

www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

ROBOTICS

Robotic Arm

Servo Control

Rolling Robot

Circuit Cellar:
Then & Now



03 >

7 25274 75349 9

\$4.95 U.S. (\$5.95 Canada)

CIRCUIT CELLAR (1988 to Present)

"Inside the Box Still Counts" — This was the title of my very first Circuit Cellar INK editorial 19 years ago. Two hundred issues later I still believe it. Like solder being my favorite programming language, I still believe that however appliance-like embedded control and personal computing implementations become, we can't forget that the process of achieving that goal isn't instant. In order to create the sophisticated devices and technologies regarded as off-the-shelf, many people still have to maintain real expertise in rudimentary design skills. Basically, somebody always has to know what's inside the box.

—Steve Ciarcia, "Inside the Box Still Counts," *Circuit Cellar* 200, 2007

1988

- Jeff Bachiochi's first article ("RS-232 Economic Tradeoffs: Board Space vs. Parts Count vs. Parts CoSt," *Circuit Cellar* 5)
- Tom Cantrell's first article ("RISC vs. Reality: An Exercise in Acronyms," *Circuit Cellar* 1)
- Ed Nisley's first article ("High Security on a Budget: Build a Video Hand Scanner/Identifier," *Circuit Cellar* 1)
- Steve Ciarcia starts the ball rolling ("Inside the Box Still Counts," *Circuit Cellar* 1)

1989

- Ed Nisley on cache ("Cache Crazy," *Circuit Cellar* 11)
- Steve Ciarcia presents the results from the first reader survey ("First INK Reader Survey," *Circuit Cellar* 7)

1990

- Steve Ciarcia comments on analog technology ("An Analog State of Mind," *Circuit Cellar* 13)
- Tom Cantrell reports on the Second Microprocessor Forum ("Earthshaking Chips," *Circuit Cellar* 13)

1991

- Ed Nisley presents data for the Furnace Firmware Project ("The Furnace Firmware Project Concludes: Hard Data for Home Control," *Circuit Cellar* 21)
- Steve Ciarcia on portable PC technology ("Why Portable?," *Circuit Cellar* 20)

1992

- Jeff Bachiochi on electronic identification technology ("Electronic Identification," *Circuit Cellar* 24)
- Ed Nisley on IR home control ("Infrared Home Control Gateway," *Circuit Cellar* 26)
- Steve Ciarcia describes a night of automated home control ("A Night in the Life," *Circuit Cellar* 26)

1993

- Circuit Cellar goes monthly (*Circuit Cellar* 31)
- Tom Cantrell on LED technology ("Smart LEDs," *Circuit Cellar* 31)
- Steve Ciarcia on the race for PC power ("The Race for Power," *Circuit Cellar* 39)

1994

- Dave Tweed's first article ("Designing Real-Time Embedded Software Using State-Machine Concepts," *Circuit Cellar* 53)
- Circuit Cellar's 50th issue (September 1994)

1995

- Tom Cantrell provides a comparison of speed and handling tips and tricks ("A Saab Story: A Tale of Speed and Acceleration," *Circuit Cellar* 57)
- Fred Eady's first article ("Take Your PIC: A Look at the PIC16Cxx Family," *Circuit Cellar* 65)

1996

- Ed Nisley updates his ImageWise receiver by adding an HCS TV display ("Firmware Furnace (Part 1): Getting Vid-Link in Sync," *Circuit Cellar* 66)
- Steve Ciarcia describes an enlightening experience at the Trinity Robotics Competition ("Bots Got No Respect," *Circuit Cellar* 71)

1997

- Tom Cantrell describes micropower impulse radar (MIR) technology ("Radar Love," *Circuit Cellar* 79)
- Fred Eady on networking embedded and desktop PCs ("Sweet Solution," *Circuit Cellar* 79)

1998

- Design98, Sponsored by Microchip, 1998
- Steve Ciarcia and Jeff Bachiochi design an entry and exit system ("Golcha! Alarming the Alarm System," *Circuit Cellar* 95)

1999

- Design99, Sponsored by Motorola, 1999
- George Martin's first Lessons From the Trenches column ("Off-the-Shelf Data Acquisition Using Visual Basic," *Circuit Cellar Online*)
- Jeff Bachiochi on JTAG ("JTAG: Working with CoolPID," *Circuit Cellar* 104)

2000

- PIC 2000, Sponsored by Microchip, 2000
- Design2K, Sponsored by Philips, 2000
- Steve Ciarcia comments on Y2K, ("You've Got to be Y2Kidding," *Circuit Cellar* 115)

2001

- Driven to Design, Sponsored by Zilog, 2001
- Design Logic 2001, Sponsored by Atmel, 2001
- Ultra-Low Power Flash MCU MSP430 Design Contest, Sponsored by Texas Instruments, 2001
- Steve Ciarcia introduces the Electronic Edition ("Electronic Evolution," *Circuit Cellar* 126)

2002

- PSoC Design Challenge 2002, Sponsored by Cypress MicroSystems, 2002
- Mad Dash for Flash Cash, Sponsored by Microchip, 2002
- Jeff Bachiochi describes a smart brake light design ("Smart Auto Brake Light Eliminates Turn Indicators," *Circuit Cellar* 148)
- Tom Cantrell on new sensor technology ("Sensors and Sensibility," *Circuit Cellar* 148)

2003

- Renesas H8 Design Contest, Sponsored by Renesas, 2003
- Zilog Flash for Cash Z8 Encore! International Design Contest, Sponsored by Zilog, 2003
- Motorola E-Field Sensor Contest & Flash Innovation 2003, Sponsored by Motorola, 2003
- Fred Eady on USB connectivity ("Mission Possible: Achieve Cheap USB Connectivity," *Circuit Cellar* 157)

2004

- AVR 2004 Design Contest, Sponsored by Atmel, 2004
- PSoC High Integration Challenge, Sponsored by Cypress MicroSystems, 2004
- Zilog 2004 Flash Nets Cash Design Contest, Sponsored by Zilog, 2004
- Wireless Design Challenge, Sponsored by Freescale Semiconductor, 2004
- Jeff Bachiochi covers USB in embedded designs ("USB in Embedded Design," *Circuit Cellar* 165)

2005

- M16C Design Contest, Sponsored by Renesas, 2005
- Fred Eady explores embedded Wi-Fi ("Embedded Wi-Fi with TRENDnet," *Circuit Cellar* 174)
- Jeff Bachiochi on VoIP ("A Fresh Look at VoIP," *Circuit Cellar* 180)

2006

- DesignStellaris2006 Contest, Sponsored by Luminary Micro, 2006
- Atmel AVR Design Contest 2006, Sponsored by Atmel, 2006
- Tom Cantrell covers using the 'Net as a long RS-232 cable ("Device Surfer," *Circuit Cellar* 192)
- Ed Nisley provides tips for choosing voltage references ("Voltage References," *Circuit Cellar* 193)

2007

- George Martin is back with a C language tutorial ("Hello World ... Want Cookie," *Circuit Cellar* 198)
- Steve Ciarcia describes how he used his HCS to save a life ("A Home Control Event Worth Remembering," *Circuit Cellar* 199)

Without a continuous effort at understanding and improving present achievements, we cannot progress to higher levels of achievement. Circuit Cellar Ink is a publication designed to increase that awareness. We must not ignore the fact that it is a combination of people AND machines that creates intelligent personal systems. Whether they be applied as toaster-like appliances throughout an industry, serve as a control system of a CAT scanner, or function as a video arcade game, the basic ingredients of computers are similar. It is the continual evolution of a computer's concept, design, and application which ultimately results in the perfection of the truly Intelligent Personal System.

— Steve Ciarcia, "Inside the Box Still Counts," *Circuit Cellar* 1, 1988

Announcing the
NetBurner PK70
EMBEDDED CONTROL PRODUCT KIT

Create a finished product in one day

The design of a finished product



Aluminum Enclosure
Customizable Logo
Built-in Power Supply

The flexibility of a module



32-Bit Freescale ColdFire 147 MHz CPU
8MB SDRAM - 2MB Flash Memory
SD/MMC Flash Card Support
I2C - SPI - Address and Data Bus
10/100 Ethernet
3 UARTs

The power of NetBurner's Development Suite



NetBurner Eclipse IDE
Graphical Debugger
 μ C/OS RTOS
ANSI C/C++ Compiler and Linker
Deployment Tools
Example Programs

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

Customize
with **NetBurner**
Personality
Boards

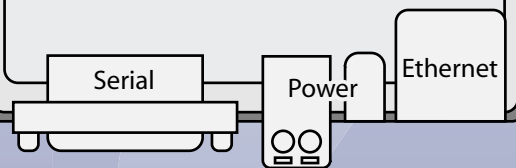


**Use a standard NetBurner
Personality Board**

- Analog to Digital Converter Board
- Digital I/O ADC Combo Board
- Digital I/O Board

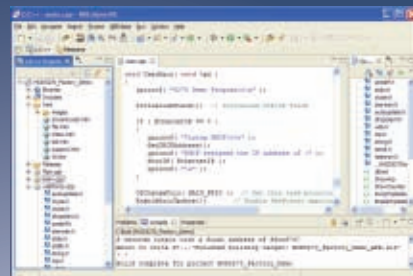
**Or create your own board with a
custom connector**

Integrated MCF5270 based
processor board with Ethernet



Featuring

NetBurner's Eclipse IDE



**NetBurner**
Networking in One Day!

one | 1-800-695-6828

 **freescale**[™]
Alliance Member

CapSense. Beauty is more than skin deep.

PSoC®-based Capacitive Touch Sensing

Maximize the design flexibility and integration of Cypress's PSoC solution to create a stylish, durable interface. CapSense replaces buttons, switches, sliders and other mechanical inputs in your product. CapSense enables:

- Fast changes to your design at any stage from concept through production. CapSense is not a fixed-function ASIC or module; you are in control of your design at all times.
- Single-chip implementation supporting multiple interfaces – buttons, sliders, touch screens, touchpads and proximity detectors – on a variety of conductive substrates.
- Unique integration of additional functions – LED control, battery monitoring, motor control, ambient light sensing, etc. – all with the same CapSense chip.
- Quick time-to-market with powerful, visual embedded design tools allowing customized, system-level design.

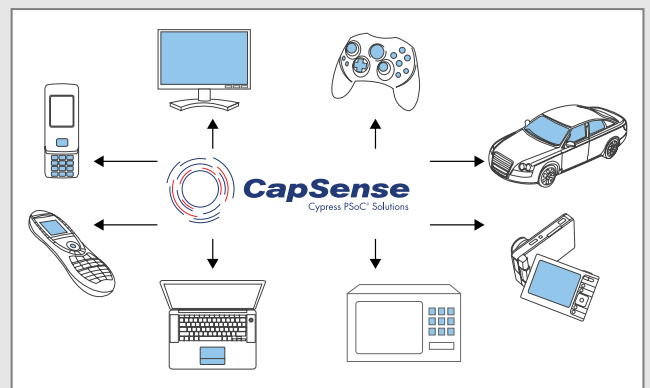
GET STARTED WITH CAPSENSE NOW

Order a discounted CapSense Development Kit:
www.cypress.com/capkit

Request free PSoC CapSense IC samples:
www.cypress.com/capchips

Download free PSoC Express™ visual embedded software:
www.cypress.com/capexpress

Register for a CapSense NetSeminar:
www.cypress.com/capseminar



Applications enabled by PSoC® CapSense.

Streamline your next design with CapSense:
www.cypress.com/gocapsense





Link Instruments

PC-Based Test Equipment

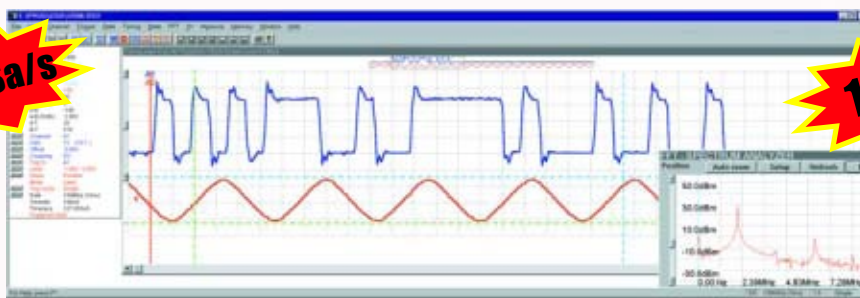
Digital Oscilloscopes

NEW!



- 2 Channel Digital Oscilloscope
- **500 MSa/s** max single shot rate
- 1Mpt sample memory
 - 250 MSa/S (Dual channel) 512 Kpts
 - 500 MSa/S (Single channel) 1 Mpts
- Advanced Triggering
- Only 9 oz and 7" x 3.5" x 1.5"
- Portable and Battery powered
- USB 2.0
- Advanced Math
- FFT Spectrum Analyzer
- \$950 (DSO, Probes, Software & power supply)

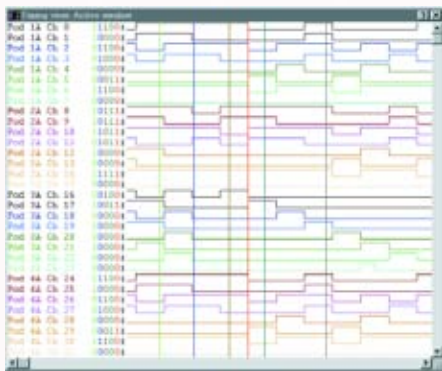
500MSa/s



1Mpts

Windows Screenshot

Logic Analyzers



Windows Screenshot



- 40 to 160 channels
- up to 500 MSa/s
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- up to 512K samples/ch
- USB 2.0 and Parallel Interface
- Pattern Generator option

| | |
|-------------------------|---------------|
| LA5240 (200MHz, 40CH) | \$1700 |
| LA5280 (200MHz, 80CH) | \$2350 |
| LA5540 (500MHz, 40CH) | \$2500 |
| LA5580 (500MHz, 80CH) | \$3500 |
| LA55160 (500MHz, 160CH) | \$7500 |



Link Instruments (973) 808-8990

17A Daniel Road East · Fairfield, NJ 07004 · Fax (973) 808-8786

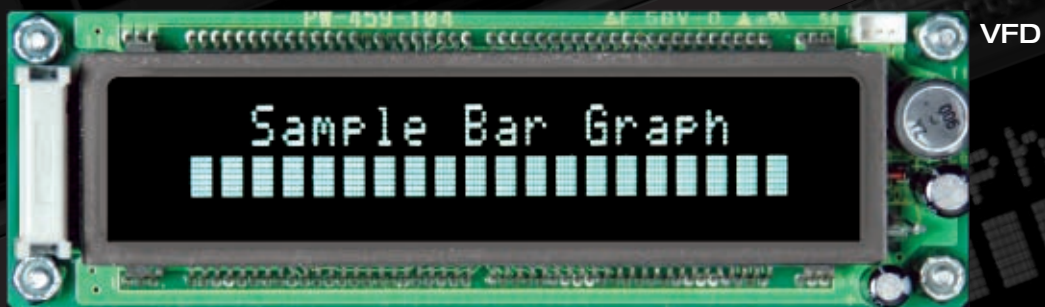
www.Linkins4.com

U-VERSION VFD

Noritake

Drop-in Replacement for LCD Modules

The new family of U-Version modules is compact, low power and lower in cost with a custom designed chip. This new VFD technology has an 8 and 4 bit parallel interface and enables the replacement of LCD's with Noritake U-Version VFD modules.



**UPGRADE
NOW...**

- Brightest Display
- Widest Viewing Angles
- Widest Temperature Range
- Fits Right In
- No Programming Change
- Low Cost

www.noritake-elec.com/53

TASK MANAGER

Issue 200

Remember the days when you had to pull into a gas station and drop a coin in a pay phone to make an important call? Today you use your Bluetooth-enabled cell phone as you're in transit.

Remember how you used to have to mark up an actual map before you took off on a road trip? Today you simply log your destination on a web site and hit Print. Some of you even have interactive navigation systems that tell you how to avoid traffic jams and toll booths on the way to your destination!

Remember how you used to have to order all of your parts (from MCUs to LEDs) with a mail-in form or over the phone? Today you simply click a few buttons and everything you need is shipped overnight.

I could go on and on for another six paragraphs. The point is, a lot of things have changed since 1988. But one thing hasn't: a brand new edition of *Circuit Cellar* hits the newsstands each month. And for those of you who have been subscribers since the beginning, you have 200 great issues to prove it.

This month we're celebrating the 200th edition of *Circuit Cellar*. To do so, we've packed this issue with great feature articles, columns, and more. For instance, check out our special foldout cover if you haven't already. There, we provide you with a look at the history of *Circuit Cellar*. It's amazing how writers like Steve Ciarcia, Tom Cantrell, Jeff Bachiochi, and Ed Nisley have been so far ahead of the technological curve for so many years. They've been covering the technologies of tomorrow in the pages of *Circuit Cellar* ever since the Reagan administration!

We also have two special feature articles this month. On page 40, Dave Tweed describes many of the technologies that have been covered in *Circuit Cellar*. On page 72, Ed Nisley tells us about some of his all-time favorite projects. Remember the soda bottle launcher project from Issue 2? We still get letters and e-mails about that one!

As usual, we also have a healthy batch of theme-related articles to keep you busy. The Robotics issue is always a crowd pleaser. This month we present a variety of projects that won't disappoint.

On page 14, Michael Hall, Aaron Patten, and Erin Simpson describe how they designed and built a complete control system for a robotic manipulator arm. They updated an old arm and designed a new control system that's capable of complex, repeatable actions. The system consists of a main manipulator arm and a teaching arm used to control it.

Ever wonder how moviemakers get their monsters to look so real? In "Animatronic System Control," Barry Stout and Eugene Zeldin describe how to build a controller for an animatronics system (p. 24). The system controls a robotic monster's head, eyes, and mouth. Although the controller was originally designed for use on the set of a horror film, you can build a similar system for any number of applications.

Jeff Bingham and Lee Magnusson's inertial rolling robot is definitely a novel design (p. 34). A DC electric motor is attached to a pendulum and suspended inside an inflated ball. They describe everything from the H8/3664-based circuitry to the process of cutting the rubber ball and patching it up again.

To round off this group of robotics-related articles, Dale Wheat describes how he designed his Servo Tester I and Servo Tester II systems. The latter features a second servo channel and an Auto Sweep function. You can build a similar system and tweak the code to suit your needs.

These projects will keep your gears turning well into April and May. Go animate!

Before I sign off, let me leave you with a little bit of encouragement: you're the engineers whose designs will change the way people live, work, and interact in the 21st century. Work hard, read (*Circuit Cellar*, of course), and keep us up to speed on your progress!

cj@circuitcellar.com



CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR

Steve Ciarcia

CHIEF FINANCIAL OFFICER

Jeannette Ciarcia

MANAGING EDITOR

C.J. Abate

MEDIA CONSULTANT

Dan Rodrigues

WEST COAST EDITOR

Tom Cantrell

CUSTOMER SERVICE

Debbie Lavoie

CONTRIBUTING EDITORS

Jeff Bachiochi

Ingo Cyliax

Fred Eady

George Martin

Ed Nisley

CONTROLLER

Jeff Yanco

ART DIRECTOR

KC Prescott

GRAPHIC DESIGNER

Mary (Turek) Sobuta

STAFF ENGINEER

John Gorsky

NEW PRODUCTS EDITOR

John Gorsky

PROJECT EDITORS

Steve Bedford

Ken Davidson

David Tweed

ASSOCIATE EDITOR

Jesse Smolin

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 © 2007 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.

Only 4 Steps...

...are required to generate efficient, reliable applications with the μ Vision IDE and development tools from Keil.

Step 1. Select Microcontroller and Specify Target Hardware

Use the Keil Device Database (www.keil.com/dd) to find the optimum microcontroller for your application.

In μ Vision, select the microcontroller to pre-configure tools and obtain CPU startup code.

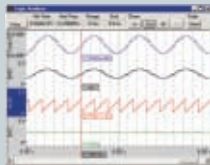
Step 2. Configure the Device and Create Application Code

The μ Vision Configuration Wizard helps you tailor startup code to match your target hardware and application requirements.

Extensive program examples and project templates help you jump-start your designs.

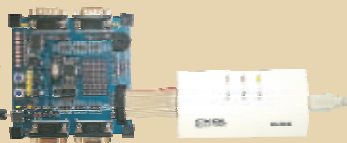
Step 3. Verify Program Execution with Device Simulation

High-speed simulation enables testing before hardware is available and helps you with features like instruction trace, code coverage, and logic analysis.

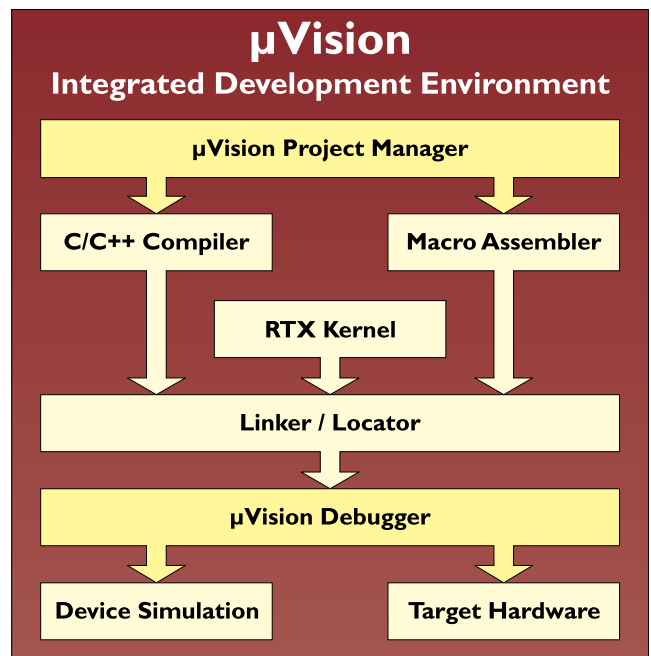


Step 4. Download to Flash and Test Application

Once your application is runs in simulation, use the Keil ULINK USB-JTAG Adapter for Flash programming and final application testing.



Keil Microcontroller Development Tools help you create embedded applications quickly and accurately. Keil tools are easy to learn and use, yet powerful enough for the most demanding microcontroller projects.



Components of Keil Microcontroller Development Kits

Keil makes C compilers, macro assemblers, real-time kernels, debuggers, simulators, evaluation boards, and emulators.

Over 1,200 MCU devices are supported for:

- **8-bit** - 8051 and extended 8051 variants
- **16-bit** - C166, XC166, and ST10
- **32-bit** - ARM7, ARM9, and Cortex-M3

Download an evaluation version from www.keil.com/demo

March 2007: Robotics

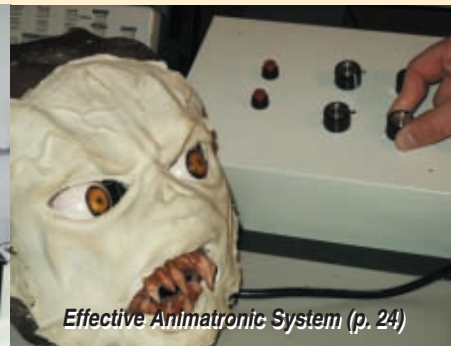
FEATURES

- 14 **Robotic Arm Control System**
Michael Hall, Aaron Patten, & Erin Simpson



Control a Robotic Arm (p. 14)

- 24 **Animatronic System Control**
Barry Stout & Eugene Zeldin



Effective Animatronic System (p. 24)

- 34 **Inertial Rolling Robot**
Jeff Bingham & Lee Magnusson



Rolling Robot Design (p. 34)

- 44 **Generic Modbus Simulator (Part 1)**
Theory and Preparation
Aubrey Kagan



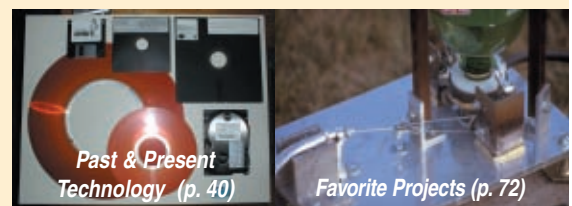
Servo Tester (p. 59)

- 59 **Servo Control**
Dale Wheat

ISSUE 200 SPECIAL FEATURES

- 40 **Then and Now**
Dave Tweed

- 72 **A Plethora of Projects**
Ed Nisley



Past & Present Technology (p. 40)

Favorite Projects (p. 72)

COLUMNS

- 52 **APPLIED PCs**
Build a PIC Platform
Fred Eady

- 67 **LESSONS FROM THE TRENCHES**
More "Hello World"
Moving What You've Learned to the Hardware
George Martin

- 75 **FROM THE BENCH**
Embedded USB Breakthrough
Jeff Bachiochi

- 80 **SILICON UPDATE**
Code Like the Wind World Tour
Tom Cantrell

DEPARTMENTS

- 4 **TASK MANAGER**
Issue 200
C.J. Abate

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

- 93 **CROSSWORD**

- 94 **INDEX OF ADVERTISERS**
April Preview

- 96 **PRIORITY INTERRUPT**
Inside the Box Still Counts
Steve Ciarcia



MYKLE-06

AVR picoPower Microcontrollers

To meet the tough requirements to modern microcontrollers Atmel® has now combined ten years of low power research and development into picoPower™ technology for AVR® microcontrollers. picoPower enables AVR to achieve the industry's lowest power consumption with 650 nA with a real time counter running and 100 nA in deep sleep.

What can AVR picoPower do for your design?

