bytes. If the entire transfer is less than wMaxPacketSize, the transfer's only data packet is a short packet. If the transfer consists of more than wMaxPacketSize bytes, only the transfer's final data packet is a short packet.

When a transfer has an exact multiple of wMaxPacketSize bytes, the endpoint returns wMaxPacketSize bytes in one or more transactions until all of the data has been sent. The endpoint then indicates the end of the transfer by responding to an IN token packet with a zero-length packet (ZLP), which is a data packet with no data bytes.

On a Windows host, every CDC virtual COM port device must have an INF file that contains the Vendor ID and Product ID values and names the software driver for the device. Windows doesn't provide a generic INF file for USB virtual COM port devices as it does for other device types, such as mass storage and human interface devices (HIDs).

You can modify an INF file provided with CDC example code. Listing 4 is the part of an INF file that specifies a Vendor ID and Product ID for a device.

Listing 4—The INF file for a CDC device must include the device's USB vendor ID (VID) and product ID (PID). In this listing, VID = 0925h and PID = 9060h.

```
[Manufacturer]
%MFGNAME%=Lakeview

[Lakeview]
%DESCRIPTION%=DriverInstall, USB\VID_0925&PID_9060
```

SETTING PORT PARAMETERS

Besides COM port data, devices with asynchronous serial ports often exchange information relating to port parameters, status and control signals, and error states. The host uses class-specific requests and notifications to send and receive the information. Devices that don't have asynchronous serial ports don't need to support these requests and notifications.

The SET_LINE_CODING and GET_LINE_CODING requests set and request the bit rate, number of stop bits, parity, and number of data bits. Photo 1 shows host and device data for these requests. The second byte in the setup transaction is the request number.

When an application changes a port's parameters, the host issues a SET_LINE_CODING request (20h) to the device. The OUT transaction contains the line-coding data. The values 80 25 in the first two bytes indicate a requested bit rate of 2580h, or 9,600 bps. The value 08 in the final bytes is the number of data bits per transmitted word on the serial port. After receiving the parameters, the device implements any requested changes. The IN transaction of the transfer is the status stage. The device indicates success by returning a ZLP (no data).

Request 21h is GET_LINE_CODING. The device sends its current parameters in the IN transaction. The OUT transaction is the status stage, where the host indicates success by sending a ZLP. In the SET_CONTROL_LINE_STATE request (22h), the host tells the device how to set the RS-232 control signals RTS and DTR. The host sends the control-line states in the third byte of the setup transaction. Bit 0 is the state of DTR, and bit 1 is the state of RTS. Device firmware detects the request, accepts the data, and implements any changes to the bits. The IN transaction is the status stage. The device indicates success by returning a ZLP.

The SEND_BREAK request (23h) requests the device to send an RS-232 break signal (a positive RS-232 voltage on the TX line) for a specified number of milliseconds. If the requested value is FFFFh, the device should maintain the break signal until receiving another SEND_BREAK request with a value of 0000h. In .NET's `SerialPort` class, setting the BreakState property causes the host to send this request with a value of FFFFh or 0000h.

The SERIAL_STATE notification provides a way for a device to send the states of the RS-232 status signals RI, DSR, and CD, the break state, and error states for buffer overrun, parity error, and framing error. The notification consists of an 8-byte header and two notification bytes. The interrupt IN endpoint returns a notification or NAK in response to received IN token packets.

The notification doesn't include the state of RS-232's CTS status signal. Device firmware can still read CTS on a local asynchronous port and take action as needed. For example, if a virtual COM port device has data to send to a remote device that hasn't asserted CTS, the virtual COM port device can store the data in a buffer and wait to transmit.

If the buffer is full, the virtual COM port device can NAK attempts by the USB host to send more data. When the remote device asserts CTS, the virtual COM port device can send the buffered data and accept new data from the host. To use CTS in this way, the USB host doesn't need to know the state of CTS. If you want to use CTS in an unconventional way, such as having a host application read a switch state on a device, you're out of luck with the CDC driver unless you can define a vendor-specific command that travels on the same bulk pipes that carry application data.

Two CDC requests (SEND_ENCAPSULATED_COMMAND and GET_ENCAPSULATED_RESPONSE) and one notification (RESPONSE_AVAILABLE) relate to AT commands. The CDC

Item	Device	Endpoint	Status	Payload
Class request IN (0x21)	4	0	OK	7 bytes (80 25 00 00 00 00 08)
SETUP transaction	4	0	ACK	8 bytes (A1 21 00 00 00 00 07 00)
IN transaction	4	0	ACK	7 bytes (80 25 00 00 00 00 08)
OUT transaction	4	0	ACK	No data
Class request OUT (0x22)	4	0	OK	No data
SETUP transaction	4	0	ACK	8 bytes (21 22 00 00 00 00 00 00)
IN transaction	4	0	ACK	No data
Class request OUT (0x20)	4	0	OK	7 bytes (80 25 00 00 00 00 08)
SETUP transaction	4	0	ACK	8 bytes (21 20 00 00 00 00 07 00)
OUT transaction	4	0	ACK	7 bytes (80 25 00 00 00 00 08)
IN transaction	4	0	ACK	No data

Photo 1—The USB communication devices class defines requests that set and get port parameters and control-line states. (Ellisys USB Explorer USB analyzer display)

specification requires abstract control model devices to support the requests and notification. A typical generic COM port device doesn't connect to a modem that supports AT commands. For these devices, the host will never send the requests or require the notification data, so device firmware doesn't need to implement them.

MAXIMIZING PERFORMANCE

Following a few guidelines can improve the performance of device firmware. For full-speed devices, set wMaxPacketSize in the bulk endpoint descriptors to 64 to enable the transfer of the most data possible in each USB transaction. With a UHCI host controller, if a full-speed bulk endpoint's wMaxPacketSize is less than 64, the host controller schedules no more than one transaction per millisecond for the endpoint. (Full-speed host controllers comply with either the OHCI or the UHCI standard. Many PC motherboards contain UHCI controllers.) High-speed bulk endpoints must set wMaxPacketSize = 512. To transfer large amounts of data to the host as quickly as possible, use wMaxPacketSize data packets. Larger packets mean fewer transactions are needed to transfer the data.

When sending data to the host in multiple transactions, avoid returning NAK. Immediately after sending a packet of data, refill the endpoint buffer and arm the endpoint for the next transaction. For the fastest response, configure the endpoint to trigger an interrupt after sending data.

When receiving data from the host, avoid returning NAK. Immediately after receiving a packet of data, retrieve the data from the endpoint buffer and arm the endpoint for the next transaction. For the fastest response, configure the endpoint to trigger an interrupt after receiving data.

On the host, be aware that setting RS-232 control lines or changing the parity type or other parameters can be slow compared to performing the same operations on an internal serial port. To perform these actions on a USB virtual COM port device, the host must send a request in a control transfer.

Jan Axelson is the author of USB Complete *(Lakeview Research, Third Edition, 2005),* Serial Port Complete *(Lakeview Research, Second Edition 2007), and many other books and articles about hardware interfacing. You can reach her at jan@Lvr.com.*

RESOURCES

Lakeview Research, "Links to CDC Firmware," www.lvr.com/serport.htm.

USB Implementers Forum, "USB CDC Specification," www.usb.org/developers/devclass_docs#approved.

SOURCE

FT232R USB UART and FT245R USB FIFO
Future Technology Devices International
www.ftdichip.com







Icy Hot

The Soft-Core Concept, FPGAs, And You

Question: Does a "soft-core" make sense for your current application? Tom covers the history of the soft-core technology and introduces the ARM Cortex-M1 soft-core and the FPGA chips it runs on. As you'll see, this solution from Actel and ARM makes the soft-core concept more viable than ever for new applications.

Question: You see an engineer staring at the screen and scratching their head. How can you tell if they're designing hardware or software? Answer: Check if they're wearing shoes.

Yes, using programmable logic to implement hardware that runs a program seems a bit circular, like being in a room full of mirrors. Nevertheless, I'll admit I'm a softie for the soft-core-CPU-in-an-FPGA concept.

Soft-core roots go back nearly 20 years, a long time in terms of silicon. Back then, FPGAs were like hardware Superglue, sticking all the little TTL chips cluttering circuit boards together in one part. Naturally, the FPGA design tools reflected the hardware-centric stance and, back in the day before HDLs and synthesis, still showed vestiges of TTL-era roots with schematic capture, ABEL/PALASM, state machines, and the like.

Sure FPGAs were cool. Compared to cobbling together a sea of TTL or developing a custom chip, designing with FPGAs was faster, easier, less risky, and less expensive. But FPGAs were still about doing stuff the old way, just doing it better. Notably, hardware (including FPGAs) was still hardware and software was still software and never the twain shall meet.

One of the first soft-core CPUs I recall was KCPSM, a skunk works project by Ken Chapman of Xilinx. As the name implies, his tiny 8-bit unit was barely more than a "programmable state machine" but nevertheless an actual processor in an FPGA. Today, the legacy of KCPSM lives on, now officially blessed by Xilinx as PicoBlaze (K. Chapman, "Solar Panel Monitor," *Circuit Cellar* 185, 2005).

KCPSM was nifty, but it was so tiny and specialized it simply reinforced the impression that a soft-core could never compete with a "real" CPU. That premise changed one day in the early 1990s when I found myself in the lair of Silicon Valley FPGA guru Philip Friedin. There I saw his home-brewed R16 16-bit soft-core CPU running in a Xilinx FPGA (see Photo 1). Beyond the cool blinking LED demo, R16 was a real CPU that did real work as a dynamically loaded test processor in one of Friedin's rocket science projects.

The message in these early soft-core designs was profound, but also way ahead of its time. Through the 1990s, FPGAs stolidly stuck to their traditional "ASIC replacement" strategy targeting hardware designers. Soft-core CPUs remained little more than eccentric curiosities for a cult of true believers.

Flash forward to 2000 and my next soft-core love affair with Jan Gray's XSOC ("Building a RISC System in an FPGA," *Circuit Cellar* 116, 117, and 118). Gray's hand-crafted design was quite elegant and delivered excellent performance and efficiency. Gray also

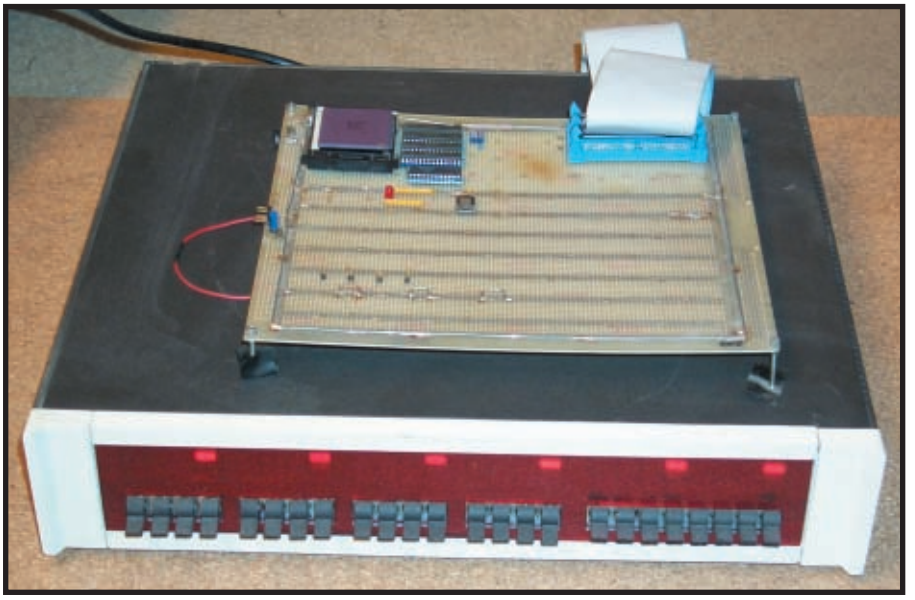


Photo 1—It was almost 20 years ago that Philip Friedin crafted his R16 16-bit processor running in a Xilinx FPGA.

ported a "C" compiler, laying to rest the historic "no-tools" objection. Though Gray's own interests have diversified, his web site is still online and remains a useful archive for soft-core information and advice (www.fpgacpu.org).