- True 1.8V supply voltage enabling operation of all features and core down to 1.8V
- Minimized leakage current enabling 100 nA Power Down sleep consumption
- Sleeping brown-out detector enabling full protection with no power penalty
- Ultra low power 32 kHz crystal oscillator enabling operation at only 650 nA

For more information, check out www.atmel.com/ad/picopower



DUAL 36-V LOW-LOSS CONTROLLERS

The LTC4416 and LTC4416-1 are robust dual “ideal diode” controllers for driving small and large gate capacitance P-channel MOSFETs in power-switchover circuits. The device’s control circuitry allows two power supplies to operate in tandem, ensuring a highly efficient O-ring of these power sources. The 25-mV forward voltage of the LTC4416 and LTC4416-1’s controlled PFET is significantly lower than a Schottky diode, thereby extending battery life while reducing power loss and heat. Applications include systems that typically take power from multiple input sources, such as high-current power path switches, uninterruptible power supplies, battery backup systems, emergency systems with battery backups, logic-controlled power switches, and automotive and industrial systems.

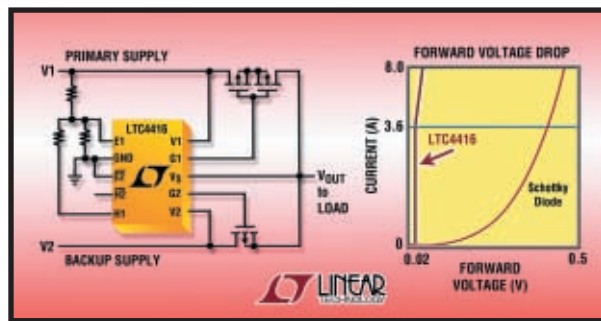
The LTC4416 is designed to work with slow-moving input supplies and provide a “soft-off” feature to minimize droop when the supply voltages change. The LTC4416-1 is optimized for rapidly moving

input supplies that must be quickly turned off. This feature is useful when protecting the load from a large input transient or preventing loading of the power supply when it is deemed undesirable. The LTC4416 and LTC4416-1 are guaranteed to meet performance specifications over a wide range of operating conditions including an ambient temperature range of -40° to 125°C and a voltage range of 3.6 to 36 V.

The LTC4416 and LTC4416-1 feature ultra-low 70- μA quiescent current (35 μA per channel), independent of the load current. In addition, the ICs use a strong gate drive for fast gate turn-on and turn-off times of 60 and 30 μs , respectively.

The devices also provide reverse battery-discharge protection and MOSFET gate-protection clamp circuitry.

Pricing for the LTC4416EMS/-1 and LTC4416IMS/-1 starts at \$2.45 and \$2.95 each, respectively, for 1,000-piece quantities.



Linear Technology Corp.
www.linear.com

CURRENT SENSORS FOR AUTOMOTIVE BATTERY-MONITORING APPLICATIONS

The HAB is a new family of current sensors for battery monitoring in automotive systems and similar applications. The 12 sensors in the family offer primary current measurement ranges from ± 20 to ± 120 A, a high accuracy of $\pm 2\%$ across the operating temperature range of -40° to 125°C , and a choice of voltage or PWM outputs.

The sensors are based on open-loop, Hall-effect transducers that simplify both installation and servicing by removing the need to cut the cable carrying the measured current. They operate from a unipolar 5-V supply and provide a fully ratiometric solution for the voltage output devices. Response time is less than 10 ms.

A watertight housing and sealed connector provide full environmental protection in under-the-hood applications. Principal applications are expected to be in the measurement of battery-pack current in electric, hybrid, and conventional vehicles.

One hundred-piece pricing for the HAB-S is \$27.50. Pricing for the HAB-V is \$33.90.

LEM USA
www.lem.com



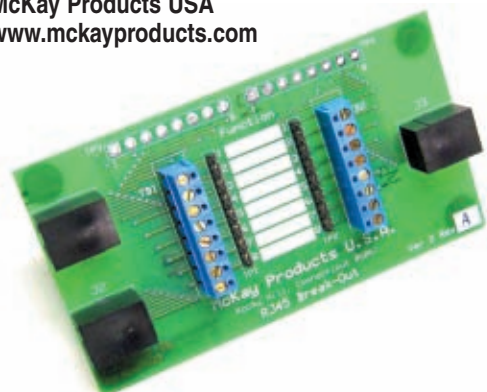
RJ-45 BREAKOUT BOX

The RJ-45-BOB is an RJ-45 breakout box that brings out all the connections to test points and binding posts for the now common RJ-45 connector found on late-model, mobile, and amateur radios, as well as Ethernet and serial cables, T1 lines, and a host of other interface applications.

Features of the RJ-45-BOB include gold-plated test points, binding post connections, and solder points for connection to custom test equipment. It also offers a label area for each RJ-45 pin. One side of the board has two RJ-45s in a “loop-thru” configuration while the other side has an RJ-45 in a “dead-end” configuration.

The unit costs \$34.95 and quantity discounts are available.

McKay Products USA
www.mckayproducts.com



NEW PRODUCT NEWS

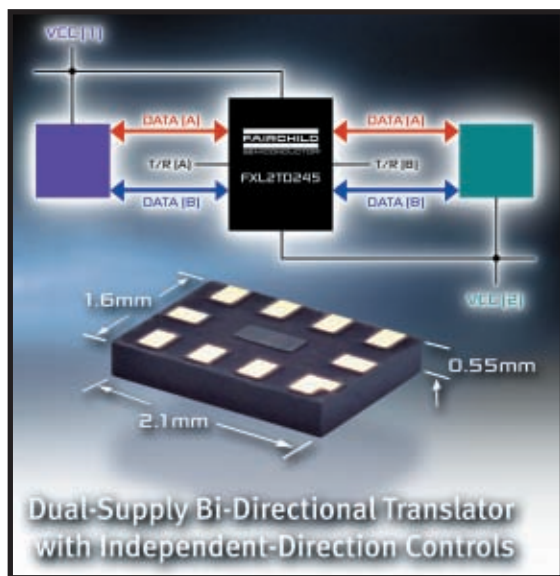
DUAL-SUPPLY BIDIRECTIONAL TRANSLATOR

The FXL2TD245 is the first dual-supply bidirectional translator on the market that's configurable for both unidirectional and independent bidirectional voltage translation between two logic levels. In addition to offering designers unparalleled design flexibility in a wide variety of low-voltage applications, this translator helps save board space. The FXL2TD245 measures only $0.55 \times 1.6 \times 2.1$ mm, which is 80% smaller than comparable dual-supply translators in SOIC packages. It also replaces two 1-bit components used in typical designs. This ultra-compact translator is ideal for use in cell phones, PDAs, gaming devices, and other portable applications, although its wide voltage range (1.1 to 3.6 V) accommodates the voltage requirements of a variety of consumer and industrial applications.

In addition to performance and space improvements, the FXL2TD245 bolsters system reliability. For instance, its 10-terminal MicroPak offers 35% more pad contact area than other leadless packages, which results in a stronger solder bond between the device and the circuit board. The FXL2TD245 facilitates more robust designs by providing outputs that switch to tristate if either supply voltage is equal to ground. It also offers built-in power-off protection.

The FXL2TD245 is priced at \$0.85 each in 1,000-piece quantities.

Fairchild Semiconductor Corp.
www.fairchildsemi.com



NEW MCUs PROVIDE ENHANCED MEMORY AND PIN COUNTS

NEC has added 16 new devices to its comprehensive lineup of 16- and 32-bit all flash memory microcontrollers. These latest microcontrollers offer up to twice the memory and a significant increase in pin-count options compared to previous devices in the line. Based on advanced 0.15-micron processes using SuperFlash, the new lineup includes two 32-bit V850ES/JJ3 microcontrollers with up to 1 MB of flash memory, and 14 new 16-bit 78K0R/Kx3 microcontrollers with up to 512 KB of memory and 144-pin packages. These devices are well suited for industrial and office applications such as, security systems, utility meters, scanners, and printers. They are also well suited for consumer applications such as, digital televisions, projection televisions, and DVD receivers.

The new devices currently cost \$13.50 for the V850ES/JJ3 microcontrollers and \$9 for the 78K0R/Kx3 microcontrollers in sample quantities.

NEC Electronics America, Inc.
www.am.necel.com

18-BIT DELTA-SIGMA ANALOG-TO-DIGITAL CONVERTER

The MCP3421 is the highest resolution ADC available in a SOT-23 package today. The low-power, 18-bit, Delta-Sigma ADC features an integrated voltage reference and PGA on-chip, which reduces the need for external components and enables a smaller overall design footprint. With its 6-pin package, low-power consumption, and high resolution, the new ADC is ideal for portable-measurement applications in the industrial, medical, consumer, and automotive markets.



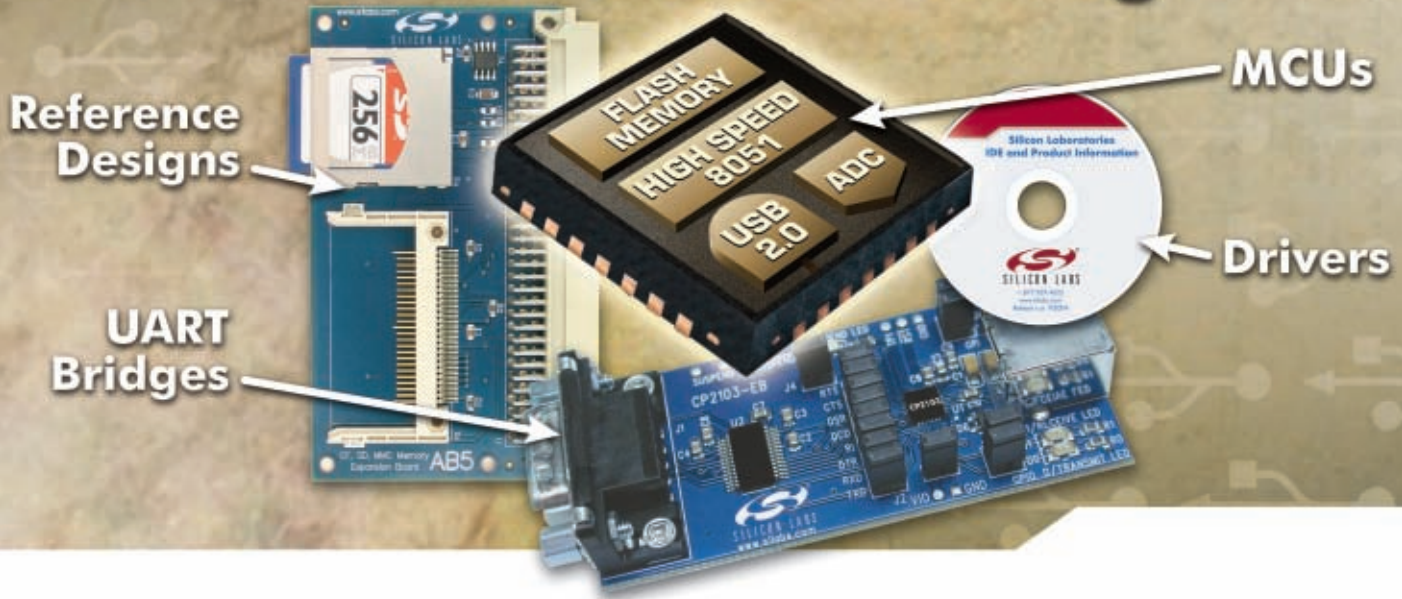
The MCP3421 ADC uses an I²C-compatible serial interface and operates from a single power supply (2.7 to 5.5 V). The low-power technology enables the device's extremely low-power consumption of just 155 μ A at 5 V of continuous conversion, which helps to extend battery life in portable measurement devices. Additional-

ly, with the device's onboard PGA, users can select the gains of $\times 1$, $\times 2$, $\times 4$, or $\times 8$ before the analog-to-digital conversion takes place, enabling very high-resolution conversion of even small input signals. The integrated voltage reference and oscillator lessen the need for external components, resulting in simpler designs and a smaller overall design footprint.

For designers wanting to evaluate the MCP3421 ADC in their applications, the MCP3421 evaluation board (part number MCP3421EV) is available for \$15. The device is available in a 6-pin, SOT-23 package for \$1.71 each in 10,000-unit quantities.

Microchip Technology, Inc.
www.microchip.com

We Provide Everything to Make USB Designs Easy



Low-Cost Embedded USB

Silicon Laboratories' extensive portfolio of USB MCUs and USB to UART Bridges include complete, low-cost development tools and drivers to make system design quick and easy. Software examples for real life systems are also available and include a mass storage device, USB audio and human interface device. The USB MCUs feature an on-board USB 2.0 function controller with an integrated transceiver that requires no external oscillator. On-chip resources include a high-speed 8051 CPU (up to 48 MIPS) with up to 64 kB Flash, multi-channel 10-bit ADC, voltage reference, internal oscillator, UARTs, SMBus, SPI, timers, counters and PWM generators.



USB to UART Bridge

- Single Chip (5x5 mm)
- USB 2.0 Controller
- UART Interface
- Update RS-232 Designs

Product details: www.silabs.com/USB



NEW PRODUCT NEWS

PROTOCOL TRANSLATION CARDS

The **ProtoCarrier** is a series of daughter cards that enable OEMs to quickly and easily interface with legacy devices to modern open-protocol networks without extensive board redesign. ProtoCessors are a family of industrial protocol coprocessors that provide comprehensive protocol translation capabilities. With access to the extensive FieldServer Technologies driver library, ProtoCessor can be embedded on a device to enable the manufacturer to meet the interoperability needs of its customer. The protocols available include Allen Bradley Ethernet/IP, BACnet IP, BACnet MSTP, Modbus RTU, Modbus TCP, Metasys N2, DNP3, LonWorks, and a wide variety of legacy and proprietary protocols.

ProtoCarrier now enables manufacturers to add the power of ProtoCessor to its device without the need of extensive board redesign. The ProtoCarrier has either an RS-232 or RS-485 port on the host side (OEM's product) and a ProtoCessor TTL socket on the board that can accept any ProtoCessor chosen to meet their user's specific need. FieldServer Technologies can preprogram the ProtoCessor to provide complete industrial or building-automation interoperability.

The ProtoCarrier with a ProtoCessor installed costs approximately \$225 in volume.

FieldServer Technologies
www.protoconnector.com



WEATHERIZED CONNECTOR

The new **CONREVSMA015** reverse-polarity SMA connector is ideal for applications requiring an external water-resistant connection. The connector features an integral rubber O-ring that compresses against the outside of the enclosure to prevent water or dust intrusion into the product's housing. Its extra-long 0.47" tail is long enough to be used with the thickest outdoor-rated enclosures. Available standard for RG-174 coaxial cable, other cable types are available through special order.

The CONREVSMA015 costs \$3.04 in quantities of 1,000.

Connector City
www.connectorcity.com



+++ NO ROYALTIES +++

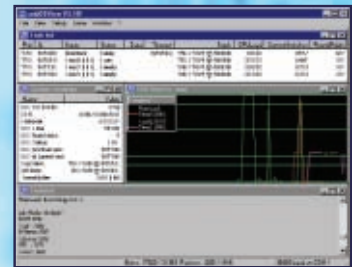
embedded software
solutions

Eval versions
available

embOS® (RTOS)

+++ 8/16/32 bits +++

Preemptive multitasking
Zero interrupt latency
Easy to use start project included
Profiling support included
Object/source code available



emWin® (GUI)

+++ 8/16/32 bits +++

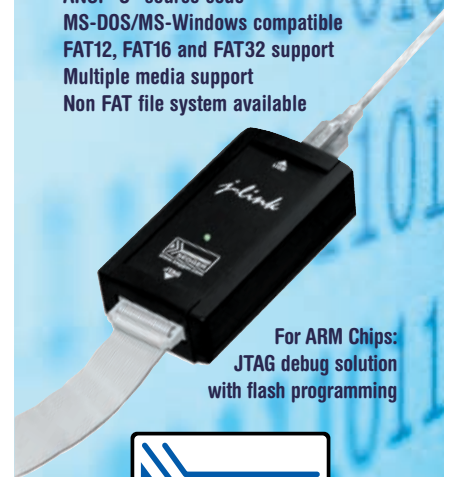
ANSI "C" source code, no C++ required
Supports b/w, grayscale and color
2D graphic library included
Variety of fonts included
PC simulation included
Window Manager/Widgets (opt)



emFile (File system)

+++ 8/16/32 bits +++

ANSI "C" source code
MS-DOS/MS-Windows compatible
FAT12, FAT16 and FAT32 support
Multiple media support
Non FAT file system available



For ARM Chips:
JTAG debug solution
with flash programming

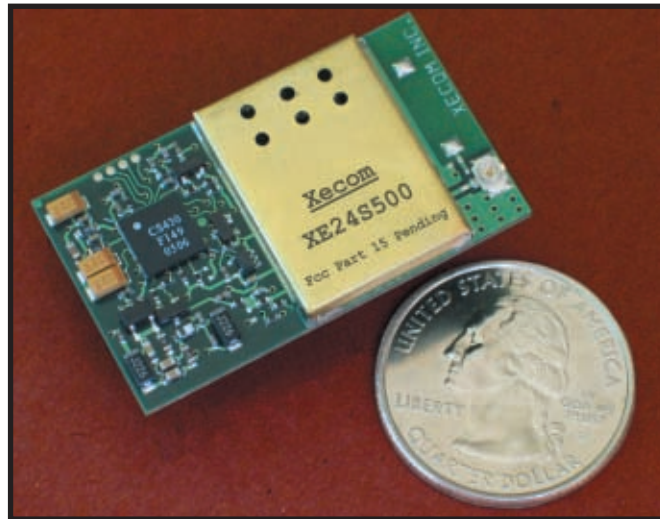


phone: 978-874-0299
www.segger.com

2.4-GHZ TRANSCEIVERS RUN APPLICATION CODE

A new feature permitting XE24S transceivers to run remote monitoring applications on the transceiver's communications controller is now available. This feature called "User Code" eliminates the need for a microcontroller at the remote node, reducing system cost and complexity.

The XE24S transceivers use the Texas Instruments MSP430 F148 to control communications functions; however, these responsibilities do not use this microcontroller's full capabilities. User Code permits the user to tap into the controller's unused capabilities to run his remote monitoring application. Note that 8 KB of flash memory space is reserved to store user-generated code. A User Code developer's CD provides the tools to use the MSP430 F148 controller without interfering with its communications responsibilities. The User Code Developer's CD includes development tools, a listing of available



system calls, and User Code examples of common system functions.

The User Code feature targets applications that use the six I/O lines on the XE24S transceivers. These applications include remote monitoring applications, such as irrigation controls or fluid tank monitoring, plus automated meter reading (AMR), security systems, and medical diagnostics.

The User Code developer's tools will be included at no extra charge in each XE24S development kit. Development kits are available for all variations of the XE24S. Each development kit costs \$295 and includes the development tools, two transceivers, two serial development boards, plus the required antennas and cables. The User Code Developer's CD is offered separately for \$49.

Xecom, Inc.
www.xecom.com

TWO- TO FOUR-CHANNEL CCFL CONTROLLERS

The DS3992 and DS3994 are two- to four-channel controllers for cold cathode fluorescent lamps (CCFLs) that are used to backlight LCDs. The DS3992 supports a one-lamp-per-channel configuration with fully independent lamp control, while the DS3994 supports configurations of one to four channels and allows multiple DS3994 controllers to be cascaded.

The DS3992/DS3994 use a push-pull drive scheme to convert a DC voltage (5 to 24 V) to the high-voltage (600 to 1,200 V_{RMS} and 300 to 1,400 V_{RMS}, respectively) AC waveform that is required to power CCFLs. The push-pull drive scheme uses a minimal number of external components, which reduces component and assembly costs and makes the PCB design easy to implement. The push-pull drive scheme also provides an efficient DC-to-AC conversion and produces near sinusoidal waveforms.

Each DS3992 and DS3994 channel drives two logic-level N-channel MOSFETs that are tied between the ends of a step-up transformer and ground. The transformer has a center tap on the primary side that is connected to the DC supply. The DS3992/DS3994 alternately turn on the two MOSFETs to create the high-voltage AC waveform on the secondary side. By varying the duration of the MOSFET turn-on times, the controllers accurately control the amount of current flowing through the CCFL.

A series resistor on the low-voltage side of the CCFL enables current monitoring. The voltage developed across this resistor is fed to the lamp-current monitor input on each device. Then, the DS3992/DS3994 compare the resistor voltage against an internal

reference voltage to determine the duty cycle for the MOSFET gates.

Each CCFL receives independent control, which results in equal brightness across all of the lamps and maximizes lamp life and brightness.

Both devices operate from 4.5 to 5.5 V. Pricing starts at \$0.60 for the DS3992 and \$0.85 for the DS3994 in 1,000-piece quantities.

Maxim Integrated Products
www.maxim-ic.com



H8SX

H8SX – 32-bit CISC, Single cycle, Powerhouse

Up to 1MIPs/MHz, fast multipliers/dividers, zero wait state Flash

Renesas Technology

No.1*1 supplier of microcontrollers in the world

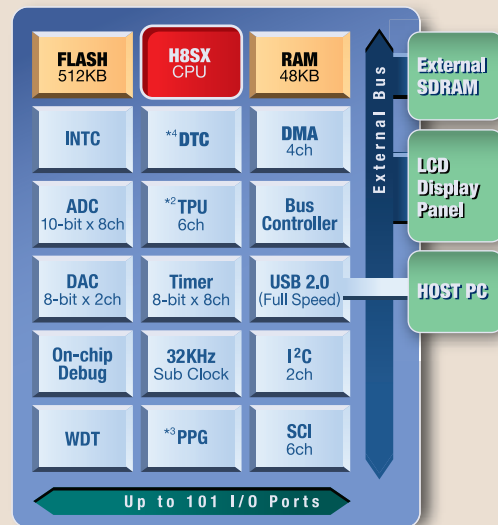
introduces the high-performance H8SX Series of microcontrollers. Upwardly compatible with the popular H8 architecture, the 32-bit, 50MHz H8SX CPU core provides the performance needed for the next generation of products. A comprehensive array of peripherals, including USB and other advanced data management functions, gives full system integration. H8SX integrated peripherals and high-speed I/O combine to provide a MCU with advanced performance in every clock cycle.

H8SX Product Roadmap

| | H8SX/1500 40MHz@5v | | H8SX/1600 50MHz@3v | | |
|------------|---|----------------|------------------------------|----------------|--------------|
| | CAN | High-speed ADC | General Purpose | High-speed ADC | USB |
| Flash Size | 100pin 120pin 144pin Coming Soon | | | 164x 163x | 166x 165x |
| 1MB | | | | | |
| 768KB | | | | | |
| 512KB | 1544 1582 | 155x | 1657 1656 1651 1650 | 163x | 1664 1663 |
| 384KB | 1527R | 155x | | 164x | 1654 1653 |
| 256KB | | | | | |
| Romless | | | | | |

*1Source: Gartner Dataquest (April 2006) "2005 Worldwide Microcontroller Vendor Revenue" GJ06333

HOT Products **H8SX/1664**
50MHz@3v



*2TPU: Timer Pulse Unit
*3PPG: Programmable Pattern Generator
*4DTC: Data Transfer Controller

Top Reasons To Select H8SX

- High-Performance**
 - 32-bit CPU with built-in hardware MAC. Up to 50MIPs
 - High-speed multipliers (single-cycle for 16-bit)
 - Majority of CISC instruction set executes in 1 cycle
 - Single-cycle access Flash (up to 1MB) and SRAM (up to 64KB) at maximum clock frequency
 - Multiple DMA engines and 3 bus architecture system, providing efficient data path management.
- Compatibility**
 - Pin compatibility within H8SX families
 - Code and peripheral compatibility with H8S devices.
- Connectivity**
 - Certified USB 2.0 Full Speed, Bus powered or Self powered
 - Six serial ports, Including IrDA and ISO7816. Two additional I²C interfaces
 - High-speed serial (921Kbps UART, 5Mbps Synchronous).
- Timers**
 - Up to 12 channels of 16-bit timers with a total of 32 input capture/output compare
 - 20ns resolution



Get Started Today -

Go online and register to be eligible for a FREE Starter Kit
www.america.renesas.com/ReachH8/d



Visit us at ESC Silicon Valley

Embedded 2007
Systems Conference
SILICON VALLEY

April 3-5, 2007

Booth #716 & #3011

Renesas Technology Corp.

Robotic Arm Control System

Need a hand? In just 11 weeks, this team of designers turned a dysfunctional 20-year-old robot arm into a system capable of complex actions. Now you can build your own.

Have you ever wished you had a third arm? We did, and now our dreams have been realized. It is our hope that you can learn from our experiences so you too can gain a helping hand. With a Microchip Technology dsPIC30F6015 microcontroller, a few mechanical components, some DC motors, and a little ingenuity, your very own third arm will be pouring your drinks in no time.

SYSTEM OVERVIEW

Our final project for Camosun College's Electronics and Computer Engineering Technology program was to implement a complete control system for a robotic manipulator arm. Our goal was to update the electronics and design a new control system that would be capable of complex, repeatable actions.

The system consists of the main manipulator arm and the teaching arm used to control it. Both arms are capable of seven degrees of motion. We aren't sure what the main arm was originally designed for, but it may have been a demonstration or prototype model. We decided that each arm would have its own control system and would communicate serially. Photo 1 shows both arms and the completed control system.

The main arm had been part of another group's project back in 1986. Although a few scattered pieces of its control system were still around, the main arm hadn't functioned in many years. We searched the Internet, but didn't have much luck finding information about either of the arms. All we could find out was that a now-defunct company called RSI had built it. We wanted to be able to control the arm in real time with the teaching arm. We also

wanted to be able to record the main arm's motions on a PC. Our primary concern was real-time control. We wanted to implement a PC interface, using either CAN or serial communication, and then write a program to record and play back the arm's movements.

Did we mention that we had only 11 weeks to complete this project? Armed with what little information we had, we got to work.

INITIAL FINDINGS

The main arm has two types of brushless DC motors. The wrist-flex,

wrist-rotate, and tool-rotate joints have smaller motors, while the waist, shoulder, and elbow joints have larger motors. Due to the age arm's age, we were unable to find any specifications for the motors. The first thing that we needed to do was determine the power requirements for both motor types. To do this, we detached one of each motor from the arm and hooked them up to a DC-bench supply. Starting from zero, we slowly raised the voltage and observed the motors' responses.

Once we were satisfied with the no-load characteristics of the motors, we

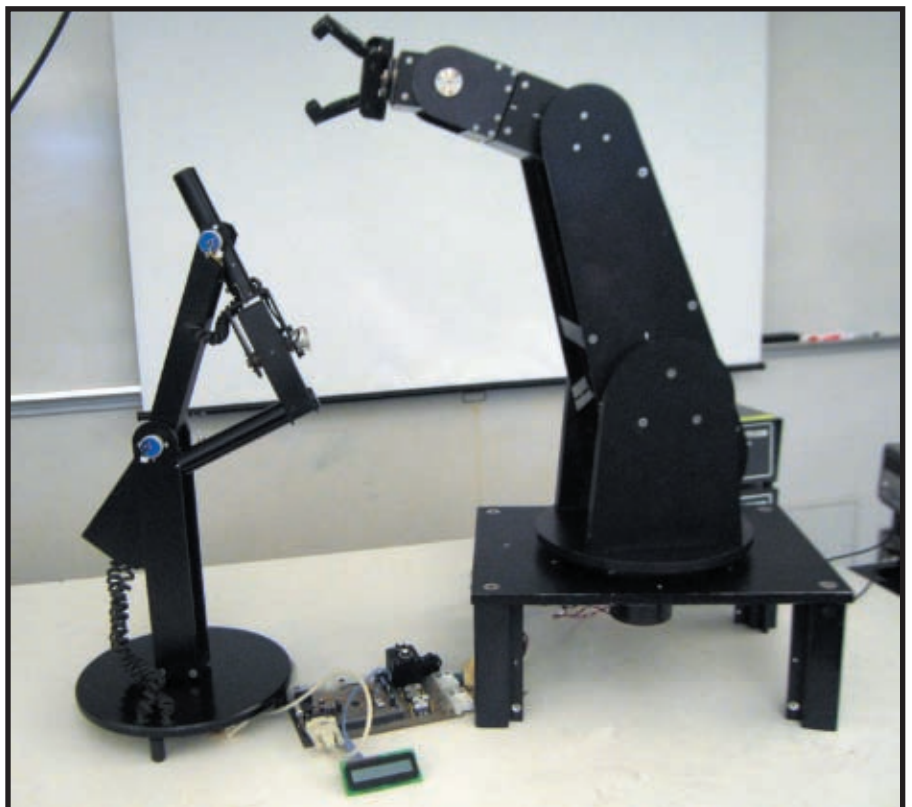


Photo 1—The finished system includes (from left to right) the teaching arm, controller boards, and the main arm. Both arms are capable of seven degrees of motion.

reattached them to the arm and repeated the experiment. Because of the arm's potential to cause damage or serious injury, we had to be very careful. We made sure that the arm was clear of other objects and that no one was in its vicinity during operation. Through trial and error, we found that the larger motors ran best at about 12 V and drew approximately 500 to 600 mA during normal operation. The smaller motors ran best at 24 V and drew 150 to 200 mA.

Next, we had to determine the layout of the original wiring.

Because nothing was color-coded or documented, we had to map out the

Microchip Technology's 16-bit DSC motor control family. We ended up

entire system. This was accomplished by performing a continuity test between all of the points on both arms. Once we knew which pins on the motors corresponded to which pins on the connectors, we labeled them. Looking back, it would have been a good idea to redo all the wiring when we started because we experienced several wiring failures.

CONTROL SYSTEM

We looked at several microcontrollers before our search led us to

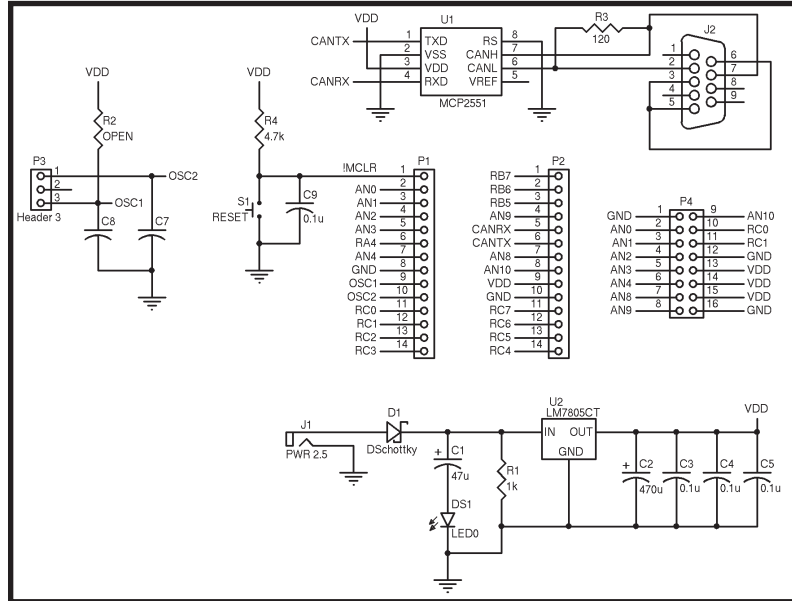


Figure 1—A PIC18F2585 attached via headers P1 and P2 reads the values from the position sensors on the teaching arm. It then transmits the values serially to the main arm.

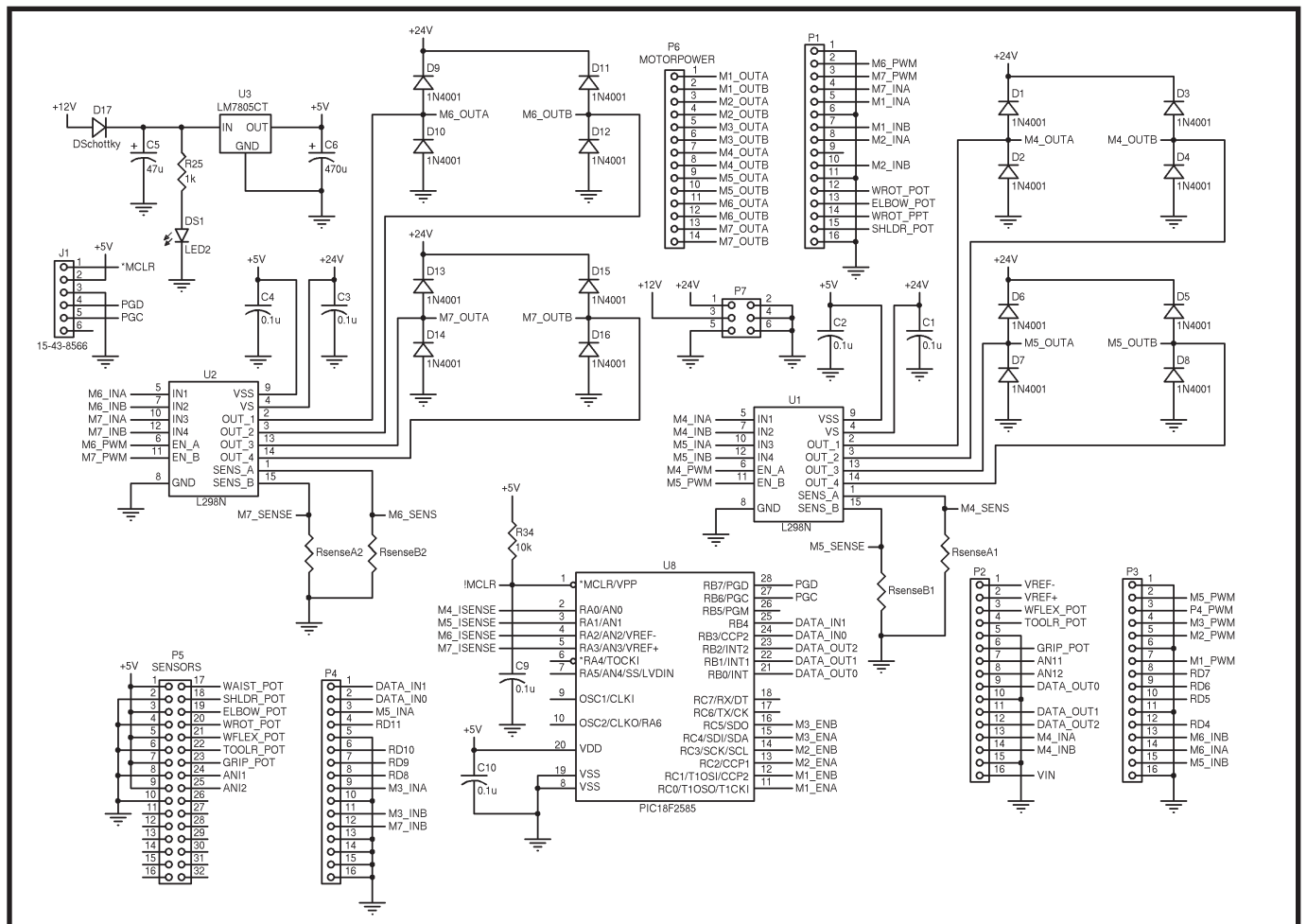


Figure 2—Take a look at the L298 motor driver. This circuit was used to control the four smaller motors. The PWM signal is applied to the ENABLE pins, and the direction is controlled using the two INPUT pins. This freed up PWM pins on the dsPIC30F6015.

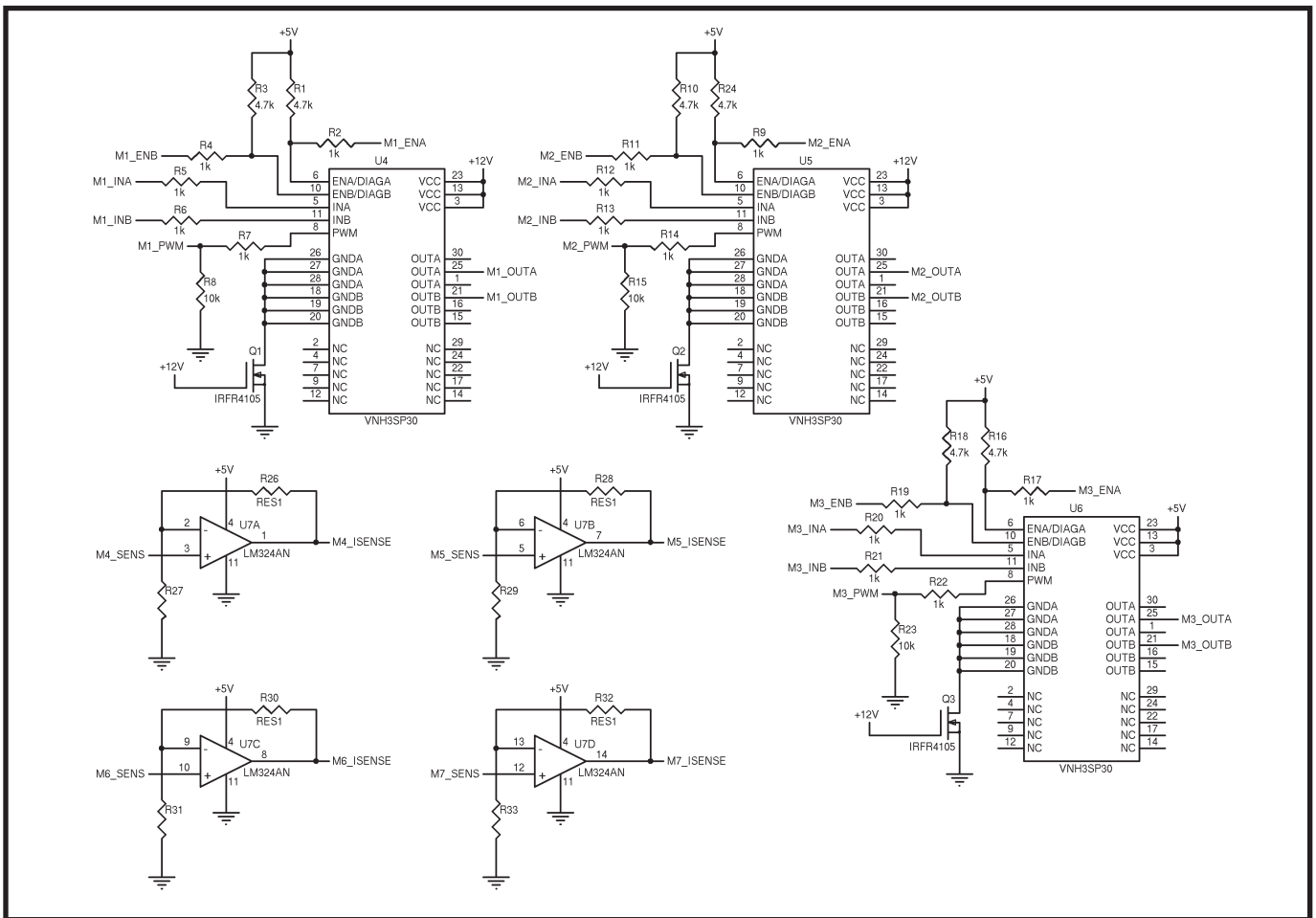


Figure 3—Check out the VN3SP30 motor driver. This circuit drives the three larger motors. For motor 1, applying a PWM signal to the M1_PWM input controls the speed. The M1_INA, M1_INB, M1_ENA, and M1_ENB lines are used for direction and enabling the motor.

choosing the dsPIC30F6015, which is a 64-pin TQFP device. Its onboard PWM modules, ample program memory, and RAM were immediately attractive. It has 16 channels of 10-bit A/D conversion, which was more than adequate for reading the position sensors on the arm. The dsPIC30F6015 also has onboard support for all of our communication requirements. We were familiar with some of Microchip's other processors, so we felt it was a solid choice.

In order to design our control system, we needed a development board for the dsPIC. (Refer to the diagram on the *Circuit Cellar* FTP site.) The budget for the project did not allow for a commercial solution, so we had to develop our own. Our requirements included in-circuit programming capabilities, RS-232 and CAN communication, and headers for an LCD and matrix keypad. We also wanted to incorporate the board in our

final design.

For initial testing purposes, we decided that the board should be able to fit on a standard breadboard using headers. We figured we could then use the same headers to interface the controller board to the motor driver board (see Figures 1, 2, and 3).

We implemented the appropriate circuitry to enable in-circuit programming, a MAX232 level converter for serial communications, and a Microchip MCP2551 CAN driver. An eight-pin header for a 4 × 4 matrix keypad and a 14-pin header for a standard Hitachi LCD completed our communication requirements. Other additions included regulator and oscillator circuits. Photo 2 shows the completed development board connected to the motor driver board.

The arm has two types of brushless DC motors. We did some tests and found that the larger motors on the waist, shoulder, and elbow joints run

best at 12 V. The wrist-flex, wrist-rotate, and tool-rotate motors have smaller motors that run best at 24 V.

We decided on a pair of STMicroelectronics L298 dual full bridge drivers for the four smaller motors, but the three larger motors posed more of a challenge. The stall current on these motors could approach 4 or 5 A, which is more than the L298 is capable of. Although we intended to implement over-current protection, we wanted to be safe, so we began searching for another solution.

The SGS Thomson Microelectronics VN3SP30 H-Bridge motor driver proved to be a good choice. Designed for automotive applications, such as power windows and seats, they can handle a massive 30 A of output current and a maximum operating voltage of 40 V.

For the position feedback potentiometers on both arms, we chose the Bourns 6539S series. Although these servo-motor potentiometers are not

2,600 NEW
BOURNS
PARTS

5,000 NEW
AMP
PARTS

2,500 NEW
MAXIM
PARTS

Wow! Jameco just added 65,000 new major-brand products!

2,000 NEW
MOLEX
PARTS

2,900 NEW
VISHAY
PARTS

19,000 NEW
**TEXAS
INSTRUMENTS**
PARTS

2,800 NEW
MICROCHIP
PARTS

The industry's fastest growing product offering!

You know that Jameco's catalog always offers over 99% *in-stock availability*—the best of any electronic components distributor...

And now, they have the fastest growing product offering in the industry!

They've just added another 65,000 *new parts* to their online catalog; and it's everything from ICs to passives, optos to interconnects, power supplies to electromechanical.

Service & Availability!

As Design Engineers know, Jameco offers great service, selection and same-day shipping!

Now you can get those same benefits for even more great brands...

6,200 NEW
FAIRCHILD
PARTS

3,000 NEW
AVX
PARTS

Check out these new and expanded lines:

Aavid Thermalloy • AlcoSwitch • AMP • Amphenol Connex • Aromat • Atmel • Augat • Avago • AVX • Bourns • Buchanan • C & K Switches • Comair Rotron • Condor Power Supplies • CTS • Cypress • Dallas Semiconductor • Elco • Fairchild • Grayhill • Intel • Intersil • Keystone • Lumex • Lumileds • Maxim • Microchip • Micron Technology • Molex • NXP/Philips • Panasonic • Power-One • Raychem • Renesas Technology • Sandisk • Siliconix • ST Micro • Texas Instruments • Toshiba • Tyco Electronics • Vishay Intertechnology • Wakefield...

Get it here. Right now:

Jameco.com/CCV

JAMECO
ELECTRONICS

Great Products.
Awesome Prices.

cheap, they provide stable and precise information about the position of the arm. The manipulator arm already had limit switches installed to prevent it from going out of range or being damaged.

When we were first presented the idea of writing firmware to control a robotic arm, we honestly thought it would be a breeze. Three years of coding embedded systems barely prepared us for the task, but we hope that our experiences can help you with your own mechatronics project.

TEACHING ARM

The teaching arm is fitted with six potentiometers. Each potentiometer is mounted to a joint on the teaching arm, which represents a joint on the manipulator arm. The single-turn servo-mount potentiometers are mounted so that the position of the wiper arm is dependent on the angle of its respective joint.

Using the built-in libraries of the Microchip C18 compiler with the ADCs of a PIC18F2585 microcontroller, we wrote code to record and transmit the voltages of the teaching arm potentiometers to the manipulator arm via a serial link. We used a running average filter to smooth the voltages read from the potentiometers before transmitting them (see Listing 1).

Every time the ADCs are read, the index of a circular buffer is incremented. This buffer is then used to calculate the average voltage. It is a simple task to derive the speed of any motion on the teaching arm. We did this by comparing the results of each A/D conversion with the previous results to produce a vector.

We chose to implement this feature in the same function as the averaging filter by simply subtracting the previous set of positions from the most recent set of positions. The resulting difference is proportional to the speed, and its sign represents the direction of the motion. In order for this difference to represent a true vector, each set of position changes must be recorded and

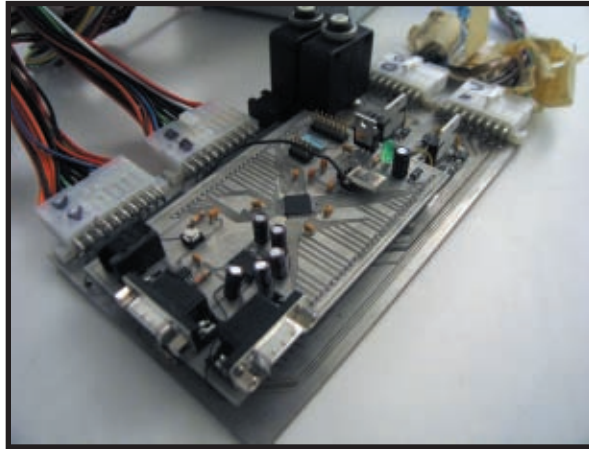


Photo 2—Check out the completed controller boards. The core of our system is a dsPIC30F6015 driving L298N full-bridge drivers and VNH3SP30 H-Bridge motor drivers. Serial and CAN communications, in-circuit programming, and current-protection circuits are also included.

calculated at discrete time intervals. The final piece of the puzzle is a timer interrupt that regulates the rate at

which data is collected, processed, and transmitted by the UART.

MANIPULATOR ARM

Once the teaching arm module was finished, we began writing code for the dsPIC30F6015 on the main processor board. We used the C30 compiler's built-in libraries to control four of the seven PWM outputs, the ADCs, and serial communications.

The manipulator arm is also equipped with six potentiometers. We chose to begin by setting up the manipulator arm's ADCs so that we could see the voltage feedback from each joint as we moved them manually. The dsPIC30F6015's ADCs were more than

Listing 1—The averaging filter is used to smooth the changing voltages of the potentiometers on the teaching arm.

```
void Update_Average_Filter(void)
{
    static int previous_result[CHANNEL_WIDTH]; // An array containing the
    // last recorded set of positions
    int results[AVG_BUF_MAX][CHANNEL_WIDTH]; // A two-dimensional array
    // containing the last recorded set of positions, the width of
    // which is decided by "AVG_BUF_MAX"
    result_difference[CHANNEL_WIDTH]; // An array containing the
    // differences between the previous elements of "results_averaged"
    // and the current elements of "results_averaged"
    results_averaged[CHANNEL_WIDTH]; // An array containing the averaged
    // results from all active analog-to-digital channels
    int temp1 = 0, temp2 = 0; // indexing variables
    for(temp1 = 0; temp1 < CHANNEL_WIDTH; temp1++) // for all channels
    {
        previous_result[temp1] = results_averaged[temp1];
        // save the last recorded averaged result for this channel
        results_averaged[temp1] = 0; // clear the channel
        for(temp2 = 0; temp2 < AVG_BUF_MAX; temp2++)
        // for all values to be averaged
        {
            results_averaged[temp1] += results[temp2][temp1];
            // Sum the values
        }
        results_averaged[temp1] = results_averaged[temp1] / AVG_BUF_MAX;
        // divide the sum by the buffer width
        result_difference[temp1] = results_averaged[temp1] -
        previous_result[temp1]; // save the difference between the current
        // average and the last recorded average
    }
}
//-----
```

ZigBee™ Certified

Wireless Modules



XBee™ and XBee-PRO™



ZigBee™ Mesh Networks

- Range up to one mile
- Worldwide approved 2.4 GHz
- Modules start under \$15 (1K quantity)

Order your
development kit
online today!
from \$129



MaxStream®
A Digi International® Brand

Toll-free 866-765-9885 • www.zigbeemodule.com/cc

adequate for this project because they are intended for DSP and can handle up to 1 million A/D conversions per second. Using the built-in libraries, we configured the A/D module so six conversions would take place each time the Convert command was issued.

To use the ADC in this configuration, we periodically called a function that would update a new set of positions to the ADC buffer. Wherever the most recent set of positions were needed in code, it could simply be read from the buffer using the ReadADC10 function. We used the LCD to display the voltages of the potentiometers as we moved them so the potentiometers could be calibrated. This is done by temporarily removing the potentiometer so the wiper arm can be moved independently from the joint, and then reattaching it once the desired voltage is aligned with the appropriate angle of the joint. Each wiper arm was positioned so the voltage would be 2.5 V at the center of its range of motion. The same steps were used on the teaching arm so that any given voltage on the teaching arm would be easily translated to a position on the manipulator arm.

To use the built-in PWM timers of the dsPIC, we modified the sample code (setDCMPWM) included with the C30 libraries. Using this configuration, the process of setting a PWM output was reduced to a function call. Each time the function is called, the appropriate PWM output pin, as specified by the arguments of the function, emits a 20-kHz PWM signal with a duty cycle that is defined by the second argument. The third argument determines whether or not the PWM output signal can be modified once set. Since it must be constantly updated, the third argument is always zero.

In order to control the direction of the motors, two other output pins are necessary. We used the differential pair of each motor driver to specify direction. The PWM signal is then applied to the enable input of the motor driver. The direction of each motor is set by clearing or setting the pins connected to the differential pair. Setting both direction pins to the same state stops energizing the coils of

Listing 2—This function compares the argument *Duty Cycle* with the global variable *Timer Period* to determine what value to write to the special function register *PR1*. This value determines when the timer interrupt will occur.

```
void SetPWM2(int Duty_Cycle)
{
    if(Duty_Cycle == 0)
    {
        LATEbits.LATE1 = 0;
        PR1 = 0;
        Timer1_Duty = 0;
    }
    else
    {
        if(Duty_Cycle > Timer_Period / 2){Duty_Cycle = Timer_Period / 2;}
        //If the desired duty cycle is greater than the maximum make it
        the maximum
        if(PR1 == 0){PR1 = Duty_Cycle * 2;}
        //If this channel of PWM is currently disabled: Enable it
        Timer1_Duty = Duty_Cycle * 2;
        //Load the PWM timer with the duty cycle
    }
}
```

the motor. Each time the PWM output is set, the direction pins are also set.

Since the dsPIC has only four built-in PWM timers, we used timer interrupts to produce the last three PWM signals needed (see Listings 2 and 3). The timer-interrupt and duty-cycle update function are shown in Listing 3. Note that it is important to temporarily disable any interrupts when doing an A/D conversion.