In the years since, the soft-core concept has gone mainstream with FPGA heavyweights Xilinx (MicroBlaze) and Altera (NIOS) fully on board. Now Actel and ARM are pairing up as a duet to sing their own version of the soft-core tune.

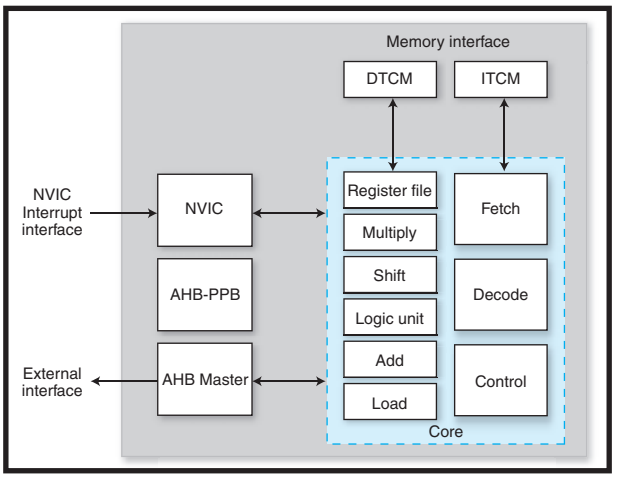


Figure 1—As delivered with Actel FPGAs, the Cortex-M1 includes a 32-bit, three-stage pipelined core as well as a standard bus interface (AHB-Lite) and an interrupt controller (NVIC). A Harvard dual-instruction/data bus architecture with tightly coupled memories boosts performance.

YEAH, BUT...

Wonderful as the concept may be, it's time for a little reality checking. A soft-core starts with a fundamental handicap, namely the fact that "FPGA gates" cost more, use more power, and are slower than "real gates."

Of course, the design-in decision isn't as simple as that. For instance, Friedin's original inspiration for R16 was to provide a self-test mechanism for an otherwise "hardware" FPGA-based subsystem. In this scenario, the conventional processor metrics (i.e., price/performance/power) didn't matter much and indeed adding an external processor would have only complicated the task.

It is no surprise that Moore's Law has had a huge impact. FPGA gates will never match real ones, but as the differences shrink at some point, they become less than material for a particular application. This month, I'm looking at a solution from Actel and ARM that makes the soft-core concept more viable than ever for new applications, maybe even your own. Actel's contribution comprises their intriguing flash memory FPGA line-up, including ProASIC (speed and density), Fusion (analog and bulk flash), and this month's chip de jour, Igloo (low power).

Moore's Law has done a good job when it comes to making chips, including FPGAs, faster and cheaper. But until recently, the power side of the equation has just gotten worse with the march of leaky silicon. For now, there's a brute force way out, namely cutting the supply voltage, in the case of Igloo to 1.2 V. Igloo also includes specific power-saving embellishments such as a "Flash Freeze" mode that preserves the internal state and keeps the I/O pins alive, while allowing most of the chip to sleep. Of course, there's no free lunch and in return for the power savings, Igloo sacrifices some speed compared to its ProASIC and Fusion cousins.

Yes, FPGA price, performance, and now power have been headed in the right direction, but that's not to say there isn't room for improvement. Particularly as it applies to soft-core processors, remember that no programmer ever met a megahertz or megabyte they didn't love (at least until a new gigahertz or gigabyte caught their fancy).

Relying on the silicon wizards to make things better is a time-honored approach. But what goes on the silicon also has an impact and, unlike fab technology, is something mere mortal designers can influence right now with their choices.

THUMBS UP

Enter the ARM Cortex-M1 core, designed to squeeze the most out of the FPGA silicon (see Figure 1). Like other FPGA soft-cores (e.g., NIOS and MicroBlaze), the Cortex-M1 is "free,"



CONFERENCE
OCTOBER 26-30, 2008

EXPO
OCTOBER 28-29, 2008

Hynes Convention Center
Boston, MA

Embedded Systems CONFERENCE BOSTON

LEARN TODAY. DESIGN TOMORROW.



The **Embedded Systems Conference Boston** was created specifically to meet the information needs of the system architects and design engineers who create these complex systems. The technical program at ESC is the Fall's largest industry conference offering in-depth and unbiased content, that delivers real world solutions from leading technology experts.

Tracks include:

- Commercial & Open Source Operating Systems
- Debugging, Verification & Test
- Design Team Management
- DSP & Multimedia: Algorithms & Implementation
- Hardware Design, including Programmable Logic
- Multi-core Processors & Programming
- Real-time Development
- Security
- Software Development
- Virtualization
- Wired & Wireless Networking

EARLY BIRD REGISTRATION NOW OPEN!

www.cmp-egevents.com/web/escb/registration

Signature Sponsor

Windows Embedded

Platinum Sponsor

AMD Embedded Solutions

techinsights EVENTS

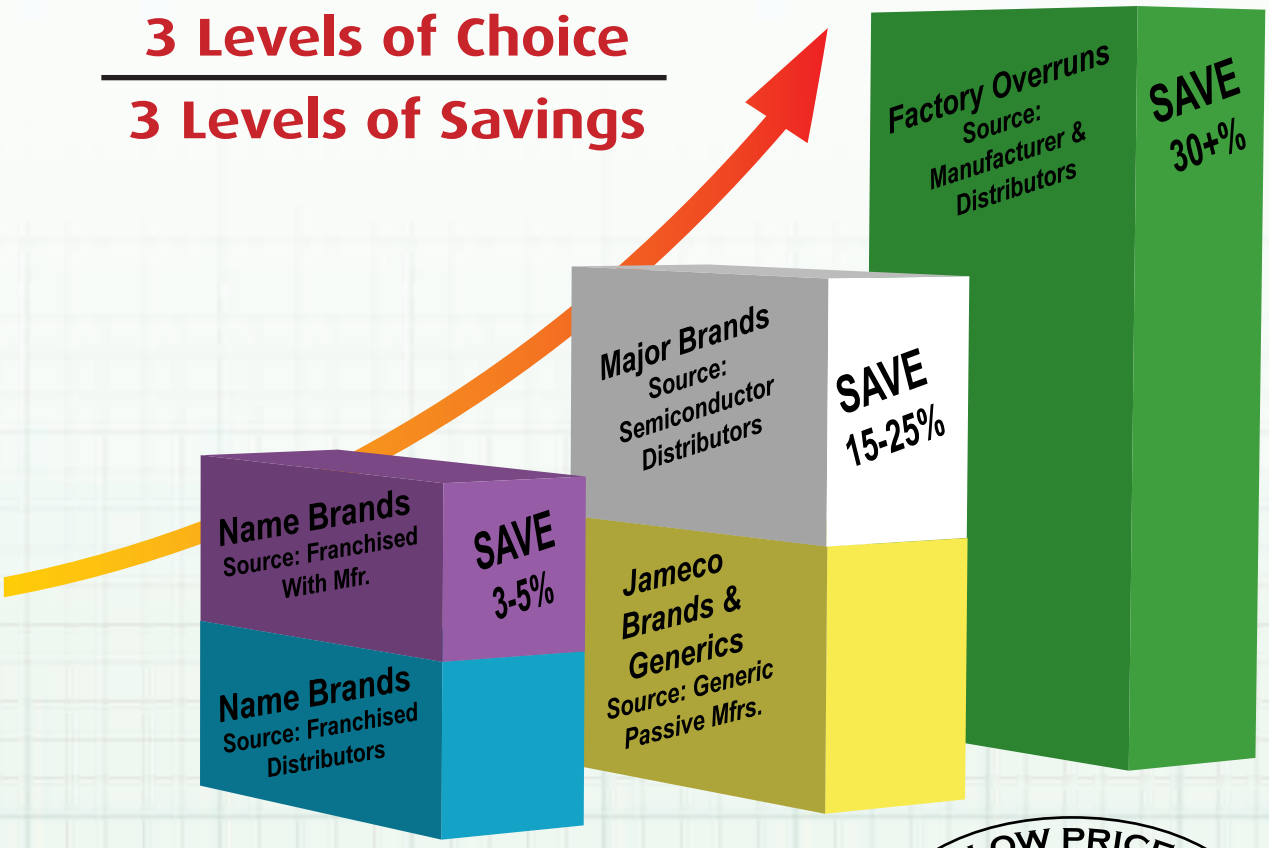
Exclusively at Jameco... The Lowest Prices... Guaranteed!

Save 3% to 5%
Looking to save on thousands of Name Brand components? Jameco offers their popular products for 3% to 5% below market price, and we guarantee it!

Save 15%-25%
Save more when a specific manufacturer is not required. Major Brand semi's are sourced from 5-6 major manufacturers, plus we offer Jameco brand and generic passives for even greater savings.

Save 30% or more!
Looking for killer deals? Jameco buyers often find Factory Overruns from some of the industry's biggest names. That means savings of 30% and more to you!

3 Levels of Choice
3 Levels of Savings



Other Jameco Advantages:

- We offer over 300 Major Manufacturers
- ***99%*** of catalog products ship the same day.
- Lowest prices guaranteed, or we pay 10%.

LOW PRICE GUARANTEE

If you find a lower print-advertised price for a product we offer, just let us know. We'll offer you that product for 10% less than the price you found, right down to our cost!

JAMECO® ELECTRONICS

Order 24 hours a day, 7 days a week

www.Jameco.com

Or call800-831-4242 anytime

SEMICONDUCTORS • PASSIVES • INTERCONNECTS • ELECTROMECHANICAL • POWER

portabledesign conference & exhibition

Power Management for a Wireless World

Conference sessions, Analyst Presentations and Panel Discussions on designing and powering portable, low-power wireless consumer devices.



- Complimentary Registration
- Keynotes
- Panel Discussions
- Technical Seminars
- Exhibition
- Lunch
- Networking Reception

September 18, 2008

San Jose, California

Wyndham Hotel

Attend for free. Register now.

www.portabledesignconference.com

an RTC Group Publication

PORTABLE DESIGN™

the engineer's resource for portable applications

sponsored by

mirasol™ QUALCOMM DISPLAYS

Actel®

expresslogic

tensilica®

	ARM7	Cortex-M1
Device	M7AFS600	M1AFS600
Tiles required	6,083	4,452
Device utilization	44%	31%
Performance	28 MHz	67 MHz

Table 1—FPGAs are finicky when it comes to hosting a soft-core. Seemingly similar architectures can exhibit quite different results in terms of price and performance.

understanding that means the core cost is buried in the price of the silicon. In Actel's case, only specific "M1 Enabled" versions of Igloo, ProASIC, and Fusion chips are allowed to run the core, a restriction enforced by the design tools as well as the usual legalese.

Even the truest of believers has to admit the last place you'll ever find a soft-core is replacing the big-iron "computer" chips, such as the 'x86 under the hood of your PC. Similarly, at the low end it's hard to imagine an FPGA-with-soft-core solution ever displacing a buck-or-less MCU, as much for reasons of system cost as the chip price itself.

My take is that middle-of-the-road applications are where soft-cores make the most sense. Modest performance expectations mean that the soft-core disadvantages don't rise to the level of showstoppers. At the same time, there's enough functionality required to garner meaningful benefit from the soft-core concept.

So why Cortex-M1? After all, Actel already offers an equivalent line-up using a full-fledged ARM7 soft-core. And for that matter, what about ARM's spiffy new Cortex-M3 core already making the rounds of standard MCU suppliers?

A key point is that soft-core efficiency and performance are highly dependent on an implementation that closely matches, and fully exploits, the underlying FPGA hardware. Seemingly innocuous architectural trinkets can sabotage a soft-core design if they go against the grain of the silicon. On the other hand, a soft-core that's tailored to fit an FPGA can deliver surprisingly competitive performance and efficiency.

So, the simple answer to the "Why Cortex-M1?" question is that it delivers a lot of bang for less bucks (i.e., less silicon, and thus less power too). Table 1 compares Actel's ARM7 and Cortex-M1 implementations and you can see the difference is nontrivial. Size-wise, the 'M1 is significantly smaller than the ARM7. In fact, the 32-bit 'M1 is only slightly bigger than an 8-bit 8051 soft-core.[1] Even more notable is the speed difference, with the 'M1 maximum clock rate fully twice that of the ARM7.

You can imagine the 'M1 designers were asking themselves "How can we carry forward the essence of ARM7 and 'M3 yet streamline for a better fit on an FPGA?" The answer they came up with is summed up in one word: "Thumb."

A bit of a history lesson is in order. Back in the RISC "renaissance" of the 1980s (when ARM was still "Acorn RISC Machines"), computer architects latched onto the concept of fixed-length 32-bit instructions. It kind of made sense at the time, I guess, and ARM was not alone in this. But meaning no disrespect, it was a dumb idea.

One truism is that computers execute some instructions far more frequently than others. Wasting a 32-bit op-code to decrement a register, compare to zero, or do a short branch makes about as much sense as using a sledgehammer to hang a picture.

It's more than a matter of just the cost of memory wasted storing fat instructions. The bloat ripples all the way through the architecture. For instance, less code density translates directly to more cache fills and spills. And a 32-bit op-code means 32 signal transitions every instruction cycle just to convey the same information that could be packed into many less. What's that do to power consumption?

Even considering the "memory, megahertz, and megawatts are free'" mentality of the RISC era, the inefficiency proved intolerable for blue-collar embedded applications. Thus, it wasn't long before the architects retrofitted their designs with schemes to improve code density. In ARM's case, they came up with "Thumb," mapping a 16-bit op-code facade onto the 32-bit architecture.

In legacy ARM chips, "Thumb" was a "mode" (i.e., the programmer was responsible for deciding which portion of the application code should use 16- or 32-bit instructions and overtly managing the transition in software). Kind of a hassle, so the new Cortex designs returned to the traditional (i.e., CISC) solution with the variable-length (i.e., 16- and 32-bit, no "modes") "Thumb-2" instruction set.

The Cortex-M1 relies on the 16-bit instructions comprising the intersection of the ARM and Cortex instruction sets plus a few of the Cortex 32-bit instructions, presumably those that are most application critical (e.g., long branch) yet don't hamstring the FPGA implementation. While it shares the shorter op-code philosophy of an 8-/16-bit MCU, the Cortex-M1 is still a real 32-bit computer inside (see Figure 2). One key Cortex advantage carried forward is a Harvard architecture with simultaneous instruction and data access, a step up from the combined instruction/data bus of ARM7. The Cortex-M1 exploits this dual-bus advantage with separate instruction and data "Tightly Coupled Memories" (ITCM and DTCM) for high-speed access, kind of programmer-directed caches. There's also the familiar nested vectored interrupt controller (NVIC) to accelerate real-time interrupt-driven applications.

APPLE VERSUS ORANGE

The 'M1 soft-core also comes with a complete hardware debug engine that has all of the fancy features designers have come to expect, including breakpoints, watchpoints, single-step, register/memory access, and so on.

This is an opportune moment to

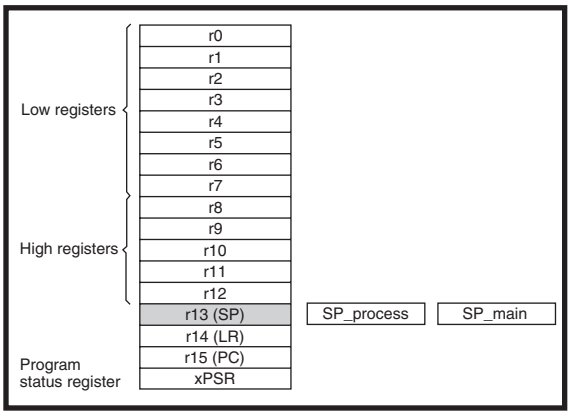


Figure 2—The Cortex-M1 has the code density and silicon footprint of an 8-/16-bit core, but there's a 32-bit engine (ALU, register file, address space) under the hood.

µC/PROBE: A PRETTIER PRINTF

These days every processor comes with a full suite of fancy tools courtesy of commercial (e.g., IAR Systems and Keil) and open-source (e.g., GNU and Eclipse) suppliers. The multi-windowed GUI IDEs have every imaginable feature and then some. But sometimes, as I grapple with their dizzying array of windows and menus, I yearn for a simpler time.

In the old days (absent the modern arsenal of simulators, emulators, and debuggers), designers often used judiciously placed PRINTF statements to get under the hood. In the hands of an expert, PRINTF can be surprisingly effective when it comes to finding and fixing hardware and software bugs.

Thanks to Micrium, PRINTF lives, albeit on steroids and gussied up for the PC screen (see Photo 1). As I see it, µC/Probe has three things going for it.

First, like PRINTF, µC/Probe is simple and relatively nonintrusive. You just add a few lines of code to your embedded application to exchange data with µC/Probe software running on the PC, which in turn references the symbol table output by your software toolchain. Virtually any embedded design is supported with connection options including RS-232, Ethernet, and JTAG.

Second is the fact µC/Probe will work today and down the road with any target processor (i.e., standard chips, FPGA soft-cores, DSPs, and more). That means you can spread the cost of µC/Probe across multiple chips and projects, which is good, because at $995, it isn't a giveaway. Fortunately, as with all the other Micrium software, there is a free 30-day evaluation version you can try before you buy.

The clincher is that µC/Probe is fun thanks to the ability to map a symbol value to an icon or graphic on the screen. Wouldn't you rather see a screen full of lights, dials, thermometers, or pictures than the typical yawn-inspiring blizzard of tiny text, cryptic symbol names, and hard-to-decipher runes like 0x1AC543D7? Who says this embedded biz has to be so drab and boring? Let's live a little!

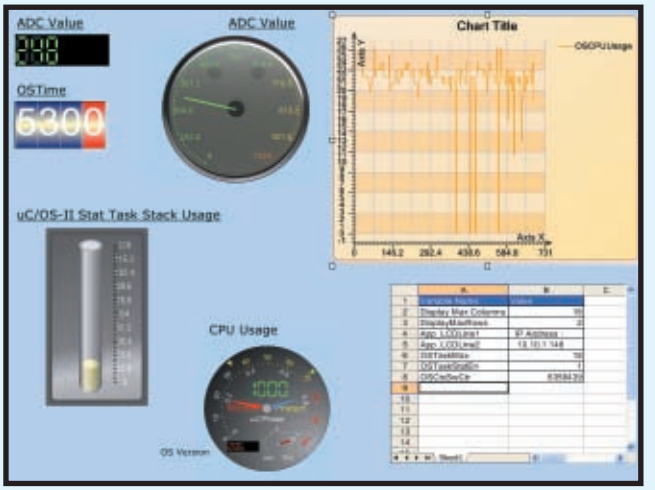


Photo 1—Check out the µC/Probe in action!

make a key point about the soft-core concept and apples and oranges. When evaluating the soft-core versus hard-core trade-off, the perspective tends to be how close (e.g., price, performance, power) the former can get to the latter. There's some insight to be had with that perspective, but also the trap of comparing apples to apples.

Let's stipulate that if there is a nice crisp standard MCU (i.e., one apple) that perfectly matches your application, it will no doubt taste better than a mushier FPGA soft-core-based solution (i.e., the other apple).

It's to the degree that a standard chip isn't a perfect fit that the window opens for soft-core. That's evermore likely the case as "kitchen sink" MCUs incorporate any and every popular I/O and glue function. Yes, it's nice to be able to buy a chip with everything you need and then some, but don't forget that you're paying for the "then some" even if you don't need it.

Which brings me back to the 'M1 debug logic. Once your application is debugged and in production, do you really need to ship a debug engine to each end-user? Unlike a standard MCU, with 'M1, you can leave the fancy debug stuff off the production units, perhaps relying on a simpler scheme if you

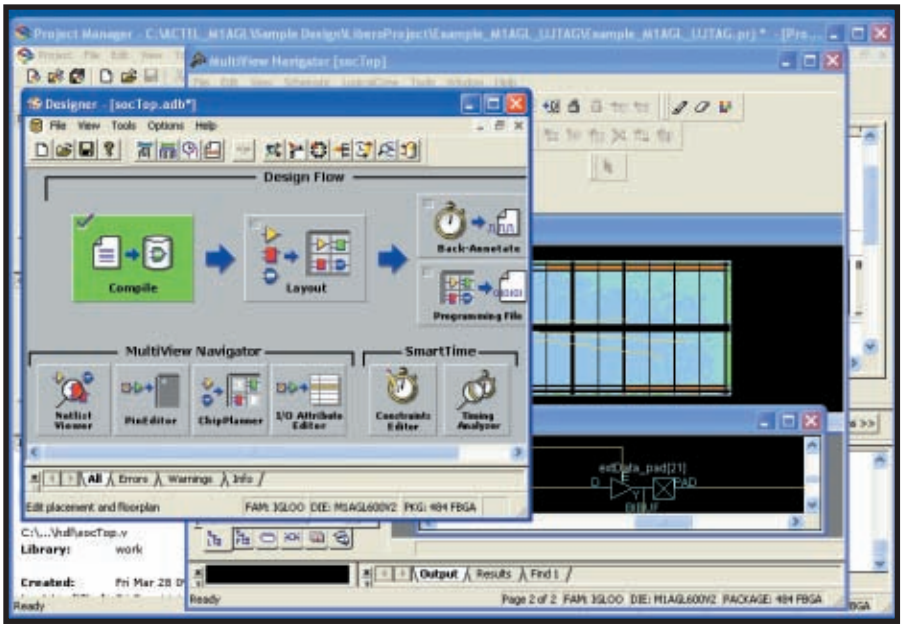


Photo 2—Libero enables you to get deeply under the hood. Like a shade-tree mechanic trying to rebuild an engine, if you don't know what you're doing, it's all too easy to "fix it until it's broke."

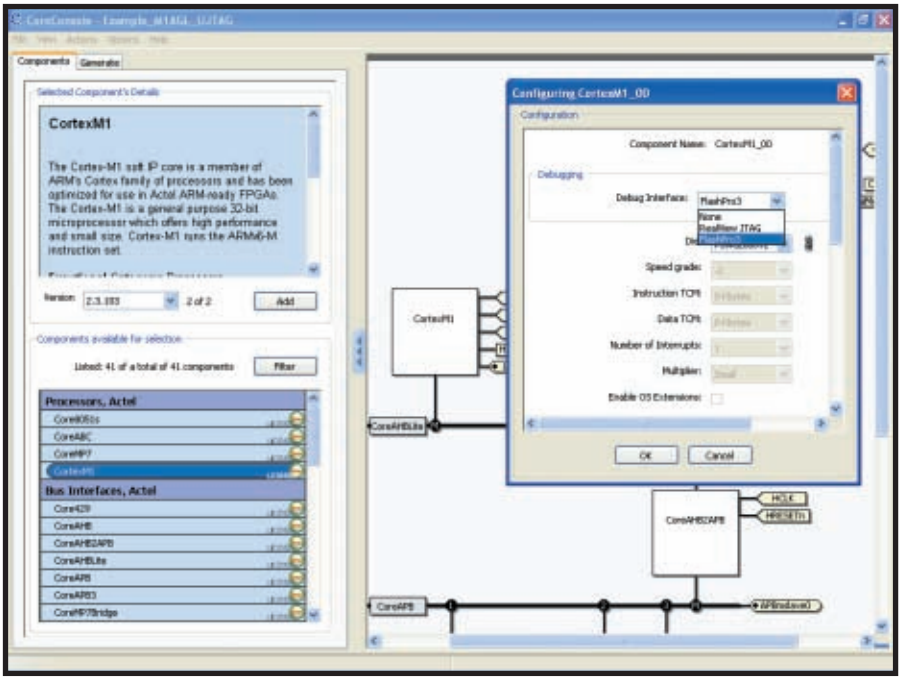


Photo 3—CoreConsole makes it easy to get a design started by cutting and pasting predefined components. Although they weren't yet implemented at the time of my evaluation, notice the planned Cortex-M1 configuration options in the dialog box.

need under-the-hood factory or field access. (Refer to the "µC/Probe: A Prettier PRINTF" sidebar.)