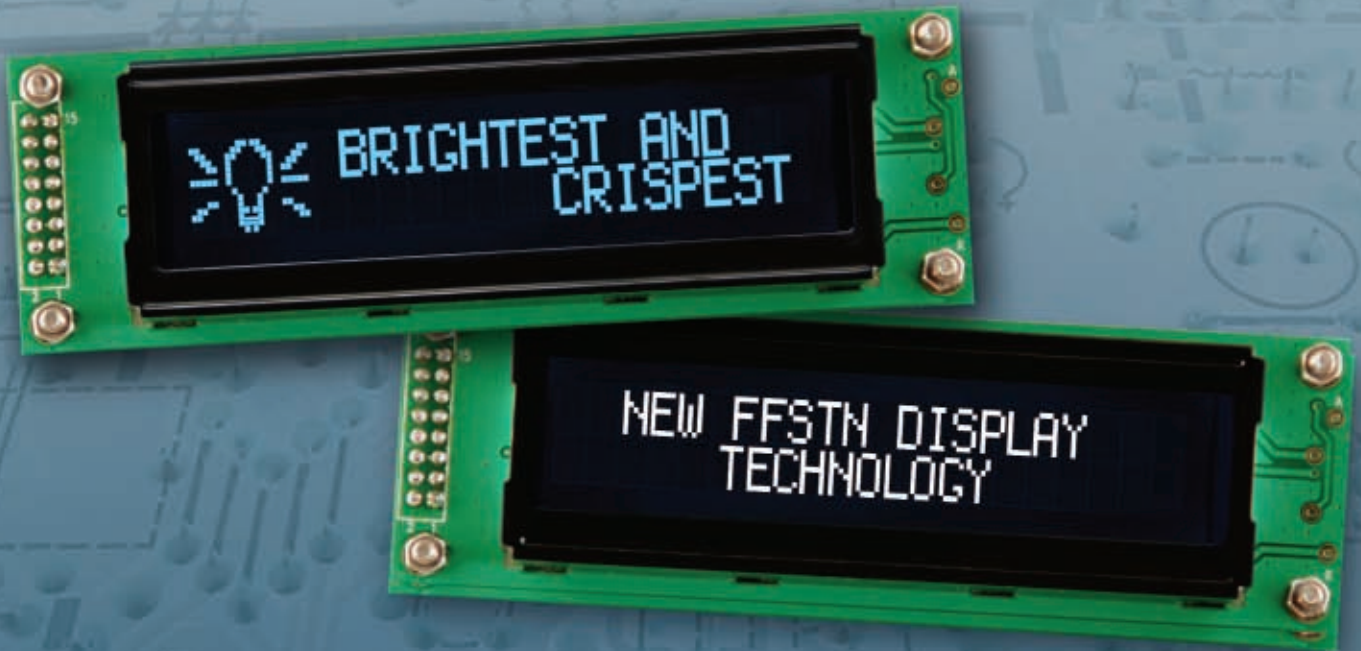
Since all seven joints have to be

controlled at the same time, the controlling functions must poll the state of the joint and immediately update the PWM output, based on both the current and desired states of the joint. The desired state of the joint is arbitrated by the teaching arm. The current state of the joint must be calculated based on the position feedback provided by the potentiometer. The speed of the motor can be set based on the vector generated by the teaching

Listing 3—This interrupt will occur at both the rising and falling edge of the PWM output. The next state of the PWM output pin is determined based on the current state of that same pin. The duty cycle will be updated only during the positive going transition of the PWM output.

```
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
    IFS0bits.T1IF = 0;           //Clear Timer1 interrupt flag
    if(PORTEbits.RE1 == 0)      //If the output is low
    {
        Timer1_Previous_Duty = Timer1_Duty; //Update the off-time of the cycle
        LATEbits.LATE1 = 1; //Set the output high and
        PR1 = Timer1_Duty; //Load the Timer with the On-Time of the
                               //selected PWM Duty Cycle
    }
    else //PWMread5 = 1 //IF the output is high
    {
        LATEbits.LATE1 = 0; //Set the output high and
        PR1 = (Timer_Period - Timer1_Previous_Duty); //Load the timer
                               //with the off-time of the selected PWM duty cycle
    }
    TMR1 = 0; //Start the count from 0.
}
```


INTRODUCING FFSTN LCD DISPLAY TECHNOLOGY



- High efficiency polarizer maximizes contrast for the brightest and crispest LCD display
- A variety of color options to choose from, such as white, blue, yellow, green and amber
- Text becomes far more visible with a darker background compared to regular LCDs
- Capable of a temperature range from -20°C to $+70^{\circ}\text{C}$

FFSTN 4x20 Displays Coming In March!

Get 5% off your next purchase online at Matrix Orbital.
Simply log on to our site and enter the code MOCC0134
in the area provided at the check-out.



**MATRIX
ORBITAL**

WWW.MATRIXORBITAL.COM

Listing 4—This function compresses the range of a 10-bit number into the upper half of its range.

```
int compressed10(int value)
{
    int compressed = value;    //Store the original value
    value = (1023 - value);    //The 10-bit negative of value
    compressed += (value >> 1); //Divide the new value by 2 and add
                                //it to the original input
    return(compressed); //the result is a number that begins at 511
                        //and increases at half the rate of the input until 1023
}
```

arm or by the angular distance the joint has to move in order to get to the desired location. Refer to the Joint-Control.doc file posted on the *Circuit Cellar* FTP site for an example of a joint control function that determines speed based on the distance the joint has to travel.

Once this function was tested and working, we modified it for the other joints. On the line labeled //Set Motor Speed, the duty cycle is set based on the variable `current_vector`. In this example, it is set to the distance of the joint on the manipulator arm from its respective joint on the teaching arm. The magnitude function ensures that the value is a positive number.

The compressed function, shown in Listing 4, is in place to ensure that the magnitude of the current vector is within the range that will move the motor. If the current vector is too small, the duty cycle will not be high enough to overcome the friction of the motor. The compression function adds half the inverse of its argument to its return value, so that any 10-bit value ranging from 0 to 1,023 will range from 511 to 1,023. Adding 25 to this ensures that all values will be able to move the motor in spite of its specific friction coefficient.

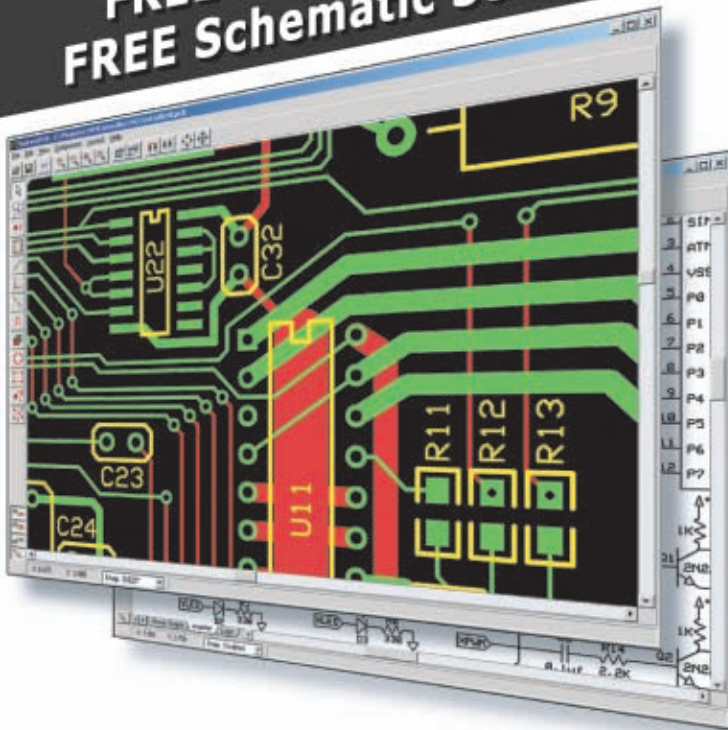
POTENTIAL IMPROVEMENTS

With the deadline looming, we were making significant progress. Joint by joint, we were taking control! The Electronics and Computer Engineering Technology program at Camosun College had prepared us well. The arm followed the motion of the teaching arm, and it seemed to take on a life of its own!

We overcame many challenges, but unfortunately, the system was not perfect. We encountered several obstacles, the first of which was the physical condition of the arms. There were problems with the original wiring of both arms, so we spent a significant amount of time trying to solve them. Then there were mechanical issues to deal with. More than one joint suffered from looseness, which was surely a sign of the arm's age. There was also the problem of getting reliable data from the position sensors. Because the potentiometers relied on a

For 3
\$51 PCBs

FREE Layout Software!
FREE Schematic Software!



- 01 DOWNLOAD our free CAD software
- 02 DESIGN your two or four layer PC board
- 03 SEND us your design with just a click
- 04 RECEIVE top quality boards in just days

expresspcb.com

friction fit, we found that they tended to drift over time. It was very difficult to maintain accuracy when the data received from the pots varied.

Looking back, we would recommend implementing a high-resolution rotary encoder for the position sensors. You could also improve the design by devising a means to securely attach the shaft of the position sensor to the mechanical portion of the arm. The tight time constraints for the project did not allow us to implement all of the features that we had planned. We had hoped to record and play back the movements of the arm, but we were unable to implement the PC interface.

ARM CONTROL

The 11 weeks flew by. After a lot of hard work, frustration, and late nights, we had a system that worked. We knew that this project would be challenging, and we knew that we would be testing everything that we had learned, but nothing beats hands-on experience. Although there is still room for improvement, we are pleased with the results. We learned a lot, and hopefully this article will help you with your own robotics project. ☺

Mike Hall (ekimllah1@shaw.ca) earned a diploma in Computer Engineering Technology at Camosun College. He works as a sound technician and hopes to find employment with an audio equipment manufacturer.

Aaron Patten (aarotech@gmail.com) graduated from Camosun College with a diploma in Computer Engineering Technology and is currently working toward a degree in Computer Engineering at the University of Victoria.

Erin Simpson (erin_simpson44@hotmail.com) holds a diploma in Electronics Engineering Technology from Camosun College. She is a devoted kick boxer and is pursuing her electronics career with the Canadian Armed Forces.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

RESOURCES

Microchip Technology, Inc., "dsPIC30F6010A/6015 Data Sheet," DS70150A, 2005, ww1.microchip.com/downloads/en/DeviceDoc/70150A.pdf.

———, "16-Bit Language Tools Libraries," DS51456C, 2005, ww1.microchip.com/downloads/en/DeviceDoc/16bit_Language_Tool_Libraries_51456c.pdf.

———, "dsPIC30F Family Reference Manual," DS70046E, 2006, ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf.


SOURCES

6539S Precision servo-mount potentiometer
Bourns, Inc.
www.bourns.com

PIC18F2585 Microcontroller
Microchip Technology, Inc.
www.microchip.com


L298 Motor driver and VNH3SP30 motor driver
STMicroelectronics
www.st.com

Add a color touch interface to your embedded product!

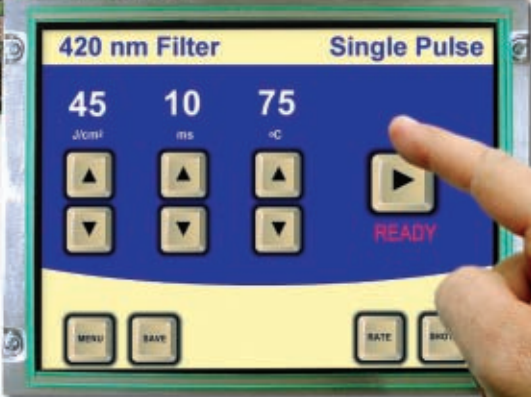


your microcontroller

Serial Commands



our SLCD5



8.4" TFT VGA

Yes, it's really this simple...


- ✓ Works with any microcontroller
- ✓ Up and running in hours
- ✓ SD card slot for images
- ✓ Easy, fast, flexible

Visit us at **BOOTH 847**

Embedded Systems Conference


April 3 - 5, 2007 San Jose

▶ Complete Kits Available
▶ Low Cost
▶ In Stock



842 Boggs Avenue ■ Fremont, CA 94539

www.ReachTech.com
5 0 3 . 6 7 5 . 6 4 6 4



Animatronic System Control

Barry and Eugene used their knowledge of robotics and a lot of creativity to design an H8/3687-based control system for a robotic monster that appeared in a horror movie. The system enabled them to control the monster with potentiometers, a touch screen, and buttons.

In 2003, Stephen Zimmer of Shadows Light Productions contacted me (Barry Stout) about being part of a huge effort to make a horror movie on a shoestring budget. He said the film would require hundreds of special-effects shots and that he needed some assistance with the development of props. At that point in my career, I had been designing motion control systems for robots, printers, and RC cars for several years. Because my other passion is film, I bit at the opportunity. Fortunately, Stephen had already assembled a team of designers who were experienced in makeup, prosthetics, mold making, and prop mechanics.

I was interested in contributing as much as possible to the project, so I brought my business partner and long-time friend Eugene Zeldin into the mix. Eugene has a lot of experience with microcontroller coding and circuit lay-

out because he has been in the industry for 10 years. Most of my experience has been with mechanical/electrical integration and controller-algorithm design, so our skills complement each other well.

In this article, we'll describe the Renesas Technology H8/3687-based system that we developed to control an animatronics mask for the movie (see Photo 1). The system features an auxiliary circuit board, a power supply, and some servos. As you can see in Figure 1, its potentiometers, touch screen, and push buttons enable us to control the mask. (Additional diagrams are posted on the *Circuit Cellar* FTP site.) This design rivals the animatronics controllers found on the sets of many big-budget feature films.

BUILDING A MONSTER

During the first few months of the project, many of the crew's dreams van-

ished because the movie's budget and timetable forced us to pass on several of the effects that we had hoped to use. For instance, we had originally planned to use flaming swords that ran on butane gas, but the budget forced us to use a computer-generated effect instead.

But not all was lost. For the movie's climactic scene, a 12' monster attacks the leading actor. To make this scene work, the actor inside the monster had to walk on stilts, while a fully animated head was mounted on top. To create this effect, we developed a body cast that eventually became a large Fiberglass mold. Between the egg smell of the foam latex and the toxic odor of the Fiberglass, the makeshift special-effects studio was never the same again. We are just grateful that we retained enough brain cells to write this article.

While the mask effects team was

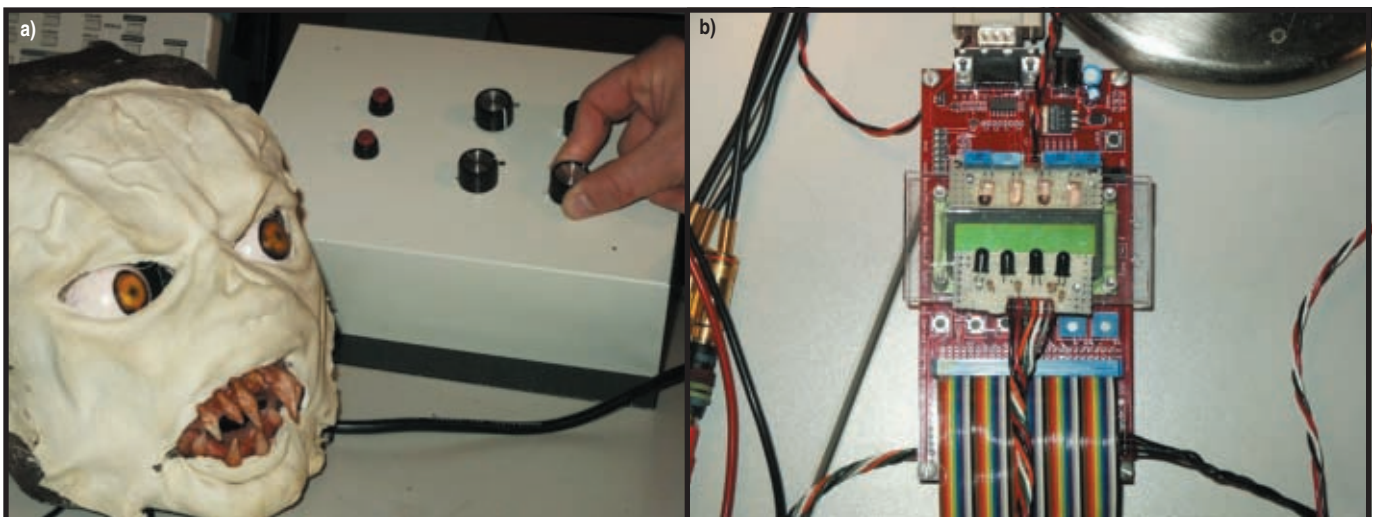


Photo 1a—This is an early prototype of the main controller board. The monster consists of an unpainted foam mask glued over a molded Fiberglass frame with servos attached to hinges and slides in the face. **b**—This is a prototype of the touchscreen's conversion electronics mounted on the evaluation board's LCD. The final design covers the light emitters and receivers, while shielding the crosstalk between the LEDs.

building a 15' oven to cook the full body suit, we rushed to complete the monster's animatronics controller (see Photo 1b). To fully animate the mask, the eyes, mouth, eyebrows, cheeks, and nose had to be mechanically actuated. (Refer to the Animatronics sidebar.)

Like all good engineers, we laid out the necessary degrees of freedom for each joint and formulated the required electronics. Besides the system's cost, there were several other challenges to address: the monster had to be lightweight, battery-powered, wireless, easy-to-use, and highly adaptable. Other fun features such as a low-cost touch panel, kinetic-metric inputs, and module coding were added to the plan, but they were not our primary focus.

For animatronics novices, kinetic-metric inputs means that a person's movements can directly control the robot's movements. In such a system, a sensor that monitors the arm movement of a puppeteer will send a signal to the main processing unit. From there, the controller moves the robot's arm in the same fashion. This is similar to the way older movies were made (e.g., *Star Wars*). The puppeteer would attach cables to a monster's joint and then pull them. With modern-day animatronics systems, all of the control is wireless. Controlling a robot is now like controlling an RC car (but without your brother trying to pull the

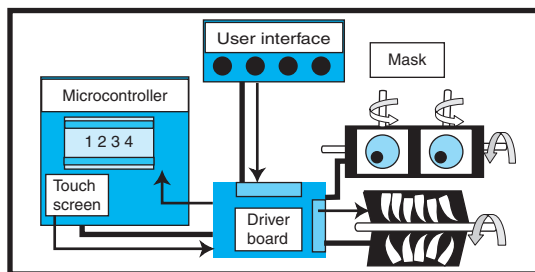


Figure 1—The user interface sends commands to the microcontroller, which in turn sends signals wirelessly to the motor driver board that controls the servos.

transmitter out of your hands).

We decided to create a driver board that would control the movement of the array of actuators and an external control box that would act as a user interface to the driver board. As a bonus, we wanted the driver board to have a built-in interface for host PC communication or preprogrammed scripts.

In the beginning, we had to decide what we wanted to build. Being the workhorse of the project, the motor controller became our highest priority. We then had to consider the user interface, options and features, power requirements, the microprocessor, and software.

Once everything was laid out, we were ready to start the architecture and make high-level design decisions. During this time, we chose the motors and controller circuitry, and we sketched the basic design on a napkin during a healthy lunch consisting of a lot of fried chicken.

For this project, we used a Renesas H8/3687 kit and a C development

platform (HEW 2.2), which included the compiler to program the microcontroller. In addition, we gathered up all the loose components such as breadboards, oscilloscopes, computers, ohm meters, and a couple of power supplies. After a quick blessing of the equipment, we were ready to go.

We had to burn the midnight oil for a few days, but it wasn't long before pieces of hardware and software were starting to come together. We had a rudimentary PWM motor controller implemented in record time.

CIRCUIT CARD

After hours of fine-tuning the hardware and software on those dreaded breadboards, we were ready to build a pilot board. Equipped with P-CAD, we finalized the schematics and created a net list for the layout. In order to create that actual circuit layout, we placed the components in their appropriate locations on the circuit board, routed a couple of the main power traces and control signals, and then let the auto router go wild and crazy. After a couple of attempts, we were satisfied with the auto router's output. How could a computer program be wrong? At this point, we cleaned up a few of the routes and finalized the circuit board.

The next step was to generate all the masks, copper pours, silkscreen, and various layers to bundle it in the

Animatronics

Most moviegoers love special effects. This is also one of the reasons why American movie making has dominated the world with high-budget effects that the independent filmmaker cannot afford.

In 1963, Walt Disney spent hundred of thousands of dollars to make controllers for his Tiki Bird and Abraham Lincoln exhibit, which consisted of analog tape machines reading silver traces to activate pneumatic actuators. This marked the beginning of the field of animatronics.

Much has changed about the way in which people make movies, but one thing remains the same: high-quality special effects usually require a big budget. Fortunately, with the advent of microcontrollers, the cost of mechanisms to control props has drastically decreased. The purpose of this

project was to quickly build an affordable controller that could remotely operate an animatronic mask.

Even though the history of animatronics started with a preprogrammed mechanism, modern animatronics for movies resembles puppeteering, where the actions of a person are replicated in the monster. There are several reasons for this approach, one of which is that programmed actions may not provide the flexibility to make last-minute changes that the movie director might need to make. In addition, when an actor can mentally connect with the creature that will appear on screen, it makes for a more believable performance. In comparison to older methods of special effects, our controller needed to be able to relay information from the user to the creature electronically instead of with strings.

proper format and send it off to a board house. Now, we had about a week to work on the software and new features while the boards were made.

When the boards were finished Eugene screamed, "The boards have been delivered and they look awesome!" Remember, it's the little things in life. Of course, now came another time-consuming task. We had to stuff and solder two prototype boards and make all of the cabling. Wow, we had forgotten how long it

takes to make at least 10 custom cables per board. After another couple of days and a few burns from the soldering iron, the boards and cables were ready.

Now came the moment of truth. Would the boards work? Would all of our effort pay off? Would the Cubs win a World Series again? The easy answer to all those questions was "no." Fortunately, a heat gun fixed a few solder balls trapped under the connectors. We then used a combination of Freon air and a Polar ToneOhm 550

meter to find the mysterious short, so the first two questions were answered. No hope for the last one.

MCU & FIRMWARE

For a large project, we would normally spec out the proper micro-processor that contained all the required functionality at the lowest cost. However, for this stunt, considering the time schedule and the low cost of the evaluation boards (\$50), we just designed the main motor controller card to socket the evaluation board. Later designs incorporated just the microcontroller, but more about that later.

As a good design note, always use optical isolators to buffer any signals going to motor drivers. When a driver melts, it commonly connects your voltage rails together and burns up your most expensive component in the circuit (it just knows which one). For bidirectional lines going to the microcontroller, you are probably fine. But, don't forget that optical isolators need a lot of juice, so you can't drive them directly from a microcontroller.

After we picked the H8/3687, it quickly became apparent that we would need to scale back the design for this 8-bit microcontroller. This was not a 32-bit power PC processor clocking along at 1 GHz, which we had been accustomed to. We knew this was going to be a basic super-loop program, servicing the major firmware features and hardware interrupts. No real-time operating system, loads of RAM, or NAND flash memory for code execution and storage. So, we wrote several modules and ISR routines to service the serial ports, motor controllers, LCD, and user interface. We wanted the code to run efficiently and use as few system resources as possible.

USER INTERFACE

An extremely important part of any project is the user interface. The end user rarely cares about the circuit board, processor, and the 1,000 hours of blood-and-guts work put in by the engineering staff (and, in most cases, neither does the project manager). The end user is more interested in ease of

Fighting against your PCB-Design Software?

Here's something that will spare your time and your budget!

Boards designed under EAGLE are found in patient monitoring equipment, chip cards, electric razors, hearing aids, automobiles and industrial controllers. They are as small as a thumbnail or as large as a PC motherboard. They are developed in one-man businesses or in large industrial companies. EAGLE is being used in many of the top companies. The crucial reason for selecting EAGLE is not usually the very favorable price, but rather the ease of use. On top of that comes 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!

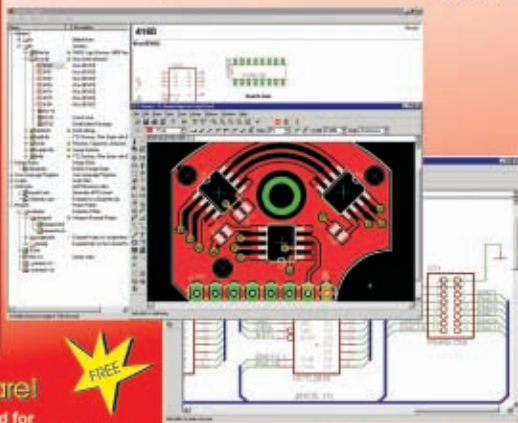


EAGLE 4.1

Schematic Capture • Board Layout
Autorouter

for Windows®
Linux®
Mac®

Now
available
for Mac!



Version 4.1 Highlights

- ▶ Powerful library management: e.g. move devices between libraries, base library for packages, generate package variants from other libraries.
- ▶ Dynamic ratsnest during routing process.
- ▶ Copy function in schematic.
- ▶ Rotate components in 0.1-degree steps.
- ▶ Blind & buried vias and pads with off-center drill.
- ▶ User-defined background color.
- ▶ Miter function for (rounded) tracks.
- ▶ Smash for groups.
- ▶ Measure distances between arbitrary points.
- ▶ Choose alternative raster on-the-fly with Alt-key.

EAGLE 4.1 Light is Freeware!

You can use EAGLE Light for testing and for non-commercial applications without charge. The Freeware Version is restricted to boards up to half Eurocard format, with a maximum of two signal layers and one schematic sheet. All other features correspond to those of the Professional Version. Download it from our Internet Site or order our free CD.

If you decide in favor of the Commercial Light Version, you also get the reference manual and a license for commercial applications. The Standard Version is suitable for boards in Eurocard format with up to 4 signal layers (max. 99 schematic sheets). The Professional Version has no such limitations.

<http://www.CadSoftUSA.com>

800-858-8355

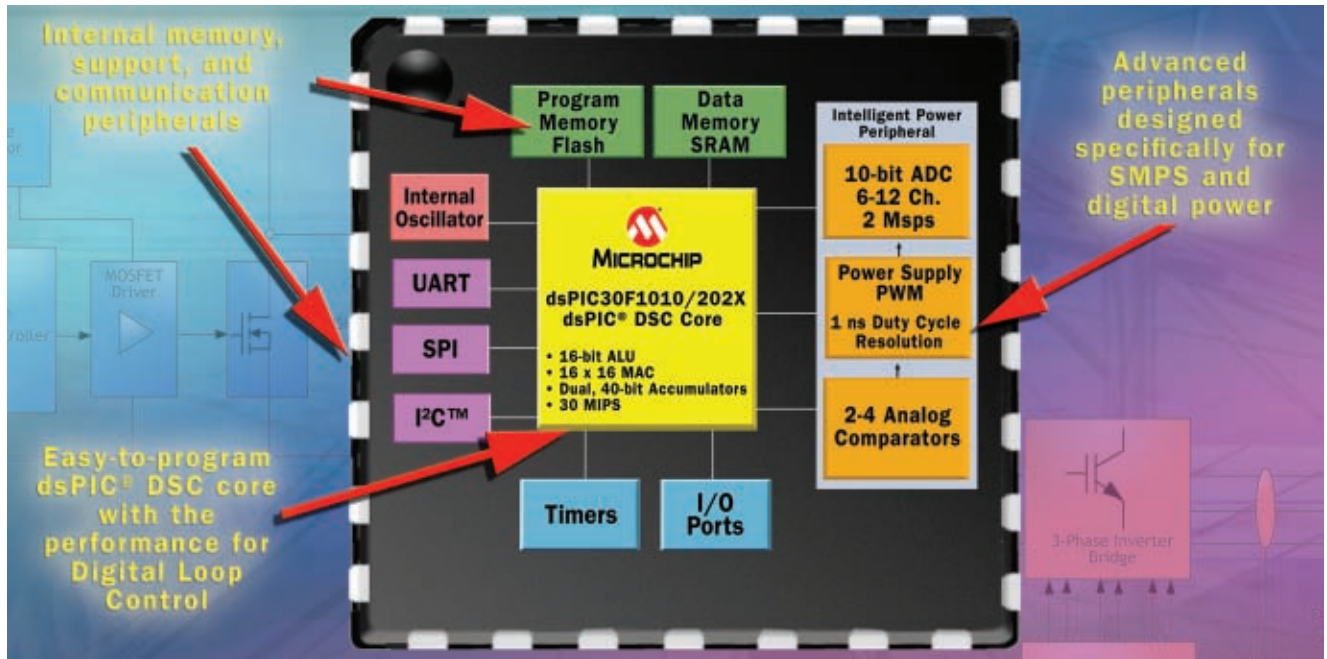
| Prices | Light | Standard | Professional |
|---------------------------------|-------|----------|--------------|
| Layout | | 199\$ | 399\$ |
| Layout + Schematic | | 398\$ | 798\$ |
| Layout + Autorouter | | 398\$ | 798\$ |
| Layout + Schematic + Autorouter | 49\$ | 597\$ | 1197\$ |

Pay the difference for Upgrades

CadSoft Computer, Inc., 801 S. Federal Highway, Delray Beach, FL 33483
Hotline (561) 274-8355, Fax (561) 274-8218, E-Mail : info@cadsoftusa.com

EAGLE is a registered trademark of CadSoft Computer, Inc. in the United States and other countries. Windows is a registered trademark of Microsoft Corporation. Linux is a registered trademark of Linus Torvalds. Mac is a registered trademark of Apple Computer, Inc.

Turn Digital Power into Real Power



Looking to bring the benefits of digital power conversion – including higher power density, lower cost features and accelerated innovation – to your product?

Microchip's dsPIC® Digital Signal Controllers (DSCs) for Switch Mode Power Supplies (SMPS) and digital power conversion are designed for applications in AC/DC power supplies, power factor correction, DC/DC converters, uninterruptible power supplies, power inverters, digital lighting and advanced battery chargers.

GOING DIGITAL THE EASY WAY!

1. Educate yourself

Learn the technical aspects of digital power conversion and SMPS design with hands-on training and **FREE** web seminars from Microchip.

2. Accelerate your development



Speed up your design time with tools targeting digital power such as the dsPICDEM™ Buck Development Board.

3. Win stuff in our Digital Power Contest

Visit our web site and get coupons for discounted development tools when you **REGISTER TO WIN** a dsPIC DSC SMPS Development System. We are also giving away an Apple® iPod® mobile digital media player to one lucky winner! Visit our web site for complete contest information.

Purchase and program your 16-bit dsPIC® DSCs and related development tools at...

microchip
DIRECT
www.microchipdirect.com

| Device | Pins | Flash (KB) | 10-bit, 2 MSPS (CH) | High Speed PWM (CH) | Analog Comparators |
|--------------|------|------------|---------------------|---------------------|--------------------|
| dsPIC30F1010 | 28 | 6 | 6 | 4 | 2 |
| dsPIC30F2020 | 28 | 12 | 8 | 8 | 4 |
| dsPIC30F2023 | 44 | 12 | 12 | 8 | 4 |

Visit our web site for more information about additional 16-bit devices supporting power conversion!



MICROCHIP

www.microchip.com/SMPS

use, whether the system can accomplish the required tasks, and its cost.

We kept the first prototype clean and simple. We populated a metal case with about a dozen potentiometers in a logical layout. Most animatronics users prefer the old-fashioned finger-numbing knobs, so they can “feel” the monster. So, we thought the potentiometers would allow the user to dial in the settings of the mask’s features, including the desired position, trim values, and speed limits. Potentiometers were chosen to be sturdy and heavy duty so they could stand up to tough use on a movie production set.

It quickly became apparent that having the controller box physically connected to the main motor controller board would limit the movements of the animatronics creatures. So, we decided to explore a wireless option. By this point, the end date



Photo 2—This is an example of screen output for the servo trim settings. In this case, the user is setting the servo’s maximum return angle.

was fast approaching and our better halves were getting impatient with us because of all the time we were spending on the project. Still, we had gone too far to turn back. We had to go wireless.

We used an off-the-shelf Reynolds Electronics RS-232 transmitter receiver (TXLC-434/RXLC-434). We pro-

grammed one microcontroller to set up and run the user interface and send the control signals. The second microcontroller receives the signal and controls the motors. Unfortunately, this required another version of code. Of course, this was kind of time consuming. We had to develop two boards (master and slave) that used separate microcontrollers and communicated wirelessly.

Now came the fun part. Actually, blowing things up is usually the most fun. In this case, creating a user interface by displaying char-

acters to the low-cost 2 × 16 LCD screen with buttons to select the options appealed to our creative senses. Remember: it’s the small things that count. This also allowed for an easy-to-implement debugging and set-up platform. Again, keeping things simple is the key. You can squeeze a lot of functions into menus, but then you must be willing to navigate various levels. We didn’t want that. Furthermore, we provided tools that used the PC’s serial port for additional enhancements, such as data logging, batch files, real-time feedback, and timed executions.

Transversing the menu via the buttons was a bit of a pain, so we decided to upgrade to a touchscreen. We felt the user would appreciate this feature. However, we quickly realized the cost for such an interface would be prohibitive; plus we already had mounted the LCD screen.

After hours of scribbling, arguing, and eating a lot of junk food, we settled on a new direction. We decided to transform our boring LCD screen into an amazing touchscreen through optical tripping sensors. Unfortunately, this used the last of the microcontroller’s unused inputs and outputs. To achieve this goal, matching LEDs (Fairchild QED123) and phototransistors (Fairchild QSD123) were placed around the LCD. Now, when you press on the Plexiglas shield protecting the LCD, your finger trips one of the photodetectors, mimicking a but-

We Listen. Think. And Create.

Distributed
I/O

Digital
I/O

Serial
I/O

Industrial
Computing

HMI

SeaLINK Ethernet serial servers are the fastest, most reliable way to connect serial devices to your network.

SeaLINK Ethernet Serial Servers Offer:

- 1, 2, 4, 8, and 16-Port Models
- RS-232, RS-422, RS-485, and Optically Isolated Versions
- Included Software Enables Virtual COM Port Operation
- Easy Installation and Configuration
- DIN Rail or Table Mount Design
- Extended Temperature Option Available

FOCUS
On Success
Call Today!

SEALEVEL
sealevel.com > sales@sealevel.com > 864.843.4343

ton being pressed (see Photo 2).

Figure 2 shows the hardware layout for the touchscreen with an example of a menu choice on the LCD. Finally, the phototransistor feeds its signal to the buttons on the microcontroller's board. This allows you to push the buttons or the screen. Pure genius! We know.

CREATING MOTION

The actuator controllers form the true heart of the project. While there are many solutions on the market for moving facial joints, such as eyebrows and jawbones, most don't have the power and speed for lifelike movement. Take for instance the lower jaw of the monster. To make it lifelike, the jaw needs to snap closed in certain cases and appear as if it is talking or growling in others. An off-the-shelf servo could satisfy one requirement but not the other. Although servos can be used for other moving parts, such as the eyebrows, the mouth, eyes, and horns need more flexibility in control. This was a good reason to create this large project and ignore our significant others.

We first counted the degrees of freedom (DOF) of each joint and quantified the needed motions. In short, the robot needed eight servos, four high-speed motion feedback controllers, and one solenoid controller. Although hydraulics and pneumatics would have been great solutions, the actor didn't want a large gas or oil tank attached to his body. (Ah, actors. Typical.) We also knew that actors know little about electricity, so we kept it that way and moved on.

For anyone familiar with the film industry, flexibility is a key to success. We had to be able to actuate numerous facial joints and not shock the actor. Fortunately, the needed circuitry and software fell into place.

Note that the producer usually tries to pick someone from the SFX staff to wear the suit. Just look polite and run like hell. The shoots are long, the suits are hot, and you know about electricity. Enough said.

CONTROLLERS

Basically, the controllers fell into two buckets: open and closed loop. For those non-controller junkies out there,

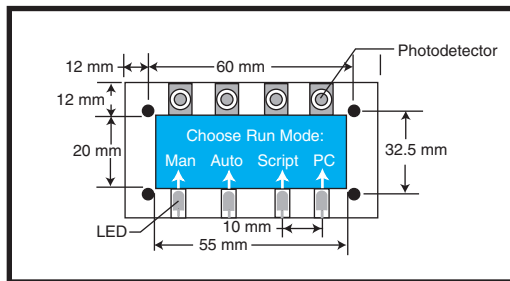


Figure 2—Your finger will trip the path from the LED to the photodetector, simulating a touch on the screen.

open-loop controllers are for stepper motors, solenoid switches, and off-the-shelf servos. Consequently, if you take apart a servo, you will find a hidden closed-loop controller, which translates a reference signal to the desired rotation. Therefore, in order to control custom components like our high-torque motors, the main board would be the closed-loop controller for the high-speed or high-strength actuators.

On the simple side, the open-loop controllers consisted of buffered digital signals from the microcontroller. Straightforward PWM signals sent the needed reference signals to the servos.

In addition, optional lines were set aside for stepper motor control.

The headaches and addiction to raw cookie dough began when we started designing the closed-loop controllers. As you know, the primary key to any closed-loop system is plenty of caffeine, while the other key aspect is clean feedback. If you've ever used a microphone on stage, then you're probably familiar with the term

"feedback." In our monster's case, feedback is good. Fire isn't. For the sake of simplicity, feedback came from an inexpensive Agilent (Avago Technologies) quadrature optical encoder (150 LPI). The HEDS-9700 provided excellent reliability, low noise, and good repeatability over different environmental conditions.

MOTOR DRIVER

For driving the motors and solenoids, we used STMicroelectronics's L6206 dual driver ICs, which can drive 1-A motors at 30 V. These provided an excellent response time while keeping

Going Wireless is Easy!
Wireless Mesh Networking

NEW! Telegesis USB ZigBee™ Stick

Bluetooth with Dip-Switch options

Multi-Channel 900 MHz Wireless RF Modules

LEMOS INTERNATIONAL

Radiometrix TX3A-914-64
Radiometrix RX3A-914-64

Call Toll Free 1-866-345-3667 or email at sales@lemosint.com

www.lemosint.com

thermal requirements at a minimum.

To efficiently use the timers to drive the motors, we had two timebases. The servos needed a 50-Hz chopping frequency with a duty cycle of 5 to 10%, which correlates to 0 to 270° of travel on most standard off-the-shelf timebases. On the other hand, the DC motors controlling the eyes and horns were set to 20 kHz, mostly to avoid the audible range of the human ear. In this case, the limits on the duty cycle came from thermal constraints.

The H8/3687 microcontroller made it easy to hang several timer trigger points off one base time, saving on the usage of independent timers. In the case of the servos, the pulses can be triggered sequentially to minimize power spikes, because the individual duty cycles are low.

CREATING CONTROLLER CODE

With the hardware in place, the true fun begins. (But it still isn't as much fun as blowing stuff up.) When people ask us what is the field of controls in terms of engineering, we tell them it's the art of moving actuators (motors, solenoids, etc.). For us, it's also the art of writing and debugging the code to move the motors, in addition to the electrical and mechanical designs for the system—the joys of a two-person team.

Not straying too far from the industry, the main controller algorithm was PID based. We simulated the controller in a real-time environment and applied genetic algorithms to tune the system.

In a perfect world, the controller algorithms would miraculously appear in the code bug free. What happens in the real world is that the compiled code finally works after days of debugging. Then comes the big question. Is it really working correctly? In other words, just because the mouth moves, it doesn't mean the controller works correctly.

The first step is to verify that the controller algorithm is in a "perfect" environment, where you are not limited by integer math, bandwidth limitations, and I/O problems. Luckily, we already had a dSpace 1103 board, which allowed MATLAB Simulink models to be compiled and run in real time. We could have used the fine suite of real-time products from

National Instruments, but the dSpace card was already sitting in the better computer. Hooking up the actual motors and optical encoders to the dSpace board gave us a good feel for the system and provided us with reference data. Taking the simulation offline, we could compare the output from the perfect controller to the output from the microcontroller's emulator to verify the algorithm implementation.

TUNING THE CONTROLLER

After verifying that the integer math came close enough to the floating-point dSpace controller for government work, the next step was tuning the real system. Most practical engineers would just hook everything up and tune away by hand. While some of that always happens in the end, we took the challenge of developing an automated tuning system. Why? Because we could.

The first pass simply involved the microcontroller logging in the encoder position along with the motor command to a host computer, which processed the data through the serial port. However, this proved only that the serial port was hogging all of the microcontroller's possible bandwidth when active. Pass two involved making the serial-communication code more efficient and passing less data.

IMPLEMENTING THE CONTROLLER

Hooking up the optical encoder, motor PWMs, and the microcontroller to a National Instruments 6602 (multiple high-speed timers) card through LabView provided more bang for the buck. (The LabView solution was quite a bit more expensive than the previous solution.) A genetic algorithm running on the PC tuned the controller through trial and error by taking advantage of the serial port from a National Instruments DIO card (high-speed digital interface). The thought was to eventually put the genetic tuning algorithm in the microcontroller, so it could tune itself. Plus, it just sounded cool. Even though LabView did a fine job of finding optimal parameters for the very nonlinear system by applying advanced algorithms found in my thesis on robotic control (so you know they are good), it took

about 20 min. of the motors shaking the monster to find the sweet spot.

The generic algorithm technique was finally dropped in the final hour for a more straightforward routine that runs the motor at fixed duty cycles and measures the response time of the system. Furthermore, our procedure sets the Kp value, assuming a loop time of 1 ms to provide 50% of the maximum performance of the system. It uses half the value for Ki that causes instability. It's not pretty, but neither is a robot shaking its hinges loose for 20 min.

Because the controller uses velocity as feedback (differentiating the position), Kd would just be multiplying noise so it was dropped out. The actual algorithm used feed-forward terms, split gains, and many signal filters—the secret is in the sauce. For special effects work, this is pretty high-tech, so we patted ourselves on the backs and moved on.

IT'S ALIVE

Putting together a project like this is challenging. You need a good plan and the proper resources to get the job done.

In the end, the electronics finally came together in perfect harmony and everything worked as we had planned. The motors hummed away while the solenoids opened and closed. We controlled the system from our wireless controller panel. Unfortunately, the producers of the movie shrank the budget even further, which shrunk the monster from 12' to 6'. The monster's head became the actor's head. Thus, we didn't need to control its eyes and mouth. We had two days to activate the eyebrows and horns using a fraction of the space originally planned. We pulled it off using RC parts, but our grand controller system was shelved until the next movie. That's show business for you.

During the last few years, we've updated our designs and used them in other projects. In order to control multiple robots, we came up with a modular design. Each motor board drives fewer motors, but the system can adapt to more situations. The microcontrollers can reflash certain areas of their own memory to remember settings, even if the power cycles.

PIC24 MCUs

Highly integrated family of 16-bit PIC® MCUs designed to meet the demanding needs of real-time control. Common attributes among all PIC24 MCUs include reliable and flexible flash memory, pinout, software and peripheral compatibility, as well as common development tools.



 MICROCHIP mouser.com/microchip/a

B6TS – Capacitive Touch Sensing ICs

Capacitive touch sensing IC developed to be highly tolerant of its working environment with adaptive features such as self-teaching, auto threshold, and intelligent filtering to meet the demands of most applications today.



 OMRON
ELECTRONIC COMPONENTS mouser.com/omron/a

Easy Ordering In Nanoseconds



NEWEST Products
NEWEST Technologies
The ONLY NEW Catalog Every 90 Days

For over 40 years engineers have relied on Mouser as their source for electronic components. And now it's easier than ever to order from our catalog via phone or online — in nanoseconds.

That's why we deliver the **ONLY** 1,800+ page catalog of the newest product information **4 times a year**. And with daily updates to over 740,000 products on-line, you can depend on Mouser to save you critical time to market!

Experience Mouser's time-to-market advantage! Our vast selection of the NEWEST products, NEWEST technologies, **new catalog every 90 days**, no minimums, and same-day shipping on most orders, gets you to market faster. We make it easy to do business with Mouser!

mouser.com (800) 346-6873



a tti company

*The Newest Products
For Your Newest Designs*

Serial ATA Cable Assemblies

Cable assemblies with positive locking latches that ensure cable connection. Features high-speed data transfer at 300Mbps, can be hot swapped without shutting down or restoring system, built-in RAID support, 26AWG wire size, 1.5A max. current, and 40V max. voltage.



 molex
one company > a world of innovation mouser.com/molex/a

Non-Isolated SMT and SIP DC-DC Converters

Open-frame construction and small footprint enable designers to develop cost-and space-efficient solutions. Features programmable output voltage, remote on/off, output overcurrent protection, and a temperature range of -40°C to +85°C.



 tyco
Electronics
Authorized Distributor mouser.com/tycopowersystems/a

RGB LED Light Engine

Light engines with independent color control for dynamic or preset pre-set color display. Features round footprint for design flexibility, 350mA drive currents, three channel control with independent input/output, -4,000V HBM, and an isolated metal base that makes wiring in series or parallel on a common heat sink possible.



 lamina mouser.com/lamina/a

We've also been working on different kinetic-metric approaches, like touch pads. Our goal is to create a suite of building blocks that can adapt to any situation when time and flexibility is of extreme importance.

Film crews still use and prefer animatronics systems even though computer-generated effects are increasing in popularity. That's our story and we're sticking to it.

If you know of a film crew that needs an inexpensive custom animatronics system, point them in our direction. We'd be happy to start a new design, so long as they don't want to shrink the monster on us. ☒

Editor's note: For more information about this project, visit www.circuitcellar.com/renesas/winners/3340.htm.

Barry Stout holds a Master's degree in Mechanical Engineering from the University of Illinois—Champaign/Urbana focusing on controls and programming. During this time, he complet-

ed all the B.S.E.E. courses as well. For the past nine years, he has worked for Lexmark, doing mostly motion control work. In the last three years, he has taken the system engineer role for consumer products. To date, he has been awarded 11 patents with 16 more pending in the fields of controls and sensors. During the past decade, Barry has helped with dozens of independent films. You may contact him at hightechanimatronics@yahoo.com.

Eugene Zeldin (ez2master@hotmail.com) holds a Bachelor's degree in Electrical Engineering from the University of Illinois—Champaign/Urbana. With many different engineering interests, Eugene has held a variety of positions, including manufacturing engineer, project engineer, hardware engineer, software engineer, technical lead engineer, and consultant on various projects in many fields: consumer (Motorola), industrial (GBC), and medical (Baxter Healthcare, Jeron). For the past four years, he has worked at Lexmark developing firmware for high-end

business laser printers. You may contact him at ez2master@hotmail.com.

PROJECT FILES

To download code and additional files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

SOURCES

DS1103 PPC Controller

dSpace
www.dspace.ltd.uk

NI PXI-6602 Counter/timer module

National Instruments Corp.
www.ni.com

H8/3687 Microcontroller

Renesas Technology, Inc.
www.renesas.com

TXLC-434 RXLC-434 RF transmitter and receiver modules

Reynolds Electronics
www.rentron.com

L6206 Dual driver ICs

STMicroelectronics
www.st.com



Serving the Electronic industry for over 25 years with New and Surplus Electronic Parts, Kits, LCD's, LED's, Audio Parts, Meters, Switches, Power Supplies, Fans, Relays, Tools, etc....

Come On Down and Visit Our Internet Store at www.bgmicro.com, or you can call our toll free number for a live person.

We'll be Expecting you!

WWW.BGMICRO.COM

1-800-276-2206



Saturday and Sunday

April 14 & 15, 2007

Trinity College, Hartford, CT

Join robot enthusiasts of all ages as they gather and compete at the 14th annual Fire Fighting Robot Contest. Feel the heat of competition in one of five divisions, and see how an exciting Robotics Symposium and Olympiad help spark the flame of robot innovation.

www.trincoll.edu/events/robot



focus more on your goals
and less on your code

PICC¹⁸™ PRO

with OMNISCIENT CODE GENERATION™

PICC-18™ PRO ANSI C COMPILER*, based on the reliable PICC-18™ Standard compiler, the PICC-18™ PRO comes with ...

OMNISCIENT CODE GENERATION™

Extracts information from multiple source files at the one time, allowing more intelligent state-of-the-art code generation that ...

- Optimizes the size of each pointer variable in your code based on its usage.
- Eliminates the need for many non-standard C qualifiers and compiler options.
- Produces more optimal interrupt context switching code.
- Customizes the functionality of the included printf library function.
- Is simple to use.

* Can run on multiple platforms including **Windows, Linux and Mac OSX.**

PICC-18™ PRO is packaged with...

HI-TECH PRIORITY ACCESS™ (at no extra cost): HI-TECH Software's 12 month **web access to updates** and **priority technical support** from arguably the industry's best team of C compiler engineers.

HI-TECH SATISFACTION GUARANTEE

HI-TECH Software offers a 30 day money back guarantee. If for any reason you are not completely satisfied with our product, simply contact us and we'll refund the purchase price.



Come and see us at Booth 937
April 3-5, 2007

Introductory Offer: Save \$300. Find out more or download a fully functional demo at <http://www.htsoft.com/products/picc18procompiler.php>



HI-TECH Software proudly supports the Microchip brand with industrial-strength software development tools and C compilers.



HI-TECH Software LLC

6600 Silacci Way Gilroy, CA 95020 USA

Ph: 800 735 5715 Web: <http://www.htsoft.com>

Inertial Rolling Robot

Jeff and Lee's H8/3664-based rolling robot is capable of inertial movement. A DC electric motor is attached to a pendulum and suspended inside an inflated ball, which provides the driving force.

We live in miraculous times. In today's world, it is now possible for you to design and build devices that even 20 years ago would have bordered on science fiction. Wheeled robots are now relatively common, and for the individual looking for novel projects, new types of motion are often sought.

In this article, we'll outline a mobile robot design that adds a twist to simple rolling (see Photo 1). We'll also discuss the electronics, software, and mechanical issues that we encountered. We hope that you enjoy this robot as much as we have!

ELECTRONICS

The robot employs a single driven

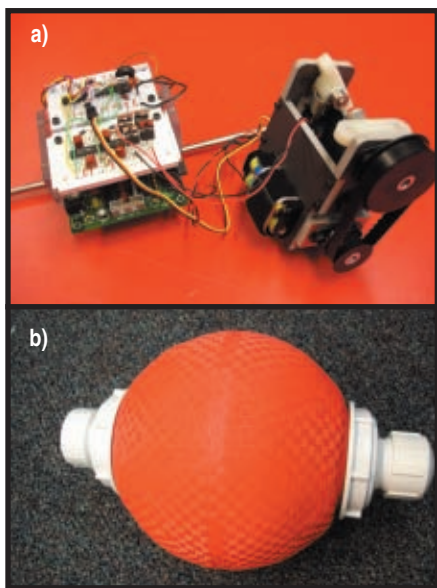


Photo 1a—The robot's inner core consists of an electronics assembly and a mechanical drive assembly. **b**—The robot's spherical shell is an inflatable playground ball. The inner core is suspended via a drive axle from the two plastic caps.

pendulum inside of a sphere. With 2 degrees of freedom, it drives and tilts the ball (see Figure 1). With this geometry, the robot can reach high speeds, steer, and jump. In addition, it doesn't get stuck when it flips over. As a toy, it exhibits enough entertaining motion to be quite endearing and very robust. As a project, it offers an exciting venture into mechanical and electronic design.

The robot's electronics perform two main functions, actuation for motors and sensors for feedback. There is actuation by means of both a bidirectional brushed DC motor and a digital proportional servo motor (see Figure 2). The two motors provide motive force and turning ability, respectively.

The robot has a pressure sensor and a motor-current sensor, although the latter was not enabled in the current software design. The main processor is a Renesas Technology H8/3664 microcontroller (see Figure 3). We used a protoboard for ease of construction. The DC drive motor is powered by a 12-V battery source consisting of 10 NiMH batteries. The servomotor and the remainder of the electronics are powered by a regulated 5-V source.

DRIVE MOTOR

We used a brushed DC drive motor that has a+ and a- inputs. Forward drive is produced by connecting the positive input to the motor-voltage supply and the negative input to ground. Reverse drive is produced by switching the two inputs. Power amplification and bidirectional rotation is accomplished by means of a MOSFET H-bridge. International Rectifier IRL3103 MOSFETs with 64 A of

continuous drain current were selected because of their low on-state resistance and their ability to handle extreme currents without a heatsink.