"Big deal," you say? You bet it is because the debug engine is practically as big as the 'M1 core itself! For example, on a mid-size FPGA, stripping out the debug cuts the amount of silicon occupied by the core from one-half of the chip to just one-third.

Needless to say, the soft-core option looks better and better if the alternative standard MCU comes with even more features you don't need. The lesson is that when you compare a soft-core to a standard chip, make sure you compare apples (the standard chips you can choose from) to oranges (a soft-core crafted with exactly, and only, the features you need).

TOOL TIME

Top Ten Predictions from the IEEE Field-Programmable Custom Computing Machines (FCCM) Conference…

1996 #1—"We will hate the tools"
1998 #1—"We will still hate the tools"
2000 #7—"We will merely dislike the tools"
2007 #10—"We hate the tools more"

Source: www.fccm.org/top10.php

I guess you could call that progress? So far, I've been talking about the Cortex-M1 soft-core and the FPGA chips it runs on. In my opinion, the combination is pretty compelling now, and will only be more so in the future.

Nothing fancy at all about Cortex-M1. And that's a good thing. Performance ranges from 24 MHz (Igloo at 1.2 V) to 66 MHz (ProASIC and Fusion), which is more than adequate for many blue-collar applications. At the same time, the 'M1 is lean enough to pass a price, performance, and power consumption reality check. The Thumb-plus (or Thumb-2-minus? Call it Thumb-1.5?) architecture gets a free ride on the ARM bandwagon. On the silicon front, all FPGAs continue their march towards mass markets thanks to Moore's Law, and I especially like what Actel is doing with flash memory FPGAs.

New-and-improved chips with a popular and practical soft-core—what more could a designer ask for? How about some good tools?

Given the historic "ASIC-replacement" strategy, it's no surprise that FPGA design tools are pretty complicated. Actel's Libero design suite is no exception, with its rather daunting mix of third-party (e.g., Synplicity synthesis, Mentor Graphics simulation) and in-house (e.g., floor-planning, power estimation, place and route) tools. No getting around the fact you really have to become something of a chip-design expert to get the most out of Libero (see Photo 2).

But FPGA suppliers are trying to make life a little easier. In Actel's case, they front the process of building an SoC with a tool called CoreConsole (see Photo 3). It's kind of a hardware "preprocessor" that allows you to cut and paste at a block diagram level, choosing from a decent library of pre-fabbed processing and I/O functions. Once you've

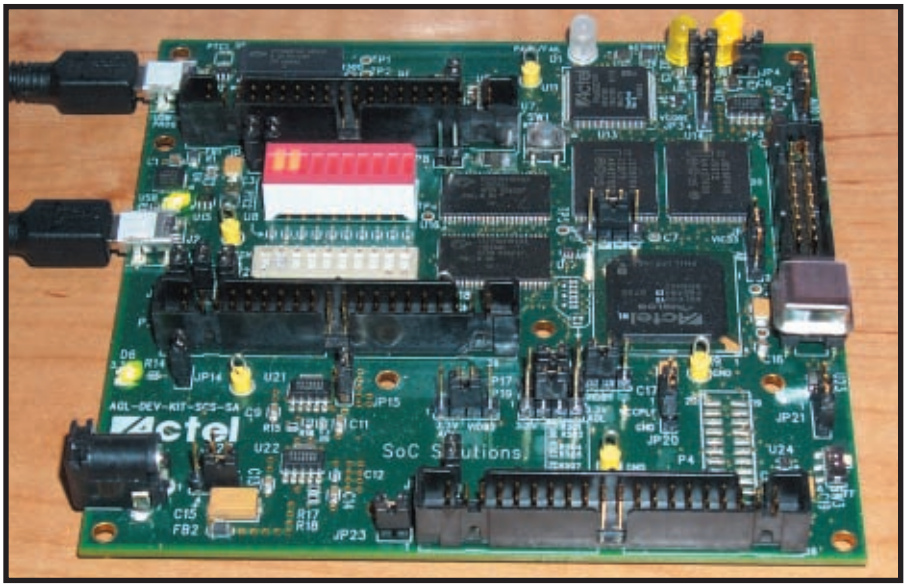


Photo 4—Actel offers 'M1 evaluation kits based on ProASIC, Fusion, and Igloo (shown here). Notice the "stereo" (i.e., two-port) USB lash-up. One port is used for debugging while the other implements a virtual serial port.



Photo 5—Success! The simple traffic light demo gives no hint of the amazing technology at work: a 32-bit pipelined soft-core running a compiled "C" program in a 600,000-gate FPGA all held together by gigabytes of chip design and software tools.

stitched together your design outline, CoreConsole hands the project off to Libero and you take it from there.

Naturally, a soft-core solution needs software tools. Actel offers SoftConsole, a serviceable and familiar solution based on GNU tools and an Eclipse-based IDE and debugger. And there's also plenty of third-party support from notables like Keil (now owned by ARM) and IAR Systems. Given its Thumb roots, the 'M1 should be an easy add-on for any tool provider supporting other ARM cores.

You can give it all a test drive with an Actel 'M1-enabled development kit, such as the Igloo-based setup (M1AGL-DEV-KIT-SCS) I played with (see Photo 4). Note that versions of the kit based on ProASIC and Fusion chips are available as well.

It seems that I'm invariably forced to upgrade my PC every time I install the latest and greatest FPGA toolchain. So, the first thing I looked for when I opened the kit was "System Requirements." Sure enough, the doc says you need XP Professional, and I'm running XP non-professional or whatever they call it. So, once again, everything came to a screeching halt so I could go through the fire drill of upgrading my PC.

Not. I just installed anyway. And, fortunately, everything seemed to work well enough to at least get through the basic demos (see Photo 5). But I suppose there are no guarantees if you don't have XP Pro. Oh yeah, the doc also says "so sorry" when it comes to Vista.

Bottom line: buyer beware (or at least be aware) when it comes to FPGA toolchains and PC system requirements.

Actel has done a lot, arguably as much as possible, to make the "out-of-box" experience fast and easy. You still have to move a lot of bits—some 32 billion of them by my count (i.e., a 4-GB install)—and jump through some licensing hoops. Nevertheless, it wasn't long before I had the 'M1 up and running in the Igloo chip, which was pretty remarkable considering how much was going on under the hood.

The getting started guide reassures that "It is not necessary to synthesize, place-and-route, or generate a programming file" to get the demo working. Not necessary, and not advisable, if you don't know what you're doing. I dabbled a bit with the design in Libero, but it was quickly apparent that I was in way over my head.

THE KNOWLEDGE

"If you want to be a cab driver in London, you first must first earn 'The Knowledge.' Students study for many months to memorize the thousands of little streets in London and learn the best routes from place to place. And they go out every day on scooters to scout around and validate their book learning.

"Similarly, if you want to be a great FPGA-optimized core designer, you have to acquire The (Device) Knowledge. You have to know what the LUTs, registers, slices, CLBs, block RAMs, DLLs, etc. can and can't do. You have to learn exactly how much local, intermediate, and long routing is available per bit height of the logic in your datapath and how wide the input and output buses to the block RAMs are. You have to learn about carry chain tricks, clock inversions, GSR nets, 'bonus' CLB and routing resources, TBUFs, and so forth.

"You also need to know the limitations of the tools. What device features PAR can and can't utilize. How to make PAR obey your placement and timing constraints, and what things it can't handle. And how to 'push on the rope' of your synthesis tools to make them emit what you already know you want.

"The Knowledge isn't in any book, alas. Yes, you can read the 'street maps,' e.g. the datasheets and app notes, but that only goes so far. You have to get out on your 'scooter' and explore…"

—Jan Gray (www.fpgacpu.org/log/aug02.html#theknowledge)

It's safe to say the soft-core concept makes sense for some applications. It's also safe to say that, considering advances in FPGAs (price, performance, and power) and the soft-cores themselves, soft-core makes sense for more applications today than yesterday, and yet more tomorrow.

Ironically, the excellence of the silicon and elegance of the core may not be the deciding design-in factors. Rather it's "The Knowledge" (i.e., FPGA design expertise) you can bring to bear. The difference in the "quality of results" (i.e., silicon area, power, and speed) with or without it is large, surely pivotal for many design-in decisions.

The catch-22 is that you need "The Knowledge" to make a credible decision whether a soft-core makes sense for your application. That means you'll have to acquire The Knowledge in order to determine if you need it!

Before I bite that bullet, I think I'll check into some other ways to skin the FPGA cat. When it comes to programmable logic, I know there are folks who share my "there's got to be a better way" sentiment. Stay tuned.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

REFERENCE

[1] Actel Corp., "The Advantages of the 32-Bit Cortex-M1 Processor in Actel FPGAs," 2007, www.actel.com/documents/CortexM1_Advantages_WP.pdf.

SOURCE

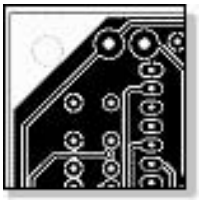
Igloo FPGA and M1AGL-DEV-KIT-SCS Cortex-M1-enabled Igloo starter kit
Actel Corp.
www.actel.com

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital submission sheet and digital files that meet the specifications on the digital submission sheet. **ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2″ x 3″ FORMAT**. Call for current rate and deadline information. E-mail adcopy@circuitcellar.com with your file and digital submission or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For more information call Shannon Barraclough at (860) 875-2199.

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



PCB

EasyPC SW

From TSI: Import Eagle libs and Designs. **Buy Online** www.pcb-sw.com Import any of the big three, using Intelligent Gerber. Download the FREE Demo from the web. **PromCode D0506**

GoFast™ MATH PACKAGE

- Improve floating point speed without increasing hardware cost
- Replaces compiler math functions
- Intrinsic functions 2-3X faster
- Complex functions 5-10X faster
- In assembly language for speed
- Many processors supported
- Conforms to IEEE 754
- Full source code, no royalties

www.smxrtos.com

Micro Digital Inc
RTOS Innovators

800.366.2491 sales@smxrtos.com

USB

Add USB to your next project—it's easier than you might think!

- USB-FIFO up to 8 mbps
- USB-UART up to 3 mbps
- USB/Microcontroller boards pre-programmed with firmware
- **2.4GHz ZigBee™ & 802.15.4**
- **RFID Reader/Writer**

Absolutely NO driver software development required!

www.dlpdesign.com

DLP Design

PRINTED CIRCUIT BOARDS

QUALITY PRODUCT
FAST DELIVERY
COMPETITIVE PRICING

PROTOTYPE through PRODUCTION

- Aluminum Backed PCB
- Single & Double sided
- SMOBC/RoHS
- LPI mask
- Through hole or SMT
- Nickel & Gold Plating
- Routing or scoring
- Electrical Testing
- Artwork or CAD data
- Fast Quotes
- Flex Circuits

SPECIAL OFFER:
10 pcs (3days) 1 or 2 layers $249
10 pcs (5days) 4 layers $695
(up to 30sq. in.ea.) includes tooling, artwork, L.P.I. mask & legend

PULSAR, INC

9901 W. Pacific Ave. Franklin Park, IL 60131 • Phone 847.233.0012
Fax 847.233.0013 • www.pulsar-inc.com • sales@pulsar-inc.com

Bluetooth 1SPP



1SPP Demo Board



PB-1SPP
Peripheral Board

The Bluetooth 1SPP Boards provide a quick and easy method to wirelessly connect two computers via a standard DB9 serial port or a standard USB port.

The host accesses the BGB203 via a physical UART interface that can be configured for a variety of characteristics, including baud rate, word length, parity, stop bits, RTS/CTS flow control, and DTR/DSR flow control.

Both kits may be purchased directly from Digi-Key.