The motor has a stall torque of only 2 A, but you can upgrade to a larger one if you want. The average voltage to the motor is varied by switching between high and low states of the H-bridge at a high frequency (10 kHz). This is accomplished by pulse-width modulating the voltage signals to the motor. The MOSFET H-bridge is driven by the two International Rectifier IR2184 Half-bridge MOSFET drivers. The IR2184 is designed for automotive use and will operate with 10 to 25 V. Thus, the same 12-V source from the batteries is used for both the gate-drive supply to this driver as well as the voltage supply for the MOSFET H-bridge.

The MOSFET driver has only a single IN command, which makes for convenient interfacing. The driver will turn on the high-side MOSFET for a logic level of 1 and the low side for a logic level of 0. This behavior is optimal in an H-bridge circuit with an inductive load, because the current

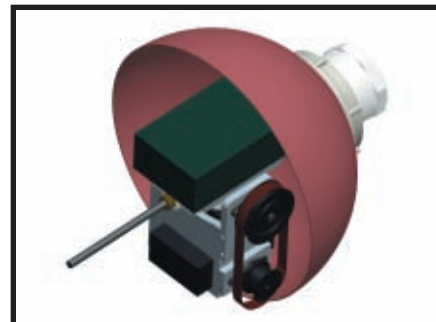


Figure 1—The robot was designed with CAD software to ensure the precise fit of components. In this view, the spherical shell is cut away to show the robot core.

will automatically continue through the low side of the bridge (see Figure 4). However, by turning on the low-side MOSFET, current can be carried more efficiently through the MOSFET, rather than through the parallel diode. Even though the IR2184 can be connected to the microcontroller directly, a buffer circuit is used to prevent damage to the microcontroller in case of misconnection. The buffer circuit also performs convenient logic functions so that only one PWM channel and one I/O port are necessary on the microcontroller.

Table 1 is a truth table for this logic circuit, which uses a 7404 inverter and two 7402 NOR gates. Thus, the logic circuit sends the PWM signal from one driver to the other, depending on the value at the I/O port. The opposite driver is sent a logic level of 0.

A Hall effect current sensor (Allegro ACS706) is set up in series with the motor in order to measure motor current for control purposes. The sensor's connection is relatively straightforward. It outputs a voltage corresponding to the motor current offset at 2.5 V for zero current, with a sensitivity of 130 mV/A, and has a range of ± 15 A. The sensor is connected directly to an ADC port, but it is not enabled for control in software at this time.

We were accidentally shipped a 6-V

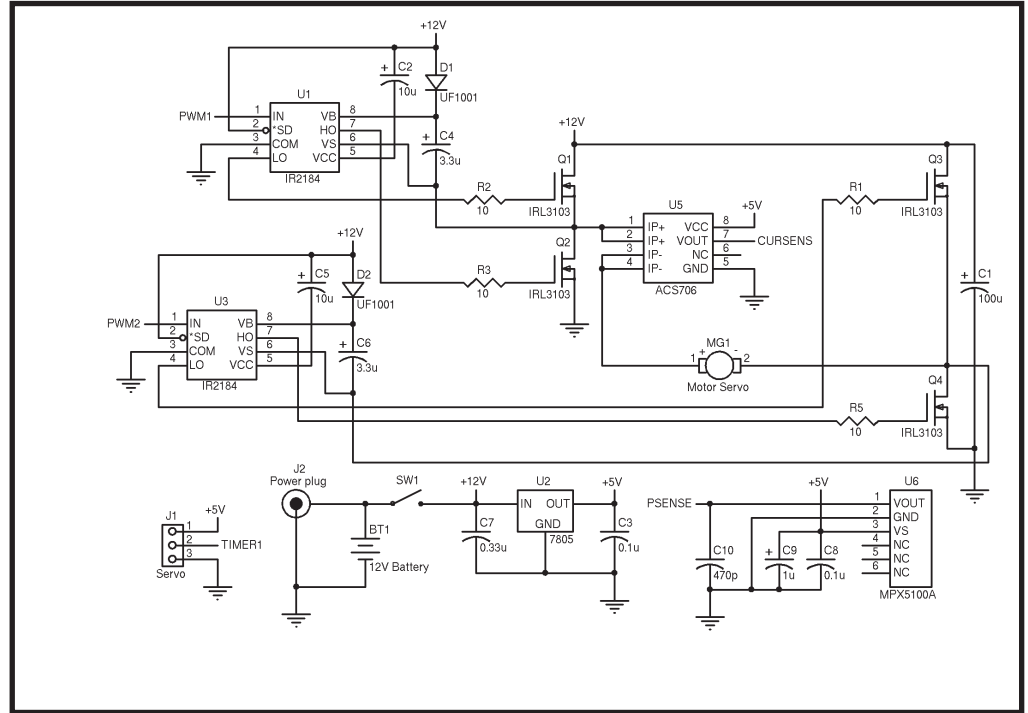


Figure 2—The H8/3664 microcontroller connects to several components that provide control and sensing for the robot. The circuit is basic and easy to connect to most other microcontrollers as well.

motor instead of a 12-V motor. The motor worked well for a while, but it eventually burned out. Be sure to purchase a 12-V motor for this electrical design.

SERVO MOTOR

The servo motor is a standard remote-control hobby servo motor. It has a built-in position feedback loop, based on an internal potentiometer, which enables its position to be commanded directly.

The servo is an extremely easy component to interface with a microcontroller. Power is supplied with a 5-V source and control is provided through a 50-Hz PWM signal. The PWM signal is connected to the FTIOB output port on the H8/3664 through a 7404 inverter, which is used as a buffer.

PRESSURE SENSOR

We used a Freescale Semiconductor MPX5100A absolute pressure sensor (with a range from 0 to 16.7 psi) so the inflated ball

structure can sense the environment. When the ball hits a barrier, it can be picked up as a pressure transient.

The absolute pressure sensor measures pressure with respect to a perfect vacuum. Thus, the sensor reads both atmospheric and ball pressure. Since atmospheric pressure at sea level is 14.7 psi and the ball may be inflated with 1 psi, the sensor has only a small range that it can sense. However, because impacts of the ball produce both positive and negative changes in pressure, the sensor is effective at detecting collisions. This sensor is easy to interface with the microcontroller: V_{OUT} from the sensor is connected directly to an ADC port. Decoupling and filtering capacitors are recommended in the MPX5100A's datasheet.

SOFTWARE

The software provides the basic drivers necessary to read sensor information and to control the two motors. The drivers are set up to run in interrupt loops or operate continuously. (Continuous conversions are set up for the A/D sensor readings.) Communication with these low-level drivers is available via function calls. Application functionality is available by pro-

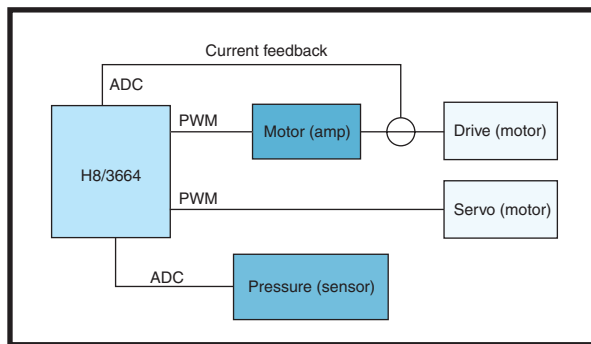


Figure 3—The H8/3664 microcontroller interfaces with PWM output to a drive motor and a steering servomotor. There is input from a pressure sensor and a current sensor on the drive motor, which interfaces through the microcontroller's ADC port.

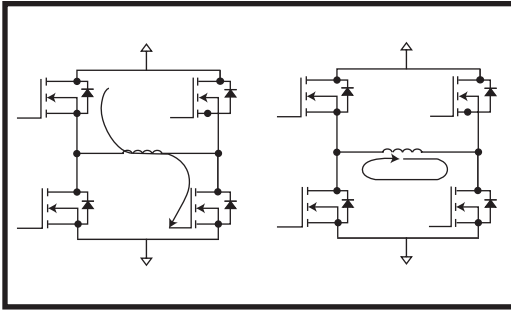


Figure 4—During the PWM on state, current passes through the upper-left MOSFET and then the lower-right MOSFET. When the PWM switches off, current continues through the motor inductance and then through the lower-left MOSFET. To reverse the motor's direction of rotation, the opposite side of the H-bridge is switched.

programming the desired robot behavior in the main function and calling low-level driver function calls.

For our robot, the main function contains a simple sinusoidal trajectory of the drive motor and the steering servo motor, followed by a straight trajectory. This trajectory demonstrates some basic capabilities of the robot. Details about the low-level drivers and the complete source listing are posted on the *Circuit Cellar* FTP site.

We experimented with the pressure sensor, but it was not used in the application due to the difficulty of debugging. Sample code is provided, which should theoretically enable the robot to switch from a straight trajectory to a sinusoidal trajectory when an obstacle is hit and thus change direction.

DC DRIVE MOTOR CONTROL

The code for the DC drive motor and the steering servo motor both use PWM. They're both in the `pwm.c` file on the *Circuit Cellar* FTP site.

The brushed DC motor must have variable speed and torque capabilities, and it must operate bidirectionally. The motor currently uses voltage control, but it has the capability to use current control. Current control hasn't been implemented in software. The torque in a brushed DC motor is proportional to the current through the motor (related by the motor's torque constant), so it may be advantageous to use current control if you want to control torque.

Voltage control is achieved by a PWM signal to the motor at

approximately 10 kHz. This is implemented on the H8/3664 microcontroller using Timer V. Since Timer V does not have a buffered register for changing the duty cycle of the PWM, an interrupt is set up on compare match to update the value in the compare register and eliminate the possibility of erroneous pulses being generated.

SERVO MOTOR CONTROL

The servo motor is controlled by pulse-width encoding at a rate of 50 Hz. The pulse for standard hobby servos ranges between 1 and 2 ms with 1.5 ms corresponding to the neutral position.

Timer W on the H8/3664 microcontroller is set up to overflow at a frequency of 50 Hz, or a 20-ms period. The period in terms of Timer W counts is 49,152. The compare-match register is set up in buffered PWM mode, so the PWM-output signals can be generated without concern for the timing of writing to the compare-match register. The new value will be automatically loaded at the beginning of the PWM cycle, eliminating the chance for making writes at the wrong time and causing erroneous PWM-signals.

In terms of Timer W counts, 1 to 2 ms corresponds to a GRB value ranging from 2,458 to 4,915, with a neutral position of 3,686. Limit checks are

included to ensure that the servo is not overdriven.

A/D conversion for the pressure sensor is accomplished in the `adc.c` file that's available on the *Circuit Cellar* FTP site. Only one channel is used (AN0) in order to store the value of the current pressure. The ADC is set up to run in continuous operation, and the current value from the pressure sensor can be read at any time.

MECHANICAL COMPONENTS

In this article, we present our parameters for constructing this robot; however, customization is the spice of life. In order to modify this robot, you should keep a few things in mind. The ideal implementation of this robot has its weight as far away as possible from the robot's driven axle. This allows the robot to convert a maximum amount of momentum or torque into rolling, because the robot essentially moves based on the "falling" of the pendulum. If the pendulum rotates too far, the ball will counter-rotate, so balancing motor torque and pendulum torque is important (see Figure 5).

Simulating the robot's motion can also help. Using the motion equations in Figure 6, different physical parameters and motor-control scenarios can be modeled to determine the optimal design. These equations can be solved using any numerical solver and plotting the output. In order to model the motion in the tilt direction (the plane perpendicular to rolling), the same equations can be simplified and used. However, the simplest and most important analysis of the tilt motion is the turning radius:

$$r_{TURN} = \frac{r_w}{\tan(\varphi)}$$

where φ is the tilt angle of the pendulum. These two-dimensional simulations should be more than adequate for predicting the robot's basic motion.

MANUFACTURING

We used off-the-shelf mechanical components for this design. Most of the parts can be purchased at your

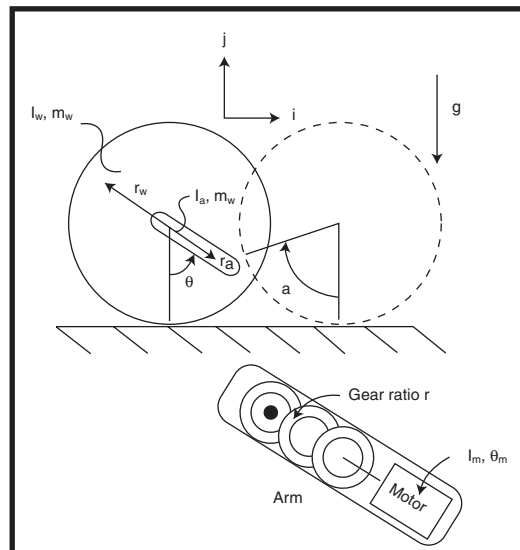


Figure 5—Driving the internal pendulum causes the robot sphere to rotate.

local hardware store.

The mechanical components that you must build must consist of several plates, which when assembled make up a pendulum and the inner-workings of the rolling robot. The pendulum is essentially a box with a series of threaded holes that are used to attach the drive motor and servo-mechanism.

The first step is to cut out the rough shapes of the different plates that make up the pendulum and a custom universal joint. The method of creating these plates depends on the materials you work with. We chose aluminum, which requires machining with a water-jet or milling cutter. However, the body could just as easily be made from polycarbonates or acrylics and cut out with a scroll saw.

Once you cut out the parts, you must drill holes to attach the drive components. Next, tap the drilled holes. Taps can be purchased as a set or individually at your local hardware store. Make sure the taps match the

| PWM (TMOV port) | I/O (P72 port) | PWM1 | PWM2 |
|-----------------|----------------|------|------|
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |

Table 1—A logic circuit provides an interface to a single PWM output and a single digital I/O port on the microcontroller. In this truth table, TMOV and P72 are the outputs on the H8/3664 microcontroller. PWM1 and PWM2 are the inputs to the H-bridge.

screws used in the project.

The brackets used to attach the servo pushrod should be made slightly different than the other plates. First, cut out a strip of 18-gauge sheet metal, drill the holes, and then bend the bracket. The part is small enough so that drilling the holes before bending makes the manufacturing process much easier.

The pushrod itself is easily made by cutting a short length of threaded rod and connecting two pushrod clevis pins. The next step is to cut the shafting to length in order to create a drive shaft, gear shaft, and a support shaft. There are several ways to accomplish this. The simplest is to use a hacksaw.

Once you cut the shafts to size, carefully round the ends with a file. Any raised burrs will make it difficult to slide anything onto the shaft. Finally, use a die to thread the drive shaft, you should need to thread only about 1" on either end of the shaft.

The final step is to cut two holes into the rubber kick ball.

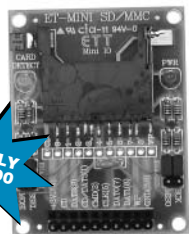
First, determine an axis through the ball. This can be accomplished by deflating the ball and folding it into quarters. The points of the quarters should approximate the location of an axis through the ball. Next, inflate the ball. Using a compass, trace two circles on the ball and use the lock ring from the drain assembly to determine the necessary diameter of the circles.

Lastly, deflate the ball and use a pair of scissors to cut out two holes defined by the circles you have drawn. Make sure you don't cut the ball's plug; otherwise, it will be impossible to inflate.

ASSEMBLY

Once the mechanical parts are man-

Save Up To 60% On Electronic Components



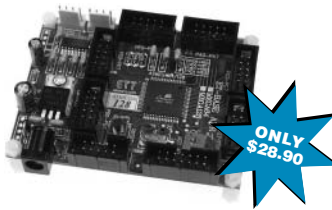
ONLY \$6.90

Exciting New Mini-Boards

- Wide Range Including,
- Real Time Clocks (DS1307)
 - Memory Cards
 - Power Supplies
 - DC Motor, Stepper Motor and More

Powerful New ATmega Controller

- Includes ATmega128 Microcontroller
- High-Speed Operation
- Heaps of I/O
- In-Circuit Programming
- Ideal Embedded Controller



ONLY \$28.90



ONLY \$4.90

Save Heaps on Components

We carry a wide range of Integrated Circuits, Microcontrollers, Capacitors, LED's and LCD's.

All at very competitive prices.

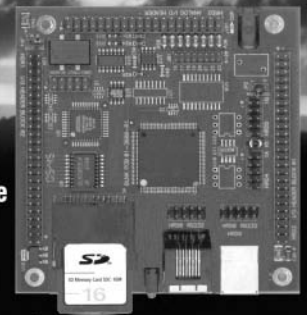
We are your one-stop shop for Microcontroller Boards, PCB Manufacture and Electronic Components.

www.futurlec.com

Data Acquisition & Control Computer

iPac 9302

- Cirrus Logic EP9302 ARM9 200 Mhz Processor
- Floating Point Math Engine
- 2 USB 2.0 Host Ports
- SD/MMC Flash Disk Slot
- 40 Digital GPIO Lines
- 1 10/100 Base-T Ethernet port
- 5 channels of 12 bit A/D & 3 PWMs
- 1 RS232 & 1 RS232/422/485 Serial Port
- Battery Backed Real Time clock/calendar
- Linux, CE, & .Net Micro Framework



The iPac has enough I/O for demanding applications & with a size of 3.5" x 3.8" it can fit almost anywhere. Please contact us for more information.

Since 1985
OVER
22
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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

$$\begin{bmatrix} I_a + m_a r_a^2 + I_m & m_a r_a r_w \cos(\theta) + I_m \\ m_a r_a r_w \cos(\theta) + I_m & I_w + (m_a + m_w) r_w^2 + I_m \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -m_a g r_a \sin(\theta) \\ m_a r_a r_w \dot{\theta}^2 \sin(\theta) \end{bmatrix} + \begin{bmatrix} T \\ T \end{bmatrix}$$

Figure 6— I_a is the moment of inertia for the pendulum. m_a is the mass of the pendulum. r_a is the length of the pendulum. θ is the angular position of the pendulum. $\dot{\theta}$ is the angular velocity of the pendulum. $\ddot{\theta}$ is the angular acceleration of the pendulum. I_w is the moment of inertia for the wheel. m_w is the mass of the wheel. r_w is the radius of the wheel. α is the angular position of the wheel. $\dot{\alpha}$ is the angular velocity of the wheel. $\ddot{\alpha}$ is the angular acceleration of the wheel. I_m is the inertia of the motor. T is the applied torque. g is the gravitational constant.

ufactured and the circuit has been built, you can assemble the components and finish the robot. Assembly is relatively straightforward. After examining the exploded view, drawing it should be an easy task to determine how the pieces fit together (see Figure 7).

The only difficult part of the assembly process involves inserting the mechanical parts in the rubber ball, because some assembly must take place inside the ball. We recommend fully testing the electromechanical assembly before attempting to place it inside the ball.

It is easiest to start by assembling three sides of the pendulum subassembly (M1, M18, and M26). The flanged bearings (M17) can be loosely fitted into the front and back plates, while the back plate spacer (M2) and servo motor can be tightly secured to the back plate. The servo arm is near the bottom of the pendulum. One end of the servo pushrod can be attached to the servo arm at this time and the other end can be attached to the servo pushrod-bracket (M25).

The motor subassembly can be put together by attaching the motor to the motor plate (M22) and the motor plate to the sliding plate (M27). Next, a flanged bearing is loosely installed. Then the bore reducer (M8) is inserted on the motor shaft, a washer is placed on it, and the power sheave (M23) is pressed and tightened onto the bore reducer and motor shaft. After checking that the motor is free to rotate, the flanged bearing can be tightened down. The motor subassembly is attached to the pendulum subassembly by loosely connecting the slide plate (M27) to the pendulum front plate (M18).

The next step is to attach two bearing blocks and the electrical housing

(M16) to the universal joint plate (M29). As an additional step, you might want to preassemble the miter gears (M20), drive shaft (M15), and gear shaft (M19) to check the tolerance of the gears and set the alignment of the bearing blocks. Once clearances are checked, the assembly can be returned.

The pendulum subassembly can be finished by attaching the remaining side plate (M26). In addition, one of the bearing blocks can be attached to the back plate (M1) by using the support shaft (M28) with two washers and two collars. Another bearing block can be attached to the front plate (M18) with one miter gear (M20), two washers, the driven sheave (M14), and the gear shaft (M19). Attach the battery boxes with Velcro. The belt (M7) can

be placed on the pulleys and tensioned. Then the slide plate (M27) can be tightened down.

Any electrical connections can be made and the pendulum subassembly can be inserted into the ball by slightly stretching the opening. Then the universal joint assembly can be inserted and the drive shaft with the miter gear can be attached. At that point, the universal joint plate can be connected to the bearing blocks on the pendulum along with the servo pushrod bracket.

The final step is to insert the rubber gaskets and the flanged ends of the drain bowls into the ball and bolt the driveshaft to the sink drains with lock nuts. Next, tighten the sink drain lock rings firmly to the ball and apply Teflon tape to the sink-drainpipe connections. Finally, flip on the power switch with a screwdriver, tighten both end caps, inflate the ball, and stand back! (Drawings of each step are posted on the *Circuit Cellar* FTP site.)

FINDINGS

The rolling robot was an intriguing and difficult project to work on. The mechanical design is novel and non-trivial. However, we found that the

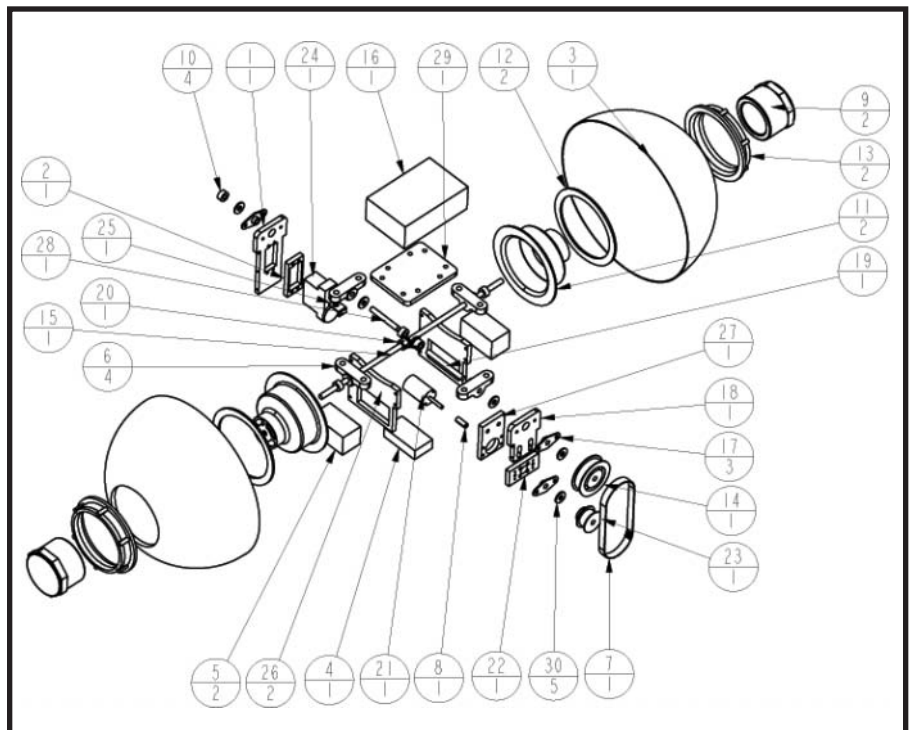


Figure 7—This figure shows an exploded view of the mechanical components of the rolling robot. Numbers in the top half of the bubbles correspond to the parts list that can be downloaded, while numbers in the lower half specify quantity.

resulting unique motion available with this robot was worth the effort. With that being said, we learned a few things that might be helpful in building a similar rolling robot.

The biggest difficulties were caused by the tight working area inside the rubber ball and not having powerful enough drive components. While assembling the robot inside a rubber ball through small openings is possible, debugging is nearly impossible. Thus, during the debugging process, use a hamster ball or some other clear sphere that allows easy access to the components.

It is also important to ensure that the components used match both the mechanical and electrical requirements. Initially, we burned out the drive motor because we were shipped a 6-V motor, instead of the 12-V motor that we expected. With a little patience and some elbow grease construction of the robot should go smoothly.

We hope that this article encourages you to build a similar robot. We had a

lot of fun building ours, and the resulting product is a very interesting accomplishment. Just remember us when you take over the world with your rolling robot army. ☐

Jeff Bingham (bingjeff@gmail.com) has a B.S. in Mechanical Engineering from Idaho State University and is pursuing his Ph.D. in the same field at the Massachusetts Institute of Technology. In his spare time, he likes to build things, such as small autonomous rolling spheres.

Lee Magnusson (leem@alum.rpi.edu) has a B.S. in Mechanical and Materials Engineering from Rensselaer Polytechnic Institute and an M.S. in Mechanical Engineering from the Massachusetts Institute of Technology. His interests include electromechanical modeling and design. Lee currently works in the MIT Biomechanics Lab (<http://biomech.media.mit.edu>), where he designs the latest in active powered prosthetics.

PROJECT FILES

To download the code and additional files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

RESOURCE

Freescale Semiconductor, "MPX5100A Datasheet," 2005, www.freescale.com/files/sensors/doc/data_sheet/MPX5100.pdf.

SOURCES

ACS706 Linear current sensor
Allegro Microsystems, Inc.
www.allegromicro.com

MPX5100A Absolute pressure sensor
Freescale Semiconductor, Inc.
www.freescale.com

IR2184 and IRL3103 Half-bridges
International Rectifier
www.irf.com

H8/3664 Microcontroller
Renesas Technology Corp.
www.renesas.com

The Better Bot

Get \$20 off your PIC[®] MCU Robot kit today!
Prices start at just \$209! Features include:

- * PIC16LF877A (ICD capabilities)
- * Electronic Compass
- * Text-to-Speech Converter with Speaker
- * Infrared Detection
- * Ball Bearing Servo Motors...