The 1SPP Demo Kit consists of two 1SPP boards and cables. Digi-Key PN 622-1020-ND

The PB-1SPP is Digi-Key PN 622-1021-ND. The PB-1SPP would most likely be used in pairs and must be utilized in conjunction with the USB-Dongle, PN 622-1007-ND.

FDI also provides low cost demo kits for NXP ARM7 LPC2xxx, LPC9xxx and ICP/ISP FLASH programmers.

FDI
www.teamfdi.com
VISA/MC/Amex

Future Designs, Inc.
2702 Triana Blvd
Huntsville, AL 35805
(256) 883-1240
Fax (256) 883-1241

EMULATE A USB
Keyboard + Mouse + Joystick



MODEL KE-USB108

-108 individual programmable switch inputs
-Produce long macros from a single input
-Send keystrokes,mouse or joystick actions
-Ideal handling of toggle switches
-2 Rotary Encoder inputs
-Potentiometer inputs for Joystick X,Y,Z

888-690-9080
HAGSTROMELECTRONICS.COM
11 FIDDLERS GREEN, LANSING NY 14882

iMAGECraftC
The Best Selling C Compiler for AVR

- High quality code generation
- First commercial whole program compression optimizer

high powered development, garage powered prices!

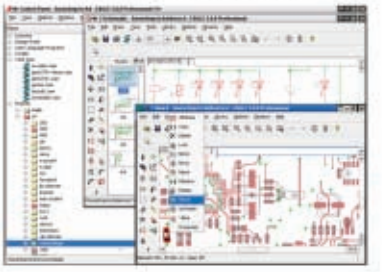
starting at $249!

- Application builder
- Multiple versions
- Global optimizer
- Simple to use IDE
- Multiple platform support
- Flexible licensing schemes

www.imagecraft.com
info@imagecraft.com
(650) 493-9326 • FAX (650) 493-9329

The number one choice in PCB-Design just got better.

EAGLE V5.0 Highlights
Display, Zoom and Grid History + Lock function for parts + unlimited name/value attributes for parts + move group to other sheet w/o modifying the board + reorder sheets with drag&drop or insert new sheets + REPLACE function for the schematic editor + selective display of unrouted tracks + time-saving new re-routing feature + minimum clearance for any combination of net classes + add elements to group or subtract elements from group.



www.cadsoftusa.com

Introducing the iLoad Mini™ - low cost, miniature load cell

Digital Core
Analog Compatibility



Loadstar SENSORS
www.loadstarsensors.com/cc | 650.938.4282

www.can232.com

Only $108 €89

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

The original (Controller Area Network) CAN to RS232 serial converter.

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

MYLYDIA, INC.

Free one time setup!
(subject to approval)

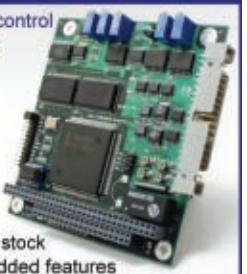
W/5 double side PCB
2 weeks

Tel: 800.695.9342
sales@mylydia.com

www.mylydia.com

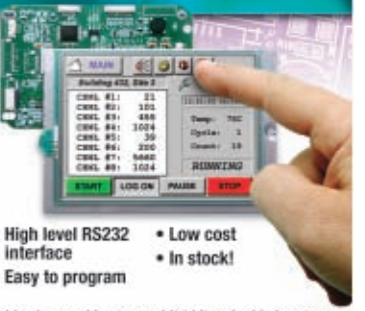
Solve complex signal acquisition problems...

- positioning & control
- environmental
- acceleration
- transients
- pressure
- vibration
- sonar
- GPS
- Linux Driver
- Guaranteed in stock
- Many newly added features
- 16-bit analog inputs and outputs
- Million sample FIFO eliminates interrupts
- Wide analog input and output ranges
- -40°C to +85°C Standard
- Order 24/7, fast and easy.



www.stx104.com
Apex Embedded Systems
help@stx104.com • 608-256-0767 x24

Add a color touch interface to your embedded product!



- High level RS232 interface
- Low cost
- In stock!
- Easy to program

Add color graphics to any 8/16 bit embedded system. Easy, fast and flexible. Up and running in hours!

REACH TECHNOLOGY INC.
www.reachtech.com • 510-770-1417
842 Boggs Avenue • Fremont, CA 94539

tincan TOOLS
Linux in a 40-pin DIP
- 200 Mhz ARM9 CPU Board
- 0.1" centers - easy to prototype
- use your existing prototype tools
- full Linux with 2.6 kernel
- CD includes source and schematic
HAMMER

Perfect for robotics, web servers, remote monitoring & communications
www.TinCanTools.com

Applied Micro Training
Online Training for Microchip PIC® Microcontrollers
8 Week Online Course
Learn at your own pace with confidence!
Uses EasyPIC5 Development Board!
See Webpage for Course Contents, Schedule and what Former Students say!
Course is loaded with Flash Videos and Lots of Hands-on programming exercises.
4 week grace period to finish!

WWW.TOMCHRANE.COM

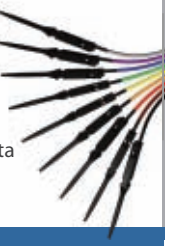


MCS - 51
EXCELLENT noise immunity!

V-E54 v1.0 (USD 59)
Embedded board MPC82E54A, the most efficient MCU from Megawin, 16K Flash speedx12. 8ch. 10bit A/D, 2ch. 12bit D/A
PRO-MISP (USD 99)
Megawin 8051 ISP Programmer, High Speed download via USB Port, 89E58AE, 82E54AE2, developer board included, 3 Pins to target board (+5V, Data, GND)

www.ispmicro.com
sales@ispmicro.com

iMAGECraft C
C Compilers for:
MSP430
CPU12
ARM
AVR
etc.
high powered development, garage powered prices!
starting at $249!
• First commercial whole program compression optimizer
• Multiple versions
• Global optimizer
• Simple to use IDE
• Multiple platform support
www.imagecraft.com
info@imagecraft.com
(650) 493-9326 • FAX (650) 493-9329

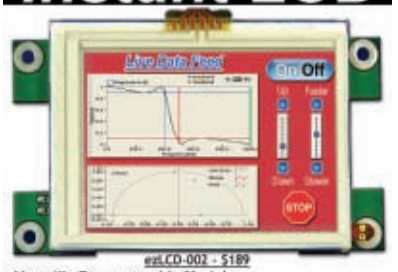

Work Smart. Get Logic.
Introducing the Logic Analyzer, Remastered.

• 8-input, 24MHz, USB
• SPI, I2C, Serial Decode
• All-Aluminum
• Mouse-over timing data
• Snap-to-edge markers
• $149
Visit saleae.com for details and video.
Also find Logic on sparkfun.com or google "logic analyzer."
Saleae

I²C™/SMBus
• Bus Monitors
• Protocol Analyzers
• Host Adapters
• Multiplexers
• Battery Applications
• Software Tools
MCC
Micro Computer Control
I²C is a trademark of Philips Corporation
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

Instant LCD



ezLCD-002 - $189

- Versatile Programmable Module
- Serial/I²C/USB/Parallel Interface
- AVR/BASIC Stamp/VB Compatible
- Onboard Flash Bitmap Memory
- Downloadable TTF Fonts
- 2.7" or 5.6" TOUCH Color TFT LCD
- 240x160 or 320x240 resolutions
- Transflective w/ LED Frontlight (2.7")
- Transmissive w/ 350 nit backlight (5.6")
- 512 colors or 65,535 colors

EARTHLCD.com We Make LCDs Work.™

PHYTEC

phyCORE® SBCs Accelerate 32-bit Designs



phyCORE-LPC3250

CDTS Single Board Computers:
- shorten time-to-market
- reduce development costs
- forgo substantial design issues and risks
- Windows® Embedded CE and Linux Board Support Packages (processor-dependent)
- $170/unit benchmark price at 1K for ARM9-based SBC

ARM: LMX31 (ARM11), LMX27 (ARM9), LPC3250 (ARM9), LPC3180 (ARM9); LPC2294 (ARM7)
XScale: PXA320, PXA270, PXA255
PowerPC: MPC5554, MPC5200B, MPC565, MPC555
ColdFire: MCF5485 Blackfin: ADSP-BF537

- Rapid Development Kits start at $399
- include SBC, Carrier Board, software, docu as well as kit-specific cables, adapters and LCD
- SBC module easily ports from Carrier Board to user target hardware
- Carrier Board serves as target reference design



www.phytec.com ■ 800.278.9913 ■ www.phycore.com

C Compilers supporting PIC® MCUs
PIC10 · PIC12 · PIC14 · PIC16 PIC18 · PIC24 · dsPIC

SAVE $10 use discount code COM20*

Command-line starting at $50
IDE w/Adv. Debugger starting at $350

Compiler Features include:
- 1, 8, 16 and 32 int types & 32-bit floating point
- RS232, USB, and LCD Libraries

dsPIC/PIC24 additional types:
- 48 and 64-bit float
- int48 & int64

IDE with Advanced Debugger:
- Multiple Compilation Units
- C Aware Editor
- RTOS
- New Project Wizard
- C Metrics Calculator...

*Express 30, Aug. 08. Not valid with any other offer. Shipping not included.

PIC® and dsPIC® are registered trademarks of Microchip Technology Inc. in the USA and in other countries.

CCS

www.ccsinfo.com/REV2
262.522.6500 x35
sales@ccsinfo.com

NAND FLASH FILE SYSTEM

- Wear Leveling, Garbage Collection
- Power-Fail Safe
- ECC and Bad Block Handling
- File System with Std C Lib API
- FAT File Structure
- Page Caching to Maximize Performance and Minimize RAM
- 20KB Code Size
- RAM Required < 1% Flash Size

www.smxrtos.com/ffscc.htm

Micro Digital Inc
RTOS Innovators

800.366.2491 sales@smxinfo.com

Overcoming obstacles with...

Mach Design Services, LLC

- Hardware Design
- Software Design
- PCB Design
- PCB Assembly
- & Much More!

Focusing on 8/16 bit control modules for the following apps:
- automotive
- industrial
- consumer

www.machdesignservices.com

Coming Soon!

A Social Networking Site for You, the Electronics Enthusiast!

We Need Beta Testers Now Sign Up At:

www.solderbynumbers.com

NEW

HI-TECH C® PRO for the Microchip

PIC32 MCU Family

C Compiler

FREEWARE

New PRO compiler also operates in Lite mode with no time or size limits:

microchip.htsoft.com/portal/CC_20

HI-TECH SOFTWARE

1,600+ Products
www.DesignNotes.com



Test Equipment
USB PC 2 CH
60MHZ Oscilloscope
$329.99 + free gift



PCSU1000

DesignNotes.com
Affordable Prototyping Products

1-800-957-6867

iMAGEcraft C

NEW: the eBox- AVR!

Compact Complete AVR Kit
- 16K ImageCraft ICCAVR compiler
- Mega16 kit with LED, keypad, LCD and more
- 10 example software projects
- Battery INCLUDED



Perfect for learning embedded programming and the Atmel AVR.

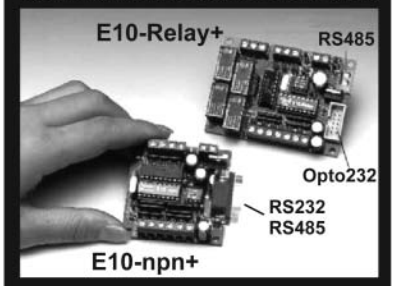


only $199!

www.imagecraft.com
info@imagecraft.com
(650) 493-9326 • FAX (650) 493-9329

The $69 PLC

Work as Stand-Alone Ladder Logic PLC.
Or as Smart Remote I/Os of PC/ PLCs.
RS485 allows 256 units to be networked.



Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!

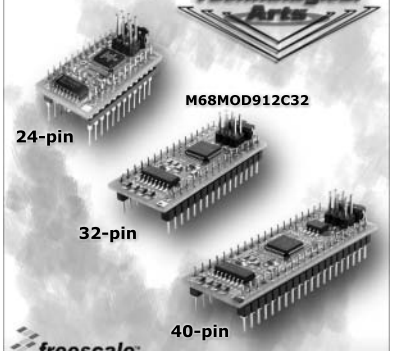
Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm

Triangle Research

Tel: 1-877-874-7527 - PLC specialist since 1993

Freescale 9S12C microcontroller DIP modules

Technological Arts



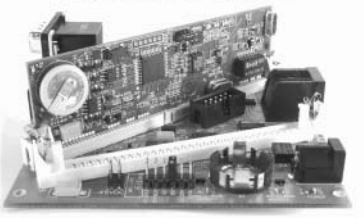
freescale Alliance Member

www.nanocore12.com

Toll-free: 1-877-963-8996

TStik!™

Rugged TINI Java™ Module with 10/100 BaseT



TStik is a ruggedized TINI400 chipset in the familiar SIMM72 form factor. Upgrade most DSTINI1 (TINI390) systems or use our new TILT socket boards (TILT Pro is shown above).
TStik is about $100, and sockets start at under $60.

SYSTRONIX®

full details at www.TStik.com



Weather Instruments for PCs

AAG electrónica

www.aagelectronica.com

Embedded Ethernet

ONLY $98



- 10Base-T Ethernet
- 186 Processor @ 40 MHz
- DOS w/ Flash File System
- 16 Digital I/O • 5V DC
- Console / Debug Port
- Hardware Clock/Calendar
- Socket for DiskOnChip
- 512K DRAM & 512K Flash
- (2) Serial Ports
- (2) 16-bit Timers
- Watchdog Timer
- 3.75" x 2.50"

NEW! picoFlash

$129 Development System

Development Kit Includes:
- picoFlash Controller
- Borland C/C++ 4.52
- TCP/IP, PPP & Web Server
- Serial Driver Library
- AC Adapter and Cables

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

JK microsystems

USB Device Stack

- Compatible with Windows USB 2.0 Host Class Drivers:
  - Mass Storage
  - Mouse
  - Ethernet over USB
  - Serial
- Composite Devices
- ARM, ColdFire, Philips ISP 1161, 1181, 1362, 158x, 1761
- Low Cost, Royalty-Free
- Full Source Code
- Standalone or RTOS
- Host and OTG Available

Micro Digital Inc
RTOS Innovators

www.smxrtos.com/usb



Tiny Linux controller

16MB fast SDRAM	10/100 Ethernet
64MHz Coldfire MCU	3 serial ports (RS232)
4.5 MB flash memory	1 CAN port
SD card socket (to 2GB)	LCD/KPD port
Eclipse/CDT dev. env.	Free serial debugger

55 I/O pins & capabilities to burn...

www.steroidmicros.com

USB Data Acquisition

ADU208 -USB Relay I/O Interface



FEATURES
-8, 5-AMP relay outputs
-8, ISOLATED digital inputs
-Port Powered, Aux 5VDC output $189.00 QTY 1

Other Models:
ADU200- 4 Channel Version with RS232 $139.00
ADR218- Solid-State Version 8 Channel $225.00
ADU100- 3 CH, 16-Bit ISOLATED Analog Inputs, PGA, 4 digital I/O, RS232 and 5 AMP Relay Output $199.00

ONTRAK CONTROL SYSTEMS INC.
PH: (705) 671-2652 Fax: (705) 671-6127

www.ontrak.net

Incredibly Low-Cost USB Chips at HexWax

- **USB-232** USB to serial UART, smarter and cheaper than the FT232R
- **USB-SPI** USB to synchronous serial (I2C, too!)
- **USB-DAQ** Data logger, plugs in like a flash drive
- **USB-Config** Configuration & diagnostic module
- **USB-TakeOff** Power take-off & battery charger
- **WattLogic** Domestic power consumption logger
- **AnniLogic** Anniversary reminder
- **expandIO-USB** I/O expanders
- **TEAleaf-USB** Security dongle

No drivers needed!

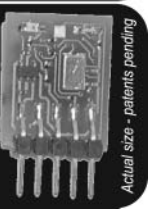
Visit www.hexwax.com today!

Amazing PIC programmer

Most devices supported
ICSP, SQTP, & copy limits

$32 at Digikey & Mouser

www.flexipanel.com



Full Speed
It writes your USB Code!
NEW! HIDmaker FS for Full Speed FLASH PIC18F4550

Creates complete PC and Peripheral programs that talk to each other over USB. Ready to compile and run!

- Large data Reports
- 64,000 bytes/sec per Interface
- Easily creates devices with multiple Interfaces, even multiple Identities!
- Automatically does MULTITASKING
- Makes standard or special USB HID devices

NEW! "Developers Guide for USB HID Peripherals" shows you how to make devices for special requirements.

Both PC and Peripheral programs understand your data items (even odd sized ones), and give you convenient variables to handle them.

PIC18F Compilers: PICBASIC Pro, MPASM, C18, Hi-Tech C.

PIC16C Compilers: PICBASIC Pro, MPASM, Hi-Tech C, CCS C.

PC Compilers: Delphi, C++ Builder, Visual Basic 6.

HIDmaker FS Combo: Only $599.95

DOWNLOAD the HIDmaker FS Test Drive today!
www.TraceSystemsInc.com
301-262-0300

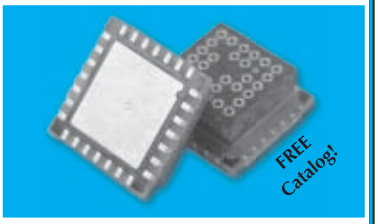
Trace SYSTEMS, Inc.

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

CONNECT TO SMT PADS



- BGA
- QFN (MLF)
- QFP
- SOIC
- PLCC

IC SMT Pads Become Connection Interface with SMT Emulation

Industry's widest selection for SMT Pattern Interconnect Technology allows Test, Prototype, Package Conversion, Emulation

Quick Custom Designs for all package types

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

Temperature Monitoring Over the Internet

-View temperatures using a web browser

-Connect up to four temperature sensors

-Send email alerts or alarms

-Two built-in relays



CONTROL by WEB
www.ControlByWeb.com
(435) 750-5999

SpectraPLUS 5.0
Audio Spectrum Analysis

Features
Sound Card based I/O
FFT sizes to 1048576pts, 1/96 Octave
Up to 24 bit, 200kHz sampling rates
3-D Surface and Spectrogram
Digital Filtering, Signal Generation
THD, IMD, SNR, Transfer Functions
DDE, Macros, Data Logging,
Vibration Analysis, Acoustic Tools

FREE 30 day trial!
www.spectraplus.com

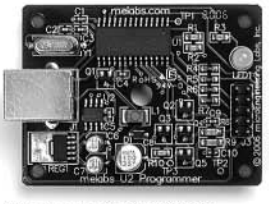
PHS
Pioneer Hill Software
360 697-3472 voice
pioneer@telebyte.com

Order online at:
www.melabs.com

microEngineering Labs, Inc.
Development Tools for PIC® Microcontrollers

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

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, 2K, & XP



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



Serial LCDs
2-line x 16 $39.95
4-line x 20 $49.95
Quantity Discounts Available!

LAB-X Experimenter Boards



Pre-Assembled Boards Available for 8, 14, 18, 28, and 40-pin PIC® MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram
Pricing from $79.95 to $349.95

PICPROTO™ Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic
Pricing from $8.95 to $19.95

BASIC Compilers for PICmicro®



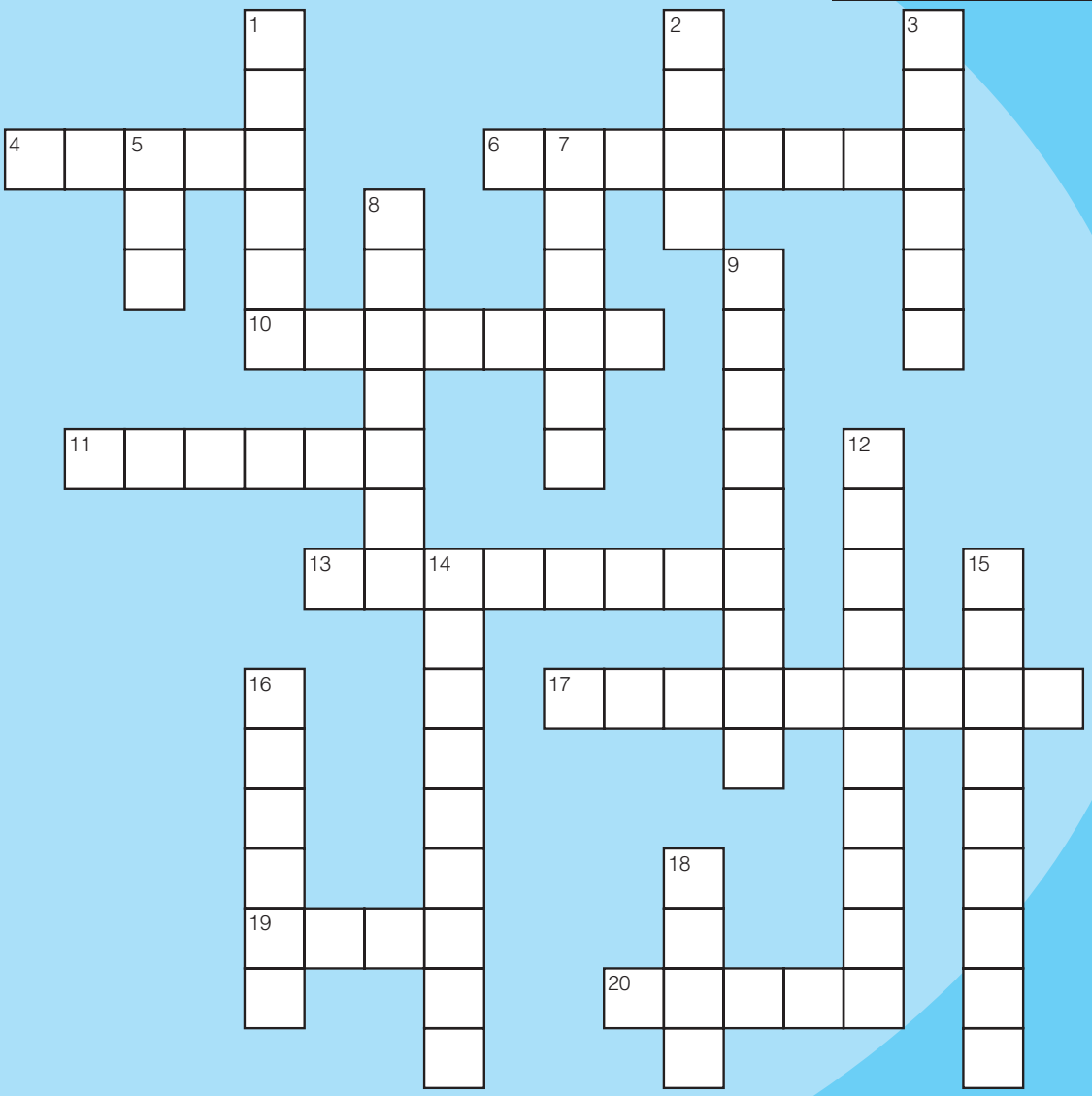
Easy-To-Use BASIC Commands
Windows 9x/Me/2K/XP Interface

PICBASIC™ Compiler $99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO™ Compiler $249.95
32-bit signed variables and math operations
Supports Microchip PIC10, PIC12, PIC14, PIC16, PIC17, and PIC18 microcontrollers
Direct Access to Internal Registers
Supports In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging

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

CROSSWORD



Across

4. Unit of magnetic flux
6. A list of a blogger's favorite blogs
10. A "station" for a laptop
11. Gas + Electric
13. Rotating part of a magnetic circuit
17. An integrated circuit with two or more processors for improved performance
19. Prevents overcurrent
20. Magnifier

Down

1. A chain of writings about a single subject in a newsgroup or forum
2. 10^9
3. Bolis, fireball
5. Text message: Back at keyboard
7. To start an app
8. A tool that converts data into code
9. THz
12. A touchscreen coated with a substance that conducts continuous electrical current across the sensor
14. A cellular tower that provides phone service to a small area
15. A circuit board with predrilled holes in the form of a grid
16. To convert something like a text file into web content
18. Electronic memory that isn't erased when power is removed

The answers are available at www.circuitcellar.com/crossword.

INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at www.circuitcellar.com under the current issue.

Page		Page		Page		Page	
91	AAG Electronica, LLC	91	FlexiPanel Ltd.	90	Mach Design Services, LLC	88	Reach Technology, Inc.
29	AP Circuits	77	Grid Connect, Inc.	87, 90	Micro Digital, Inc.	39	Renesas Technology
92	All Electronics Corp.	88	Hagstrom Electronics	91	Micro Digital, Inc.	1, 95	Saelig Co.
88	Apex Embedded Systems	90	HI-TECH Software LLC	5, 33	Microchip	89	Saleae LLC
89	Applied Micro Training	19	HobbyLab LLC	92	microEngineering Labs, Inc.	90	Schmartboard
7	Atmel	55	Holtek Semiconductor, Inc.	57	Mouser Electronics	48	Sealevel Systems
49	Bitscope Designs	89	ISP Micro	88	Mylydia, Inc.	11	SEGGER Microcontroller Systems LLC
65	CWAV	89, 90	IMAGEcraft	C2	NetBurner	91	Systronix
58, 88	CadSoft Computer, Inc.	19, 88	IMAGEcraft	37	Nurve Networks LLC	C3	Tech Tools
31	Calao Systems	91	Intec Automation, Inc.	91	Ontrak Control Systems	2, 3	Technologic Systems
79	Comfile Technology, Inc.	92	Ironwood Electronics	73	PCB West Design Conf.	91	Technological Arts
90	Custom Computer Services, Inc.	64, 66	JKmicrosystems, Inc.	72	PCB-Pool	87	Technology Sales, Inc.
87	DLP Design	91	JKmicrosystems, Inc.	C4	Parallax, Inc.	89	Tern, Inc.
90	Designnotes	81	Jameco	90	Phytec America LLC	24	Tibbo Technology, Inc.
77	EMAC, Inc.	37	Jeffrey Kerr LLC	92	Pioneer Hill Software	89	Tin Can Tools, LLC
80	ESC Boston	17	Keil Software	13	Pololu Corp.	71	Total Phase, Inc.
90	Earth Computer Technologies	37	Lakeview Research	82	Portable Design Conf. & Exhib.	92	Trace Systems, Inc.
19	Esensors	88	Lawicel AB	87	Pulsar, Inc.	91	Triangle Research Int'l, Inc.
20	ExpressPCB	29	Linx Technologies	63	R4	64	WCSC (Willies Computer Software Co.)
31	ezPCB	88	Loadstar Sensors, Inc.	21, 74	Rabbit, A Digi International Brand	47	Wiznet
87	FDI-Future Designs, Inc.	89	MCC (Micro Computer Control)	89	Rabbit, A Digi International Brand	92	Xytronix Research & Design, Inc.

Preview of September Issue 218
Theme: Data Acquisition

Content Acquisition And Display: Build An Internet-Connected News Ticker

Universal Machine Control: How Proper Hardware Emulation Leads To Prototype Development

Dynamic Animatronic Remote (Part 1): Build A Custom Controller With A Color LCD

Fast Digital Filtering

Prototyping With QFP Parts

Real-Time Vibration Measurement For Aircraft

LESSONS FROM THE TRENCHES Living And Working Off The Grid (Part 2): Installation And Performance

FROM THE BENCH Fingerprint Data Acquisition: Print Sensing With Embedded Technology

SILICON UPDATE Embedder Than Ever: A Look At New Embedded Technologies

ATTENTION ADVERTISERS

October Issue 219 Deadlines

Space Close: Aug. 11
Material Close: Aug. 19

Theme:
Signal Processing

BONUS DISTRIBUTIONS:
ARM DevCon; Embedded Systems Conference East

Call Shannon Barraclough now to reserve your space!
860.875.2199
e-mail: shannon@circuitcellar.com

Saelig

UNIQUE PRODUCTS & SUPPORT

www.saelig.com

Testgear

Color LCD Scope



Owon - 25MHz 2-ch + trigger standalone USB bench scope with 7.8" color LCD. Battery powered optional.

PDS5022S (25MHz) **$325**
PDS6062T (60MHz) **$599**

100/200/300MHz DSO



100/200/300MHz 2 ch, 2000 wfm/s refresh rate. FreeCase!

DS1102C (400MS/s) **$999**
DS1102A (2GS/s) **$1295**
DS1202A (2GS/s) **$1595**
DS1302A (2GS/s) **$1995**

Scope + Analyzer



100MHz 2-ch /16 logic scope and logic analyzer. 2000 wfm/s refresh rate. See waveforms and logic data simultaneously. Trigger on data or signal edge. FreeCase!

DS1102CD **$1499**

15/20MHz Wfm Gen



15/20MHz USB 14-bit Function/ Arbitrary wfm Generators. 10 built-in wfms; FM, AM, PM, FSK, SWEEP and BURST; 6 digit 200MHz frequency counter.

DG1011 / DG1021 **$650 / $795**

1/2GHz RF Generators



High accuracy/stability, wide range, low phase noise/leakage, serial control. Keypad/rotary setting. TGR1040: 10MHz-1GHz, FM int/ext TGR2050: 150kHz-2GHz AM/FM/PM

$2498/4558

EMC Spectrum Analyzer



Handheld Palm PC-based 2.7GHz Spectrum Analyzer. Multiple sweep settings – store wfms, set-ups, etc. Use WiFi PC for email, reports, calculations.

PSA2701T **$1990**

Bargain MSO



25MHz 100 MS/s 2/16ch scope/ logic-analyzer 5ns-5s/div with complex triggering. See wfm + data lines. 7.8" TFT Color LCD. USB-connected. 4MS buffer.

MSO5022 SALE! $799 ~~$899~~

Low-Cost Scope



2-ch 40/100/200MS/s 8-bit scope range with 5/10/25MHz bandwidth and USB2.0 for fast screen updates. Inc. scope/FFT/logging software, case, probes.

PS2203/4/5 **from $297**

50MHz USB Scope



PS3206 - Is an impressive 2-ch 200MS/s 8-bit PC scope adapter - 10GS/s for repetitive signals! No power supply needed, simple USB 2.0 connection

PS3206 Bundle **$1574**

Logic Analyzer



ZeroPlus - Intuitive full-featured 16-ch 4MB 200MHz sampling memory. Sophistcated triggering. I2C, SPI, and UART monitoring. Includes clipleads, USB cable and software.

LAP-16128U **$299**

World's Fastest



PS5204 - 2-ch 250MHz Scope with 128 MS/s and real-time sampling of 1GS/s. Adv triggering. 250MHz spec. analyzer and 125MS/s waveform gen.

PS5203 32MS buffer **$2237**
PS5204 128MS buffer **$3360**

60/100/120MHz AWG



60/100/120MHz USB 14-bit ARB w. USB RS-232, LAN/GPIB. 10 built-in wfms; 6 digit 200MHz frequency counter; opt. 16-ch gen.

DG3061A **$1895**
DG3101A/DG3121A **$2795/$3995**

6 in 1 Scope



6-in-1 USB scope adapter! 200kHz 2-ch 10-bit scope, 2-ch spectrum analyzer, 2-ch chart recorder, 16-ch 8MHz logic analyzer, 5-ch signal gen, 8-ch pattern gen. SPI/I2C/1-w/UART decoding.

PoScope with probes **$199**

Automotive Testing



Auto Diagnostics - Award-winning kits turn yours into vehicle-electrics diagnostic tool. **PS3423 kit $2293** 3-ch logger for 16-bit current/voltage/ temperature measurement.

KLARI-MOD MC **$1900**

Mixed -Signal PC Oscilloscope



100 MHz Scope and Logic Analyzer lets you do complex triggering to find hard-to-get-at glitches, spikes, etc. Huge 4 or 8 MS buffer for deep data drilling and zooming. Optional built-in swept signal generator. 2 Analog channels x 10, 12, 14 bit with more than 60 dB dynamic range. 8 digital inputs for mixed signal display/trigger.

CS328A-4 (4MS Buffer) **$1259**
CS328A-8 (8MS Buffer) **$1474**
CS700A (signal generator) **$299**

Handheld Scope



20MHz or 60MHz rugged hand-held USB scope with 3.8" color LCD. Built-in meter - great for your tool kit.

HDS1022MN (20MHz) **$593**
HDS2062M (60MHz) **$699**

Pen Scope



PS2105 - 25MHz USB powered scope-in-a-probe! Up to 100MS/s, 24kS buffer C/VB/ Delphi/LabView/VEE drivers.

PS2105 (25MHz, 100MS/s) **$369**
PS2104 (10MHz, 50MS/s) **$234**

RF Testing



Economical, portable RF test enclosures, for initial compliance testing. **from $1295**
Shielding Tents with external frame, instant up/down for emissions-free room. **from $3500**

Electronic DC Load



For testing dc power/battery sources Const. current, resistance, conductance, voltage and power modes 0 to 80 V and 0 to 80 A 300W Transient generator. Variable drop-out.

LD300 **$1532**

DMM



6 1/2 digit 50KS/s, 2MS mem. Opt. 16-ch mux. Automeasure DC/AC, 2/4-wire R, C, Continuity, Diode, Frequency, Period, Ratio, Temp, Sensor, High/Low Limit Math.

DM3061 **$795**

USB Bus Analyzers



USB1.1/2.0/WiMedia analyzers and now - **EX260** can test USB3.0 5Gbps too! Emulate host/device, verify compliance.

TR110 / EX200 / EX260
$999 / $2999 / CALL

SPI Bus Analyzer



SPI Bus Analyzer - SPI protocol exerciser and analyzer for standard SPI and non-standard 4-wire and 3-wire serial protocol interfaces up to 50 Mbps. PC control via free GUI 8PI Control Panel Software.

SPI Xpress **$699**

GPS Logger



TrackStick II - Pocket-sized logger tracks position data for vehicles, personnel, etc. Records date, time, location, route and speed. Shows data on Google maps! **$175**

Super TrackStick II **$239**

Misc

Serial-Ethernet Cable



Network your serial product easily without a PC using this 28" cable. Provides an instant IP address. RS-232 port compatible with most 10/100mpbs Ethernet hubs.

eCOV-110-P **$89**

Instant Ethernet



10/100 Ethernet controller on a chip! Fully hardwired TCP/IP core. No OS required. TCP offload improves system performance. PCB too!

W5100 (integrated PHY) **$5.88**
WIZ110SR **$35**

Ethernet-IO



Ether-IO - UDP/IP-controlled 24 digital I/O board 3 x 8-bit TTL ports each independently programmable. Connects to any TCP/IP Ethernet network.

Ether-IO 24 **$119**

Multiparameter Loggers



Mini-logger with built-in temp/humidity/ pressure/3-axis accel sensors. Customize msmt/record parameters; send data to PC after or during logging. Synchronize multiple loggers, merge data to single record

MSR145S **from $468**

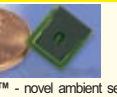
Serial Server



RS232/422/485 device server for network-enabling legacy serial devices. 1/2/4/8/16/32 ports 10/100Mbps-230kbps. Static IP/DHCP/PPPoE.