[Code CCBOT must be used at time of purchase. Not valid with any other offers. Expires April 30, 2007.]

CCS
Custom Computer Services, Inc.
phone 262.522.6500 x35
www.ccsinfo.com/ccad

PIC[®] is a registered trademark of Microchip Technology Inc. in the U.S. and other countries.

Enhance Your Connectivity

The RCM4000 — Ethernet And Much More

- Ethernet with royalty-free TCP/IP stack
- Up to 25 configurable GPIO
- On-board 12-bit A/D
- Up to 5 serial ports
- 512K flash (program), 32 MB NAND flash (data)
- Rabbit[®] 4000 at 58.98 MHz

Complete Development Kit \$149
Regular \$239 Limited Time Offer

RABBIT Semiconductor

Order Online At rabbit4000.com

06227

Then and Now

Dave traces the 200-issue history of Circuit Cellar. Embedded applications have come a long way since 1988.

It's always fun to take some time out to remember how things used to be, and to see how they've changed over the years. The 19 years that *Circuit Cellar* has been in existence as an independent publication (not counting Steve's *BYTE* column) have seen the embedded computing field—and indeed, the microcontroller itself—grow from a tiny niche market into a major industry.

We've had over 800 authors (829 as of October 2006) contribute articles to these pages, not to mention the countless advertisers over the years, and this represents a significant cross-section of the industry from which we can draw some lessons and maybe even make some predictions about the future.

If you have been active in this field since the 1980s, you'll resonate with the following ramble through the memory banks; if not, maybe you'll learn something new!

My own involvement with embedded computers began around 1976. The father of one of my high school classmates was an electrical engineer, and he had built himself one of the 8008-based "Mark8" computers from the plans published in *Radio-Electronics* magazine in July 1974.

He was involved in the power industry, and one of the problems he faced had to do with three-phase power factor correction capacitors. It seems that these were packaged three to a container, connected in delta fashion, with just the three leads at the corners of the triangles brought out. Each capacitor had a series fuse for protection

that wasn't directly accessible. It was very difficult to determine whether any given capacitor canister had blown fuses.

It occurred to him that the phase relationships among the three currents in the three leads would change in various recognizable patterns, depending on which fuses were conducting current. It also occurred to him that one of these new-fangled microprocessors would be a good way to monitor the zero crossings at the outputs of the current transformers and report the problems. I helped him develop the software to accomplish this, written in 8008-machine language (numbers, not assembler mnemonics) and entered into the machine via the front-panel toggle switches. It was extremely simple by today's standards, but it nonetheless represented an example of a real-time, microprocessor-based embedded computer system.

Fast-forward through the next 12 years, which saw the rise and fall of the 8080 and CP/M machines, along with many other TV-based computers and game consoles and the eventual rise of the IBM PC (see Photo 1). In early 1988, there were about 8 million DOS machines deployed, mostly 8088/86 (IBM PC and PC/XT, along with compatible systems from other manufacturers) and 80286 (PC/AT) based. The 80386 had been available for a few months (at 16 and 20 MHz) and the 68030 was just being introduced by Motorola. Note that 1,200-bps modems were pretty much state of the art, but within a year, 9,600-bps units were available. Microsoft Windows didn't exist yet, at least not in usable form (version 3.1 appeared in 1992). This was the era of 1-Mb DRAM chips and 256-Kb SRAMs. Desktop CPUs operated at frequencies in the 10- to 20-MHz range. Photo 2 shows the evolution of the associated mass storage technology.

BIRTH OF CC INK

It is instructive to go back and look at the first issues of *Circuit Cellar Ink*, published bimonthly in 1988. The first issue dealt almost exclusively with video projects, including a motion-triggered multiplexer from Steve Cia-rcia, a hand scanner for security applications by Ed Nisley, and a video format converter from Mark Voorhees. The exception was a discussion about RISC CPU architectures by Tom Cantrell.

The second issue focused on more fundamental data-acquisition issues, including tempera-



Photo 1—Personal computing in the 1980s. On the left is my CP/M machine, which is based on the J.B. Ferguson "Big Board." A spare CPU board sits in front of the keyboard. On the right is the Zenith Data Systems PC (an IBM PC/XT clone) that replaced it. To the left of the CP/M monitor is a KIM-1 board, a 6502-based single-board "system" that was popular with hobbyists. It includes a hex keypad and display, along with a monitor program in PROM to operate it.

ture and high-resolution timing. Issue three explored X-10 power-line communications, video signal timing, and software emulation (bit-banging) of UARTS.

In issue 4, Steve and Ed teamed up for a scanning ultrasonic sensor using a stepper motor. Tim McDonough and Dennis Grim wrote an article about a stepper motor-based robot arm. Ed also wrote an article about making use of the IBM PC joystick port.

Issue 5 covered a broad range of topics, including the debut of Jeff Bachiochi's "From the Bench" column, in which he discussed RS-232 interfaces. Also, a 10-MHz, 8-bit ADC was presented, along with more about the X-10 system and Steve's Remotely Operated Video-based Electronic Reconnaissance (ROVER) system.

Issue 6 was the final issue in 1988. It included articles about the ROVER software, fiber optics, an update on the ImageWise video digitizer, and a complete 6809-based data acquisition system.

I should note that a significant portion of each of these issues was devoted to engineer-to-engineer communications, both in the form of letters to and from the *Circuit Cellar* staff and reprints of selected conversations pulled from the *Circuit Cellar* BBS. We'll come back to this topic later.

THE TECHNOLOGY BASE

There were very few outside advertisers in that first year, so, to get a broader feel for the general level of the available technologies at the time, it's helpful to look at some of the other journals from the same year, such as *BYTE* and *Dr. Dobb's Journal*. (Finally! Being a magazine pack rat pays off!)

Obviously, the key element in an embedded computing application is the processor itself, which might be a microprocessor that requires external memory and I/O, or a microcon-



Photo 2—This collection of disks (sitting on top of the CPM disk drive cabinet shown in Photo 1) shows the advances in magnetic disk storage. From left to right across the bottom are a 14" platter and an 8" platter. The former shows evidence of a head crash about halfway out. To the right of these is the ubiquitous 3.5" hard drive, which comes in capacities of hundreds of gigabytes these days. Below that is a 4-GB "Microdrive" that fits in a compact flash memory slot. From right to left across the top are 8", 5.25", and 3.5" floppy disks.

troller that integrates most of that on-chip. The embedded applications appearing in the pages of *Circuit Cellar* were mostly based on 8051-architecture chips, Z80/64180-based boards, and in a few cases, the 68000 (see Photo 3).

Over the years since then, the 8-bit microprocessor has been almost completely replaced by increasingly powerful single-chip microcontrollers, which now have CPU cores of 8, 16, or 32 bits and operate at clock rates into the hundreds of megahertz. Microprocessors are still used in the larger embedded systems that need huge amounts of

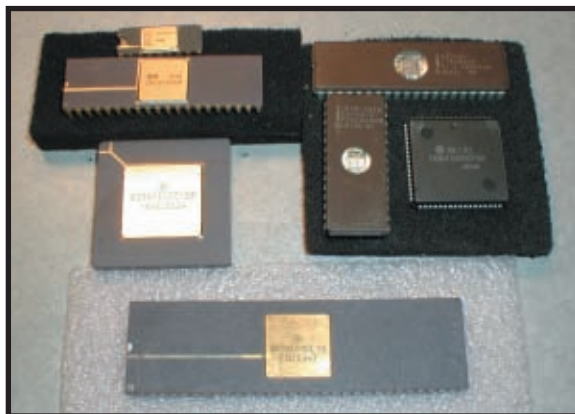


Photo 3—The evolution of the underlying technology is shown by this collection of chips. At the upper left are the Intel 8008 and 8080A. In the group at the upper right are the single-chip Intel 8748 microcontroller across the top, a 2764 UV-EPROM on the left, and a Hitachi HD64180 CPU chip in a PLCC package. Across the bottom is the Motorola MC68000 CPU in its enormous 64-pin DIP package. Clockwise from there is the newer MC68020 in a pin-grid array package.

storage, usually in the form of a prepackaged PCB module that takes care of the low-level details of interfacing to high-performance chips.

In 1988, nearly all embedded firmware was stored in UV-erasable programmable read-only memory (PROM) chips. These chips contained arrays of bits that could be flipped one way (typically 1 to 0) electrically, but required exposure to ultraviolet light in order to flip them the other way (0 to 1). Any modification to the firmware required getting a freshly-erased chip out of the eraser, programming it, swapping it with the chip in the target, and then putting the old chip

in the eraser.

Some microcontrollers were available with on-chip UV-EPROM, but these tended to be quite expensive and still needed to be removed from the target circuit for erasing and programming.

One of the biggest fundamental changes for embedded firmware was the development of the electrically erasable PROM chip, known as EEPROM or flash PROM, the latter referring specifically to chips in which the minimum erase size is larger than a single address—either a "sector" or the entire chip.

Finally, in-circuit reprogramming became a possibility, and microcontrollers with on-chip code memory became much more widespread. Indeed, a significant fraction of *Circuit Cellar* articles today are devoted to the topic of chip programming hardware and software.

Gate arrays existed in 1988, but they were relegated to high-volume applications by the high nonrecurring engineering (NRE) costs. Programmable logic for low-volume applications was limited to PROMs, programmable logic devices (PLDs), and programmable array logic (PAL).

All three types of devices share a common architecture. The difference is in which parts

of the architecture are field-programmable. The architecture is a set of input pins that drive an array of AND gates. The outputs of the AND gates form an array of intermediate signals that drive a second array, this time of OR gates. The intermediate signals are often called pterms (product terms), because the AND operation is sometimes referred to as a Boolean product. The outputs of the OR gates are the outputs of the device.

In a PROM, the AND array is fixed in the form of the internal address decoder, which has the further restriction that only one pterm (a word select line) is active at a time. Nonetheless, useful logic can be (and was) implemented using PROMs to drive the registers of state machines.

The PLD is the most general device, in which both the AND array and the OR array are programmable.

The PLD had a relatively short lifetime, however, because manufacturers soon realized that the full generality of having every pterm available to every output was rarely used, and it was

using up valuable chip real estate that could be better used in other ways. The PAL contains a programmable AND array, but a fixed OR array, in which groups of pterms are OR'd together for a particular output. At this point, on-chip flip-flops were incorporated into some devices, with feedback directly into the AND array, and now you could build simple on-chip self-contained state machines.

The PAL proved to be extremely popular and has evolved into the complex programmable logic device (CPLD) we know today. However, CPLDs still make certain assumptions about the ratio of internal logic to external I/O, and this trade-off isn't always appropriate for certain applications. The field-programmable gate array (FPGA), which first appeared in the early 1990s, has evolved as an alternative approach that fits the other end of that spectrum very nicely.

DEVELOPMENT TOOLS

Let's talk a bit about the development tools for embedded computing.

Chief among these, of course, is the issue of programming languages. Back in 1988, assembly language was king, but nearly every microprocessor had some sort of dialect of BASIC available as an interpreter as well. C compilers were also becoming available for most of the small processors. C++ was just starting to make an appearance (for larger computers), but did not yet have any significant "mind share."

The other big development tool for an embedded system is the debugger. In 1988, this was mostly accomplished by either ad hoc means—if the code gets *here*, blink this LED—or by the use of in-circuit emulators. The latter were usually expensive, and required that the target chip be removed and replaced with a cable connected to a fairly large box containing a special version of the chip, or some other implementation of the architecture.

The JTAG interface standard was adopted by the IEEE in 1990. Originally intended as a means for testing chip-to-chip connections through a PCB, it was sufficiently general that



I missed issues

Fill in the Missing Pieces

with Circuit Cellar magazine's CD-ROM archives.

These convenient PDF archives contain complete issues, just as they appeared when they were printed. Get back on track and follow along the uninterrupted Circuit Cellar timeline.

For package deals on all of our archives, visit: www.circuitcellar.com/archives/

features relating to the chips themselves could be added as well. It took a few years to catch on, but by the late 1990s, many chips could be debugged and/or programmed through their JTAG interfaces. I think it took a while for the manufacturers to realize that the investment of chip area devoted to these functions was well worth it down the road in terms of convenience to the design engineer, which translated into more design-ins and ultimately greater sales volumes.

Even chip vendors who did not adopt the JTAG standard came up with their own low-pin count interfaces for debugging and programming their chips. Today, it's practically unheard of NOT to do in-system programming and debugging.

EMBEDDED APPS

Let's take a second look at the kinds of applications that *Circuit Cellar* addressed back in 1988. They can best be described as an interrelated mesh involving video signals, home control, general data collection, and communications. Communications consisted primarily of RS-232 asynchronous serial links.

Interestingly enough, if you look at the articles published by *Circuit Cellar* in 2006, there's a very similar pattern. Video projects still deal with NTSC signals, although there's a smattering of information related to higher-resolution computer displays. Ethernet, USB, and various wireless technologies have largely supplanted RS-232 for communications, enabled by the availability of single-chip implementations, although we still run articles dealing with bit-banging asynchronous serial interfaces on small microcontrollers. The latest tricks involve bit-banging video and USB interfaces.

Home control and basic data acquisition, including weather monitoring, continue to be perennial favorite topics. The issues related to driving and using stepper motors are still revisited from time to time.

That's not to say there isn't a much broader range of topics that has been enabled by all of the new chips that are available now. MEMS sensors for acceleration, rotation, and pressure have appeared in all kinds of applications. Mixed-signal chips for audio and

RF have brought the cost of RF spectrum analyzers and other instruments down to the hobbyist range. The proliferation of communication interfaces has brought about the need for devices that can convert from pretty much any one protocol to any other.

Although GPS has actually been operational from the early 1980s, it has been only in the last 10 years or so that the receiver technology has dropped in price (and size) to the point where it is suitable for mass consumer applications. Nowadays you can get an OEM receiver module for a few tens of dollars.

The other big wireless technology kicker has been the wireless telephone a.k.a. "cell phone," named for the honeycomb-like arrangement of the terrestrial base stations, known as "cells." Although the underlying technology is highly proprietary, many embedded applications have been able to make use of it by treating it as a "black box." The suppliers of OEM interface modules have been able to take advantage of the technology advances that are driven by the market for consumer cell phones.

NETWORKING

I alluded to engineer-to-engineer communications earlier. In 1988, this consisted of bulletin board systems (BBSs), and for those who had access to it through work or school, Usenet-based newsgroups, and e-mail lists.

A BBS was typically a PC of some sort connected to one or more phone lines via modems. Users would dial into the machine periodically to download new messages and upload their own responses. The signal-to-noise ratio tended to be very high, since everyone participating had to have sufficient motivation and discipline to participate in the first place.

Usenet was based on (mostly) Unix-based minicomputers and larger systems that phoned each other up periodically using a protocol known as UUCP (Unix-to-Unix CoPy) to exchange mail and news traffic. Some machines had full-time access to a high-speed network known as ARPANET, but the concept remained the same. Again, the value of the messages tended to be high, since you had to have certain technical qualifications in order to get access to the

machines in question to begin with.

Contrast this to the situation today. ARPANET has evolved into the Internet we know and love/hate today, and there are virtually no barriers to access for anyone, be they technically inclined or not. BBS machines have disappeared, although I hear there are still a few diehards out there. Usenet news and e-mail list servers still exist but most unmoderated groups have been inundated with so much off-topic content that they're virtually useless.

The advent of the World Wide Web and the underlying HTTP servers has created a new form of group communication, the "web forum," which relies on a more-or-less graphical user interface to convey the same kind of information as before. Frankly, I don't see the point, and even with broadband Internet access, I find that the real-time interaction with the forum server via a web browser is orders of magnitude slower than reading locally-cached news and e-mail messages using my mail-client software.

SKILLS IN DEMAND

Well, I hope this wander down memory lane has been at least entertaining, if not somewhat instructive. I think it's interesting to note what hasn't changed in the last 19 years, along with all of the new opportunities that have arisen. I think it's clear that if you have the ability to put hardware and firmware together, there will always be a need for your skills. 📧

Dave Tweed has been developing hardware and real-time software for microprocessors for more than 30 years, starting with the 8008 in 1976. His system design experience includes computer design from supercomputers to workstations, microcomputers, DSPs, and digital telecommunications systems. Dave currently consults for various companies in projects involving embedded DSPs and FPGAs. You may reach him at dtweed@acm.org.

RESOURCE

D. Tweed, Circuit Cellar Ink Index, www.dtwweed.com/circuitcellar/index.htm.

Generic Modbus Simulator (Part 1)

Theory and Preparation

Aubrey introduces the topic of a Modbus implementation on a PC. He covers the Modbus protocol and an approach for documenting the register set of a Modbus slave and even a family of slaves in Excel.

In the industrial world, there are many data-transmission standards that have endured. The 4- to 20-mA current loop and the Modbus communications protocol are two examples. It is not only resistance to change that maintains their popularity. They are simple to understand, error resistant, and enjoy a huge installed base. And probably just as important, they are in the public domain.

The Modbus concept, originally developed by Modicon, allows a master controller to interact with a remote slave over an RS-232 serial line to read and modify industrial inputs and outputs. The concept has been expanded and standardized to use modems and RS-485 communication, allowing longer distances of communication and multiple remote controllers.

In this series of articles, I will describe how to create a universal Modbus simulator residing on a PC. To do so, I will build on the concepts that I first presented in "Visual Basic 2005 and the Serial Port" (*Circuit Cellar* 197, 2006), in which I described the creation of serial port communications using Visual Basic 2005 Express.

I have been involved

with creating a Modbus interface for two products. In both, I needed to simulate a controller in a way that I could send the same communication again and again, while trying to debug the software. In its simplicity, Modbus categorizes the I/O space as single-bit digital (coils) registers or as 16-bit registers. Interpretation of holding registers and input registers (16 bit) is left to the slave's designer. They could be used for ADC values (obviously this

would use a 16-bit register) as well as relays and proximity sensors. The master controller reads and writes to these registers when the slave is addressed over the serial network.

Each product has a different interface, so I needed to create a technique for the user to enter the memory map on the PC and then interpret this data in order to access the target device over the Modbus. The slave's registers are normally presented as a table to document the address space for the network integrator.

One look at the details on the Datagrid class of Visual Basic, and I decided that Excel contained everything the project needed for the register table entry along with the added bonus of user familiarity. In addition, the few followers of my publications will tell you that I have more than a passing interest in Excel. Of course it would have been possible to create the serial control within Excel, but the seed for this project is a dedicated test jig that runs on a PC that doesn't have Microsoft Office.

| Slave address (1 byte) | Function code (1 byte) | Data byte (function dependent) | | CRC (2 bytes) |
|---|---------------------------|-----------------------------------|---|---|
| Function code 01 _{HEX} Read coils | | | | |
| Function code 02 _{HEX} Read discrete inputs | | | | |
| Function code 03 _{HEX} Read holding registers | | | | |
| Function code 04 _{HEX} Read input registers | | | | |
| Function code 05 _{HEX} Write single coil | | Coil address 2 bytes | Data bytes FF00 _{HEX} to energize output, 0000 _{HEX} to de-energize output | |
| Function code 06 _{HEX} Write single register | | Register address 2 bytes | Data bytes 16-bit Data | |
| Function code 0F _{HEX} Write multiple coils | | Coil start address 2 bytes | Number of outputs 2 bytes | Data bytes Bits formatted into bytes |
| Function code 10 _{HEX} Write multiple registers | | Register start address 2 bytes | Data bytes 16-bit value for each register Number of bytes = number of registers x 2 | |

Figure 1—Take a look at the construction of the master transmission frame. The data field content depends on the function code. There are other codes in the Modbus specification, but I have not considered them in this article. All 16-bit addresses and data are sent with the most significant byte first. However, the CRC is transmitted with the least significant byte first.

MODBUS PROTOCOL

The Modbus serial protocol standard allows for the communications to be

in ASCII format or transmitted in 8-bit bytes (RTU format). The ASCII format is optional, but all Modbus units must be capable of supporting the RTU format. Thus, I am going to consider only the RTU format.

The serial port hardware interface has several parameters associated with it: the data rate, the number of stop bits, and parity. Confidence in the integrity of the message is provided by a 2-byte cyclic redundancy check (CRC) that is attached to the end of the transmission frame; so, strictly speaking, parity is not required.

The maximum length of a Modbus packet is 256 bytes. The specification requires 3.5 transmission bytes between Modbus packets. But in this example, the interframe delay is not an issue because we will be sending a single command and waiting for a response. User interaction is unlikely to be faster than the few milliseconds required.

The Modbus transaction consists of a master sending a packet to an addressed slave and then waiting for a response. When a slave receives a command, it responds, provided the address matches and the CRC is correct. If there is an error in the instruction, it responds with an exception message indicating the problem. In some cases it may take some time for the slave to complete the instruction. In this case, the exception code is an "acknowledge." In a normal operation, if the transmission fails either through a lack of response or an exception response, it is up to the master to act on this by reporting a problem and attempting a retransmission. Because this application is only intended for development, only single transactions are generated and monitored.

The generic transmission packet from the Modbus master is construct-

| Slave address (1 byte) | Function code (1 byte) | Data bytes (function dependent) in 2-byte multiples | | CRC (2 bytes) | | | | | | | | | | |
|--|---|---|--|------------------|--|---|---|--|--|---|--|--|--|--|
| <table border="1"> <tr> <td>Function code 01_{HEX} Read coils</td> <td>Number of bytes of data following</td> <td colspan="3">Data bytes Bits of data compressed into 8-bit bytes</td> </tr> <tr> <td>Function code 02_{HEX} Read discrete inputs</td> <td></td> <td colspan="3"></td> </tr> </table> | | | | | Function code 01 _{HEX} Read coils | Number of bytes of data following | Data bytes Bits of data compressed into 8-bit bytes | | | Function code 02 _{HEX} Read discrete inputs | | | | |
| Function code 01 _{HEX} Read coils | Number of bytes of data following | Data bytes Bits of data compressed into 8-bit bytes | | | | | | | | | | | | |
| Function code 02 _{HEX} Read discrete inputs | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>Function code 03_{HEX} Read holding registers</td> <td>Number of bytes of data following</td> <td colspan="3">Data bytes 16-bit values Number of bytes = number of registers x 2</td> </tr> <tr> <td>Function code 04_{HEX} Read input registers</td> <td></td> <td colspan="3"></td> </tr> </table> | | | | | Function code 03 _{HEX} Read holding registers | Number of bytes of data following | Data bytes 16-bit values Number of bytes = number of registers x 2 | | | Function code 04 _{HEX} Read input registers | | | | |
| Function code 03 _{HEX} Read holding registers | Number of bytes of data following | Data bytes 16-bit values Number of bytes = number of registers x 2 | | | | | | | | | | | | |
| Function code 04 _{HEX} Read input registers | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>Function code 05_{HEX} Write single coil</td> <td>Coil address 2 bytes</td> <td colspan="3">Data bytes FF00_{HEX} to energize output, 0000_{HEX} to de-energize output depending on master's command</td> </tr> </table> | | | | | Function code 05 _{HEX} Write single coil | Coil address 2 bytes | Data bytes FF00 _{HEX} to energize output, 0000 _{HEX} to de-energize output depending on master's command | | | | | | | |
| Function code 05 _{HEX} Write single coil | Coil address 2 bytes | Data bytes FF00 _{HEX} to energize output, 0000 _{HEX} to de-energize output depending on master's command | | | | | | | | | | | | |
| <table border="1"> <tr> <td>Function code 06_{HEX} Write single register</td> <td>Register address 2 bytes</td> <td colspan="3">Data bytes 16-bit Data (same as in master's command)</td> </tr> </table> | | | | | Function code 06 _{HEX} Write single register | Register address 2 bytes | Data bytes 16-bit Data (same as in master's command) | | | | | | | |
| Function code 06 _{HEX} Write single register | Register address 2 bytes | Data bytes 16-bit Data (same as in master's command) | | | | | | | | | | | | |
| <table border="1"> <tr> <td>Function code 0F_{HEX} Write multiple coils</td> <td>Register address 2 bytes</td> <td colspan="3">Data bytes Number of outputs 2 bytes</td> </tr> <tr> <td>Function code 10_{HEX} Write multiple registers</td> <td></td> <td colspan="3"></td> </tr> </table> | | | | | Function code 0F _{HEX} Write multiple coils | Register address 2 bytes | Data bytes Number of outputs 2 bytes | | | Function code 10 _{HEX} Write multiple registers | | | | |
| Function code 0F _{HEX} Write multiple coils | Register address 2 bytes | Data bytes Number of outputs 2 bytes | | | | | | | | | | | | |
| Function code 10 _{HEX} Write multiple registers | | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>Function codes 81_{HEX}, 82_{HEX}, 83_{HEX}, 84_{HEX}, 85_{HEX}, 86_{HEX}, 8F_{HEX}, 90_{HEX} MSB indicates Modbus slave exception to a particular function</td> <td>Exception code 01—Illegal function 02—Illegal data address 03—Illegal data value 05—Acknowledge</td> <td colspan="3"></td> </tr> </table> | | | | | Function codes 81 _{HEX} , 82 _{HEX} , 83 _{HEX} , 84 _{HEX} , 85 _{HEX} , 86 _{HEX} , 8F _{HEX} , 90 _{HEX} MSB indicates Modbus slave exception to a particular function | Exception code 01—Illegal function 02—Illegal data address 03—Illegal data value 05—Acknowledge | | | | | | | | |
| Function codes 81 _{HEX} , 82 _{HEX} , 83 _{HEX} , 84 _{HEX} , 85 _{HEX} , 86 _{HEX} , 8F _{HEX} , 90 _{HEX} MSB indicates Modbus slave exception to a particular function | Exception code 01—Illegal function 02—Illegal data address 03—Illegal data value 05—Acknowledge | | | | | | | | | | | | | |

Figure 2—This is the slave response packet. As in the master's transmission, the content of the data field is context-dependent. If the slave takes exception to the master's request, it issues an exception response, which it indicates by setting the most significant bit of the function code. Although the acknowledge is not (strictly speaking) an exception, it is treated in the same way. All 16-bit addresses and data are sent with the most significant byte first. However, the CRC is transmitted with the least significant byte first.

ed as shown in Figure 1. The response packet from the slave is shown in Figure 2. Refer to the documents listed in the Resources section of this article for more information about the different instructions and their responses, including a step-by-step description of

| Register range (decimal) | Modbus function access | Read single code | Read multiple code | Write single code | Write multiple code |
|-----------------------------|---|------------------|--------------------|-------------------|---------------------|
| 0–10000 | Coil (bit access) | 1 | 1 | 5 | 15 |
| 10001–30000 | Input status (read only) (bit access) | 2 | 2 | N/A | N/A |
| 30001–40000 | Input register (read only) (bit access) | 4 | 4 | N/A | N/A |
| 40001– | Holding register (bit access) | 3 | 3 | 6 | 16 |

Table 1—Check out the Modbus functional allocation. The Modbus address space is divided into single bit and word types and read only and read/write.

generating the CRC. I strongly recommend that you study the documents for a deeper understanding of the Modbus. For this application, I implemented the most common function codes: 1 to 6, 15, and 16 (all numbers are decimal unless otherwise indicated). The other codes are for more complex products. If you need them, you could simply use this program as a basis to implement the extra functionality.