Portbase-3010/RS232 **from $149**

Temp/RH Sensors



UPSICAP™ - novel ambient sensors and modules accurately measure temp/RH. Amazing range/response: 0% to 100% RH, -90 to +85degC; 0.25s! Interchangeable without recalibration. **from $3 / $40**

SD Storage



Tiny module for adding up to 2GB SD data storage to microcontrollers. Simple serial data transfer. 3.6-6V @ 25mA. Status LEDs

uDrive-uSD-G1 **$39**

Engineering Software



AudioWave: LF signal generator **$60**
LochMaster: strip-board design **$80**
RealView: realtime curveplotter **$80**
Profilab: control/log DMM or I/O bd **$80**

Easy OLED Display



4D Systems - Compact, economical smart OLED with graphics for 'stand-alone' functionality to your project, drive from USB or RS232.

uOLED-96-G1 **$89**
uOLED-128-GMD1 **$99**

Tiny USB Serial



World's smallest USB adapters. Ready-made CP2101 miniboard USB interface to 1 Mbps.

microUSB **$25**
HR-USBUART **$29**

USB Loggers



EL-USB-1 - Standalone USB temp data logger (-25°C to +80°C). **$49**
EL-USB-2 (+ humidity) **$79**
EL-USB-3 (voltage) **$69**
EL-USB-4 (current loop) **$69**
EL-USB-CO (Carbon Monoxide) **$89**

FREE COFFEE



Call 1-888-7SAELIG
Mention Offer# SBW
to get a free Starbucks Card with your $50 order or more!
While supplies last - not available with any other offers

USB to I2C



Devasys - Provides a simple "drop-in" solution for connecting your PC to 90kHz I2C + 20 I/O lines. Free software. Use multiple boards for more I2C/IO.

USB I2C/IO **$89**

I2C for PCs



PCI93LV: industry-standard I2C card for PCs. **WINI2C/PCI** software gives windows-interface to develop and debug I2C bus systems. **UCA93LV** is USB version. Transparently monitor at 400kHz!

$499

RF Modules



RF TX/RX Modules - simultaneously transmit composite video and stereo audio signals. 2.4 GHz ISM band. 4-ch RF, 5V operation

AWM630TX **$16**
AWM634RX **$27**

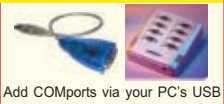
Wireless Modules



Easily create mesh networks at 910MHz - like Zigbee only better! Complete RF solution for fast & reliable data transmission.

B915FHtiny/40-SMD-WA **$73**
B915FHtiny/80-SMD-WA **$73**

1/2/4/8/16 x RS232



Add COMports via your PC's USB Port. **USB-COM-S $20**

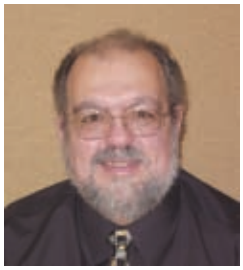
USB-2COM **$48**
USB-4COM **$105**
USB-8COM **$195**
USB-16COM-RM (rackmount) **$459**

Saelig

Alan Lowne says: "I started Saelig as an engineer - for engineers! Saelig - an Olde English word meaning 'happy, prosperous, blessed' - which is what I want for my customers, my staff, and myself! See my blog for behind-the-scenes company details, business hints, kudos, etc."

Saelig .com
unique electronics
888-7SAELIG info@saelig.com

We search the world for unique, time-saving products - see our website for 100s more: volt/temp data loggers, wireless boards, LCD display kits, Ethernet/IO, USB/RS232/485, USB-OTG, instant Ethernet-serial, CAN/LINbus, USB cables/extenders, line testers, logic analyzers, color sensors, motion controllers, eng. software, wireless boards, SMD adapters, I2C adapters, FPGA kits, GPS loggers, automotive testing, security dongles, video motion detectors, crystals/oscillators, custom switches, barcode scanners, robots, DSP filters, PLCs, Remote MP3 players, etc. Don't forget to ask for your FREE Starbucks card with your order! Check ***www.saelig.com*** *often for special offers, bargains, business hints, blog, etc.*



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

A Reactive Necessity

I don't know how we got started on this conversation at the party last night, but we were all sitting around the fire pit and the subject of sports obsessions came up. Almost all of the wives started describing the various lifestyle accommodations they had to deal with during the baseball, basketball, and football seasons. Sympathy was offered to those dealing with multiple sports or seasons that never seemed to end. There was an almost unanimous opinion among the wives that their husbands were absent a good deal of the time fulfilling various obsessions.

While I do watch an occasional event, I felt confident that I was going to receive the "dream husband" award when I piped in with, "I don't have any sports obsessions."

But then, from the other side of the fire pit, my wife yelled, "You'd have to take time off from all your other compulsions to fit it." Amid laughs from the crowd she continued, "Have you looked at all the black boxes, flashing lights, and wires around here lately?"

OK, Dear. You got me. ;-)

It's laughable, but true. It seems like I'm always pulling wires to connect one thing or another. In the early days, it was RG-6 cable everywhere to distribute cable TV and C-band satellite signals. Then came speakers, security system sensors, video cameras, etc. The real volume wiring started with my forever-evolving and expanding home control system (HCS) iterations—and it has never stopped. Let's just say that the latest wiring is mostly CAT-5, and I'm on my fifth 1,000′ reel since I started expanding the HCS to include web-interactive monitoring about six years ago.

Call it an obsession if you want to, but I like the satisfaction of knowing what's going on. There is nothing worse than coming home from a trip and finding a foot of water in the house or getting a call from one of those vacation home monitors screaming "low temperature alert"—then what? In fact, during the last two years, my oil heating system has hiccuped twice while I was 1,000 miles away.

Fortunately, I can diagnose a lot via the web through a combination of video cameras and temperature sensors. A web cam pointed at the boiler's digital control panel shows the LEDs representing the current state of the thermostats, burner, and zone pumps. It's a simplification, but let me just say that being told I have a "low temperature alert" is better understood when I can go see that the bedroom thermostat is indeed calling for heat, the zone pump for the bedroom is on, and the physical room temperature is 48°F—obviously, that particular zone pump is bad and I need to call the oil maintenance guys to replace it.

For the most part, the quantity and type of sensors I've connected to my HCS have been predicated on catastrophe avoidance rather than audio/video switching or lifestyle enhancement. I still feel that's the way it should be; but today, there's a new wrinkle. All of the energy monitoring electronics on my new PV system, along with the prospect of $5/gallon home heating oil, suggests that I should be doing more "oil monitoring" too. How long does each zone pump run (i.e., Which zone is sucking all of the heat?)? What is the total oil burner run time (i.e., How much total oil consumption?)? What are the temperature swings in each heating zone (i.e., Heat transfer rates, dirty filters?)?

The first rule in home-brew sensor connection: Do No Harm. Basically, try not to trash your heating system or blow up your house when you accidentally short the connecting wires or the HCS self-destructs. Recording zone run times requires HCS sensors on the pump control circuits or the physical zone pipes. My initial thought was just to bond a 140° bimetallic sensor on each zone pipe, but it was grossly inaccurate. The burner might run for 5 min. and the zone pump for 10, but the amount of time for the pipe to warm and cool enough to switch the sensor depends on the rate of heat transfer in each zone—too many variables. For later analysis, and specifically to see if there was any interesting correlation among those variables, I attached a Xytronics four-channel temperature monitor on the pipes. But as for connecting the HCS to the furnace, I was dealing with the 24-VAC actuator signals.

Converting a 24-VAC on/off actuator signal into isolated contact-closure output to the HCS is a relatively easy concept. Because there are six actuator signals, however, doing it with the least current requirement from the furnace power supply was the challenge. Ultimately, I chose to use capacitance reactance. Typically used in transformerless power supplies or trickle chargers, the technique utilizes capacitance reactance to limit current in a Zener-regulated AC-to-DC converter. The converter's DC output powers a low-current reed relay whose isolated contacts go to an HCS input. Certainly, I could have dug through the junk box for some appropriate opto-22 commercial solution, but going back and experimenting with basic electronics was fun for an evening, just like reading *Circuit Cellar*. Besides, as a home-brew electronic answer, it truly does no harm.

steve.ciarcia@circuitcellar.com



PC Based Logic Analyzers

NEW Model DV3400

- Faster Sampling (200/400 Msps)
- More Channels (36)
- More Memory (4x)
- Advanced Match circuits (8)
- ±6V Adjustable Threshold (x2)
- Cascadable Sequencers (4)
- Enhanced Compression

from $499.00

DigiView



Professional Hardware Capture + Software Analysis

- Automatic Real-time Hardware Compression eliminates the need to reduce resolution. Our newest version makes "dead time" insignificant.
- Edge and Pattern Triggers on all models. Advanced Model also includes Range (>, <, =, !=, >=, <=) and Stable Matches with Duration, and 4 flexible cascadable sequencers with pass-counters.
- Pattern Searches with Match & Duration.
- Specialized Sequential Searches for Serial and State Mode Signals.
- Display I²C, Synchronous (SPI), Asynchronous (RS-232), State, Boolean, Bus and Analog Data.
- Single or Dual Waveform Views.
- Resolution Zoom in Wave Views.
- Time based Link Groups for all views.
- Specialized Exports from Data Tables and List Views.
- Multi-Signal Data Tables.
- Drag & Snap Markers.
- "Click to Center" function.
- "Snap Previous/Next" function.
- Print or Save Images with comments.
- USB 2.0 (480 Mbps).
- USB 1.1 compatible (12 Mbps).

Very flexible, "easy to configure" Advanced triggering.

PICmicro® MCU Programmer

Multi-Function, In-Circuit & Gang Operation

only $199.00



QuickWriter™

See our Memory Emulators from $179.00

TechTools

www.tech-tools.com

(972) 272-9392 • sales@tech-tools.com

Copyright © 2007 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

Propeller Proto Board

Parallax's Propeller Proto Board is a low cost, high quality solution for permanent projects using the Propeller chip. The Propeller Proto Board is now available in Serial and USB versions. For projects which do not need the USB interface after programming, we recommend purchasing the Propeller Proto Board (Serial Version) and Prop Plug (not pictured; #32201; $24.95) as the reusability of the Prop Plug reduces overall cost when programming multiple boards.

Key Specifications of the Propeller Proto Boards:

- Power requirements: 6-9 VDC 300 mA - 1.5 A (current requirement depends on application)
- Dimensions: 3 x 4 in (7.62 x 10.16 cm)
- Operating temp range: Industrial (-40 °C - 85 °C)

Order the optional Accessory Kit which includes all the parts necessary to connect peripherals like VGA, keyboard, mouse and servos to the Propeller Proto Board.

Pictured below from left to right: Propeller Proto Board - Serial (#32212; $19.95), Propeller Proto Board - USB (#32812; $39.95), Propeller Proto Board Accessory Kit (#130-32212; $14.95)



Enter your Propeller-based project in the Parallax 2008 Propeller Design Contest for a chance to win $4000 in cash and prizes!

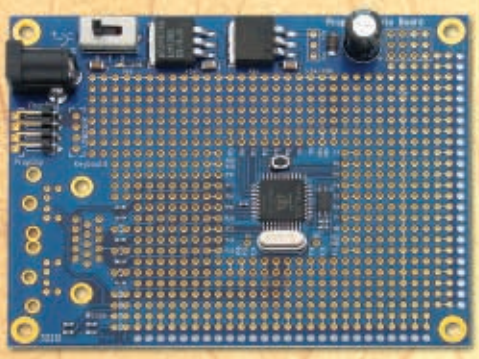
Projects will be judged on the following factors: Propeller Design Appropriateness, Originality, Professionalism, and Practicality.

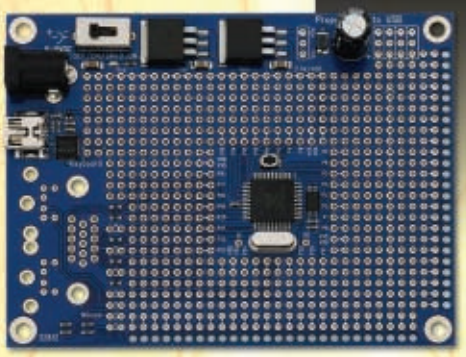
Prize money distribution:

- First Place Prize - $2,000*
- Second Place Prize - $1,000*
- Third Place Prize - $700*
- 3 Honorable Mention - $100 ea.*

** or equivalent in Parallax products*

Hurry! Deadline to sign up for a project number is September 1, 2008.







Order the **Propeller Proto Board** (Serial #32212 or USB #32812) online at http://www.parallax.com or call toll-free 888-512-1024 (Mon-Fri, 7a.m. - 5p.m., PDT).

PARALLAX inc.
www.parallax.com

Propeller, Parallax and the Parallax logo are trademarks of Parallax Inc.