MODBUS ADDRESSING

The different categories of the Modbus register space and their address allocations are shown in Table 1. Different function codes are used in order to differentiate between the register ranges. The master must select the correct function code based on the address being accessed. Modbus system designers typically arrange all the remote's locations in a single memory space. Depending on the function access, an offset would be added to the address.

Let's assume that there was a coil at address 1, an input status register at address 2, an input status location at address 3, and an input register at location 4. The aforementioned transformation would make the addresses 1, 10002, 40003, 30004. When the command is translated into

Photo 1—Take a look at the setup for the ADAM-4017+. Note the hidden lines from rows 17 to 206, which are blank and have been hidden to compress the image to show here. Read-only addresses are placed in read/write memory space.

the transmission frame format, the first address would be 0 (Modbus creates an offset of 1 from the official address to the accessed address) followed by 1, 2, and then 3. Remember that the function codes will be different.

Groups of similar locations are typically collected together to allow the use of the group functions that enable access to multiple locations of the same type with a single command. Thus, all coils are normally grouped together, all holding registers, and so forth. This convention does not actually appear in the Modbus specification and is not applied universally.

Products like the Advantech ADAM-4024 four-channel analog output module (one of the modules that I experimented with) allow the memory spaces to overlap since they use different function codes (see Table 2). The user interface presented in this article allows for this overlap, as you will see later. For development, I used members of the Advantech ADAM family of Modbus interface modules.

Table 3 shows the different types of modules included in this application.

EXECUTABLE FILE

For those of you who don't want to

| Address (decimal) | Description | Access |
|-------------------|--------------------------------------|------------|
| 1 | Digital input 0 (bit) | Read only |
| 2 | Digital input 1 (bit) | Read only |
| 3 | Digital input 2 (bit) | Read only |
| 4 | Digital input 3 (bit) | Read only |
| 40001 | D/A output 0 (16-bit word) | Read/write |
| 40002 | D/A output 1 (16-bit word) | Read/write |
| 40003 | D/A output 2 (16-bit word) | Read/write |
| 40004 | D/A output 3 (16-bit word) | Read/write |
| 40201 | Output 0 configuration (16-bit word) | Read/write |
| 40202 | Output 1 configuration (16-bit word) | Read/write |
| 40203 | Output 2 configuration (16-bit word) | Read/write |
| 40204 | Output 3 configuration (16-bit word) | Read/write |
| 40211 | Module name 1 | Read only |
| 40212 | Module name 2 | Read only |
| 40213 | Version 1 | Read only |
| 40214 | Version 2 | Read only |
| 40215 | Communications safety enable | Read only |
| 40216 | Communications safety flag | Read only |
| 40217 | Channel enable | Read/write |

Table 2—Here you see the Modbus register map of the ADAM-4024 analog output module. Note that it also has four digital inputs that overlap the D/A outputs. There are several read only registers in the 40000 register space. It doesn't really matter whether my interpretation of the Modbus convention is correct. You still have to be able to interface to this device.

get into the operational details of the Excel worksheet and the Visual Basic code, I supplied an executable file along with the ADAM1 worksheet and an Excel template in addition to some user instructions. They are in the ModbusExecutable.zip file posted on *Circuit Cellar's* FTP site.

Read the user's guide before installing the application. Keep in mind that I have not had many opportunities to test this with different versions of Windows and Excel, so there may be some compatibility issues.

EXCEL USER INTERFACE

Before you use the Excel files posted on the *Circuit Cellar* FTP site, you should open them with your version of Excel and save them. Some weird problems occurred between the Modbus test application and Excel when the files had been processed by a different version of Excel.

I decided to allow for a product family approach to the Excel interface so that a group of products could appear in a single workbook. Excel lends itself to tabbed operation, so each member of the family occupies a tab. In fact, the ADAM-4024 occupies two tabs because there is an overlap of the analog and digital address spaces (see Table 2).

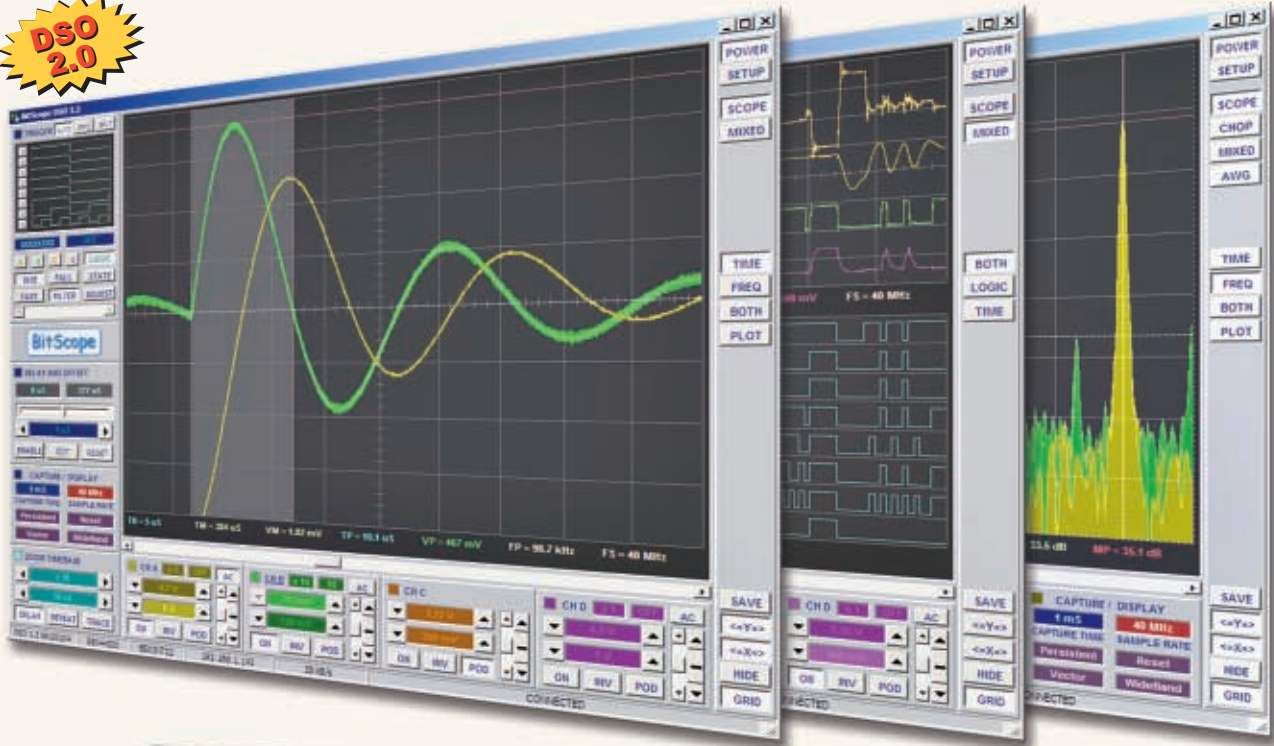
I allowed for 256 address spaces within a Modbus map. There are cases in which more are required and can be easily added by expanding the table. I should mention that the worksheets are protected (without a password) to prevent erroneous modifications to the formulas. Any changes to the worksheet like copying or writing to a protected cell requires the worksheet to be unprotected (click Tools, Protection, and Unprotect Sheet). After making modifications, I recommend that you protect the worksheet again.

To create a worksheet from a blank template (supplied with the files), consider Photo 1, which was derived from the ModbusTemplate.xls work-

BitScope PC Oscilloscopes & Analyzers

DSO Test Instrument Software for BitScope Mixed Signal Oscilloscopes

DSO 2.0



4 Channel BitScope



2 Channel BitScope



Pocket Analyzer

Digital Storage Oscilloscope

- ✓ Up to 4 analog channels using industry standard probes or POD connected analog inputs.

Mixed Signal Oscilloscope

- ✓ Capture and display up to 4 analog and 8 logic channels with sophisticated cross-triggers.

Spectrum Analyzer

- ✓ Integrated real-time spectrum analyzer for each analog channel with concurrent waveform display.

Logic Analyzer

- ✓ 8 logic, External Trigger and special purpose inputs to capture digital signals down to 25nS.

Data Recorder

- ✓ Record anything DSO can capture. Supports live data replay and display export.

Networking

- ✓ Flexible network connectivity supporting multi-scope operation, remote monitoring and data acquisition.

Data Export

- ✓ Export data with DSO using portable CSV files or use libraries to build custom BitScope solutions.

BitScope DSO Software for Windows and Linux

BitScope DSO is fast and intuitive multi-channel test and measurement software for your PC or notebook. Whether it's a digital scope, spectrum analyzer, mixed signal scope, logic analyzer, waveform generator or data recorder, BitScope DSO supports them all.

Capture deep buffer one-shots or display waveforms live just like an analog scope. Comprehensive test instrument integration means you can view the same data in different ways simultaneously at the click of a button.

DSO may even 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 other PCs as if a BitScope was locally connected.

BitScope DSO supports all current BitScope models, auto-configures when it connects and can manage multiple BitScopes concurrently. No manual setup is normally required. Data export is available for use with third party software tools and BitScope's networked data acquisition capabilities are fully supported.



www.bitscope.com

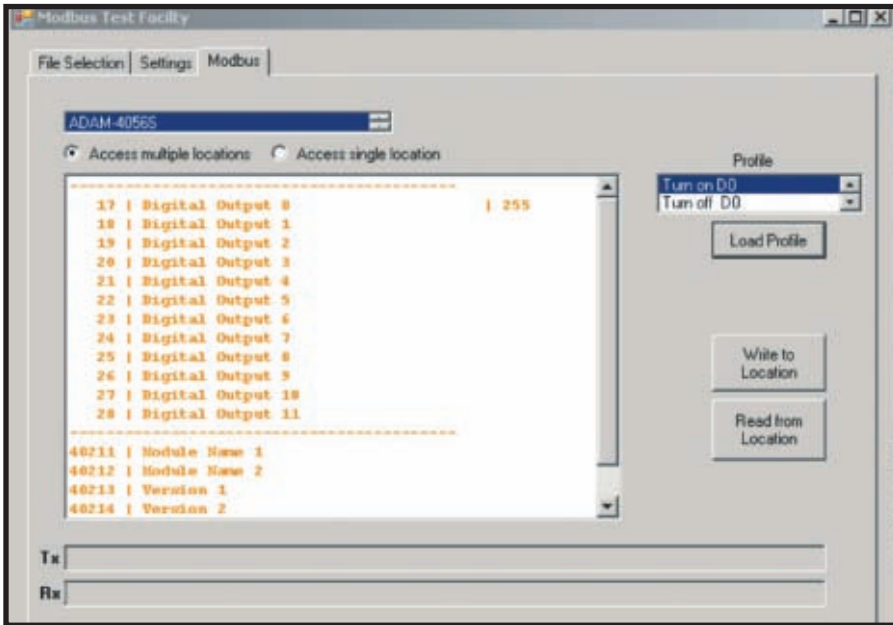


Photo 2—The user interface derives information from the Excel worksheet in Photo 3. Note how the address gaps have been compressed. The addresses and the descriptions have been imported directly from the Excel worksheet. In addition, I also clicked on Load Profile for Turn on D0. The value on the far right of the line for address 17 has been as a result of a Load Profile action.

book. You can add or remove worksheets (each represented by a tab) by right clicking on the tab and selecting the appropriate menu item. You should also rename the tab to the name of the product. This is the descriptor that will be loaded into the user interface as you can see in the list box in Photo 2 (just beneath the ADAM-4056S Modbus tab). If you add any worksheets, copy the range A1: S263 from the first sheet to the new sheets.

In an unchanged template, cell D1 is blank, as are columns C and D from row 8 and down. First, you enter the Module address in cell D1. Entering a description in column C, such as “Analog Input 1” is purely descriptive. It is helpful when selecting a location to access, but it has no influence on the results of the worksheet. This text appears in the large list box on the Modbus tab of the user interface, as seen in Photo 2, where it reads “Digital Input 0,” and so on.

For every valid address, you must fill in the Modbus address in the corresponding cell in column D. For a block of addresses, you don’t need to individually enter each address. You can use Excel’s autofill feature by filling in the first two addresses 40001

and 40002 and then clicking and dragging the autofill handle to get to 40008.

You can take a second approach: enter 40001 in cell D8 and then enter “=D8+1” in cell D9. You can then copy this cell down to D15. There are several other options. Imagine the work you would have to do if you were to use the Datagrid control in Visual Basic. You must enter only those addresses that you want the Visual Basic interface to recognize. Even if there are large gaps, as there are in this example, they must be filled in their correct location. I wrote the code in the Visual Basic user interface to allow for the compression of the display. An address is combined with the associated description to create a single line (see Photo 2). Also note the dashed lines (“-----”), which indicate the display compression.

Once you enter the address in the Excel worksheet, the interface type in column E automatically reflects a value

| Module identification | Description |
|-----------------------|---|
| ADAM-4017+ | Eight-channel analog input (16-bit ADC) |
| ADAM-4024 | Four-channel analog output (12-bit DAC) |
| | Four digital inputs |
| ADAM-4051 | 16-channel digital inputs |
| ADAM-4056S | 12-channel digital outputs |

Table 3—These Modbus modules are set up in ADAM1.xls.

of 1 to 4, where 1 is within the address space 1–10000, 2 is equivalent to the address space 10001–30000, 3 relates to address space 30001–40000, and 4 is of course for 40001 and up. If there is no address entry in column D, the corresponding cell in column E is blank. This is realized with the following formula in cell E8:

```
=IF(D8="", "", (IF(D8<10001, 1, IF(D8<30001, 2, IF(D8<40001, 3, 4))))))
```

This is copied to each of the cells in this column in the table.

The next column (F) generates the actual address that will be used in the Modbus message sent through the serial port. Cell F8 contains the formula:

```
=IF(E8="", "", CHOOSE(E8, D8-1, D8-10001, D8-30001, D8-40001))
```

Based on the value of a cell in column E, the associated cell in column F contains the address of column D modified for transmission.

The individual and block write and block read commands are derived in a similar fashion in columns G, H, and I. For instance, cell G8 contains:

```
=IF($E8="", "", CHOOSE($E8, 5, "n/a", "n/a", 6))
```

As you can see in cell D3 (f=COUNT(B:B)), there are a number of Excel formulas that count different things. However, I also wanted to know where the last entry in column D was in order to aid processing when I access the table from Visual Basic. Excel doesn’t have a formula to calculate that, so I wrote my own function (see Listing 1). You can find this in the Excel Visual Basic workspace accessed using <Alt>+<F11> within Excel.

Photo 3 shows the worksheet for a module with digital outputs. The setup procedure for the Excel worksheet is identical to the digital input in Photo 1. In order to allow certain predetermined actions, I provided for several profiles to store the associated output patterns. Simply add a name for the profile (without carriage returns, line feeds, etc.)

PROTEUS

ELECTRONIC DESIGN

Version 7 Available NOW!

- * New User Interface.
- * New Design Explorer.
- * New 3D Visualization.
- * New Simulation Advisor.
- * New Diagnostic Tools.
- * PIC24 Simulation Available.

FROM CONCEPT

TO COMPLETION

SCHEMATIC CAPTURE PROSPICE EMBEDDED SIMULATION PCB DESIGN

ISIS SCHEMATIC CAPTURE

A powerful capture package tailored for today's engineer and designed to allow rapid entry of complex schematics for simulation and PCB Layout.

PROSPICE MIXED MODE SIMULATOR

A customised implementation of the industry standard Berkeley SPICE 3F5 engine with extensive optimisations and enhancements for true mixed mode simulation and circuit animation.

VSM VIRTUAL SYSTEM MODELLING

The world's first and best schematic based microcontroller co-simulation software. Proteus VSM allows you to simulate the interaction between software running on a microcontroller and any analog or digital electronics connected to it. This streamlines the project lifecycle and obviates the need for expensive hardware analysis tools.

ARES PCB DESIGN

A modern and professional layout package which seamlessly integrates with the ISIS capture software. Features such as autoplacement and autorouting, interactive DRC and an intuitive interface all serve to maximise productivity and reduce time to market.

LABCENTER ELECTRONICS LTD.

A technology pioneer in the EDA industry since 1988.
Technical support direct from the program authors.
Flexible packages and pricing tailored to customer requirements.

R4 Systems Inc.
1100 Gorham St
Suite 11B-332.
Newmarket Ont.
Canada L3Y 8Y8

Authorized Distributor of Proteus
Other products include:
Proton + Development Suite
Discovering PIC Books and Kits
PIC Educational Courses

CONTACT US NOW

to discuss requirements or
request a **FREE** evaluation copy.

Toll Free 866.499.8184
Tel: 905.898.0665
Fax: 905.898.0683
email: info@r4systems.com
www.r4systems.com

*The premier conference and exhibition for PCB
engineering, design and manufacture professionals*

PCB DESIGN CONFERENCE WEST

CONFERENCE: MARCH 25-30, 2007

EXHIBITION: MARCH 27-28, 2007

Santa Clara Convention Center in Santa Clara, CA

Get ready for an amazing conference!

- Over 35 Professional Development and Technical Conference courses—43% of them new to PCB West!
- 14 "FREE Tuesday" courses—12 of them new!
- Keynote Address by Mentor Graphics' Henry Potts!
- Opening Night Extravaganza!
- Exhibitor Presentations, giveaways and more!

Download your copy of the 2007 conference brochure today at www.pcbwest.com.

Gold Sponsor:

EMA | Design Automation™

Media Sponsors:

PRINTED CIRCUIT
DESIGN & MANUFACTURE

CIRCUITS
ASSEMBLY

UP
Media Group

| Index | Item Description | Modbus Address | Type | Physical Address | Read | Individual Write | Block Write | Turn on D0 |
|-------|------------------|--------------------|-------|------------------|------|------------------|-------------|------------|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | 1 | | | | | | | |
| 9 | 2 | | | | | | | |
| 10 | | | | | | | | |
| 11 | 16 | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | 17 | Digital Output 0 | 17 | 1 | 16 | 1 | 5 | 15 |
| 15 | 18 | Digital Output 1 | 18 | 1 | 17 | 1 | 5 | 15 |
| 16 | 19 | Digital Output 2 | 19 | 1 | 18 | 1 | 5 | 15 |
| 17 | 20 | Digital Output 3 | 20 | 1 | 19 | 1 | 5 | 15 |
| 18 | 21 | Digital Output 4 | 21 | 1 | 20 | 1 | 5 | 15 |
| 19 | 22 | Digital Output 5 | 22 | 1 | 21 | 1 | 5 | 15 |
| 20 | 23 | Digital Output 6 | 23 | 1 | 22 | 1 | 5 | 15 |
| 21 | 24 | Digital Output 7 | 24 | 1 | 23 | 1 | 5 | 15 |
| 22 | 25 | Digital Output 8 | 25 | 1 | 24 | 1 | 5 | 15 |
| 23 | 26 | Digital Output 9 | 26 | 1 | 25 | 1 | 5 | 15 |
| 24 | 27 | Digital Output 10 | 27 | 1 | 26 | 1 | 5 | 15 |
| 25 | 28 | Digital Output 11 | 28 | 1 | 27 | 1 | 5 | 15 |
| 26 | 29 | | | | | | | |
| 27 | 210 | | | | | | | |
| 28 | 211 | Module Name 1 | 40211 | 4 | 210 | 3 | 6 | 16 |
| 29 | 212 | Module Name 2 | 40212 | 4 | 211 | 3 | 6 | 16 |
| 30 | 213 | Version 1 | 40213 | 4 | 212 | 3 | 6 | 16 |
| 31 | 214 | Version 2 | 40214 | 4 | 213 | 3 | 6 | 16 |
| 32 | 215 | Comm Safety Enable | 40215 | 4 | 214 | 3 | 6 | 16 |
| 33 | 216 | Comm Safety Flag | 40216 | 4 | 215 | 3 | 6 | 16 |
| 34 | 217 | | | | | | | |
| 35 | 218 | | | | | | | |

Photo 3—This is the worksheet for the ADAM-4056S, a digital output module. I hid some rows (37 to 216) again to compress this picture.

starting in cell J7 and on (see Photo 3). These will be shown in the user interface for user selection. The user interface program will import the associated inputs. If a cell is empty, no value is imported into the user interface. The profiles can be overlaid, as you will see later.

In the UI, if I select Turn On D0 and click Load Profile, followed by a click on Turn on D1 and a second click on Load Profile, then both values will be imported to the user interface and be able to be used for a block write. The 255 in the top right-hand corner of the large list box in Photo 2 is a result of a Load Profile of Turn on D0 in the Profile list box. You can have as many

profiles as you like. The number of profiles is maintained in the cell D5 that contains the formula:

$$=COUNTA(\$7:\$7) - 8$$

which counts all the nonblank entries in row 7 and removes the eight columns used for features other than profiles. In case you haven't guessed, Photo 2 shows the user interface with the selection of the worksheet in Figure 2.

Once you finish working on the worksheet, you should protect it again if you previously unprotected it. Click Tools, Protection, and Protect Sheet and then ensure that Select Unlocked

Cells is the only option checked.

It is important to note that the Visual Basic application opens a second copy of Excel. Any changes made to the workbook since the last save will not appear in the user interface. Any subsequent changes, no matter how often the workbook is saved, will not appear until the user reloads the workbook.

GAINING ACCESS

Up till now, all that I have discussed has been theory and preparation. The next step is to access the contents of the Excel worksheet from within Visual Basic. Unfortunately, I will have to leave you in suspense until next month, when I'll also describe how to send and receive Modbus messages from the PC. Stay tuned! ☒

Aubrey Kagan is a professional engineer with a B.S.E.E. from the Technion—Israel Institute of Technology and an M.B.A. from the University of the Witwatersrand. He works at Emphatec, a Toronto-based design house of industrial control interfaces and switch-mode power supplies. In addition to writing several articles for Circuit Cellar and having ideas published in other periodicals, Aubrey wrote Excel by Example: A Microsoft Excel Cookbook for Electronics Engineers (Newnes, 2004).

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

RESOURCES

Modbus-IDA, Modbus Application Protocol Specification, V1.1a, 2004, www.modbus.org/docs/Modbus_Application_Protocol_V1_1a.pdf.

Modbus.org, "MODBUS Over Serial Line: Specification & Implementation Guide," V1.0, 2002, www.modbus.org/docs/Modbus_over_serial_line_V1.pdf.

SOURCE

ADAM-4024 Analog output module
Advantech
www.advantech.com.tw

Listing 1—You can find the last entry in the column. This is accessed by the `=LastEntry($D:$D)` formula in cell D4. The function looks at each of the cells in column D, starting at the bottom and working its way up until it finds an entry. Because this function works on every page (the range is valid on all worksheets), the switch between worksheets can sometimes lead to an incorrect value in D4. The way to solve this is to add the *Calculate* procedure to the *SheetActivate* event as seen at the end of this listing. The upshot is that everything is recalculated at every tab selection change.

```
Dim iI As Integer
For iI = Range("d3") To 1 Step -1
    If Cells(iI + 7, 4).Value <> "" Then
        Exit For
    End If
Next iI
LastEntry = iI
End Function

Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    Calculate
End Sub
```



Build a PIC Platform

Microchip Technology's PIC18F97J60 microcontroller will be a great fit for many of your remote control and automation applications. But you might find it difficult to work with due to its TQFP packaging. Fred has the perfect solution: build up your own PIC18F97J60 platform.

The Microchip Technology PIC18F97J60's datasheet comprises 472 pages. If you read through the first couple of pages, you'll notice that the PIC18F97J60 actually comes in a trio of flavors. The PIC18F97J60 is the largest of the group, hosting 100 pins, 70 of which are capable of some kind of I/O operation. The PIC18F97J60's little brother, the PIC18F87J60, comes fitted in an 80-pin TQFP. The baby bear PIC18F67J60 is contained within the smallest 64-pin TQFP. The PIC18F87J60 offers 55 usable general-purpose I/O pins, while the smaller PIC18F67J60 weighs in with 39 free I/O pins.

Each member of the PIC18F97J60 family is stocked with 3,808 bytes of SRAM and an 8-KB chunk of Ethernet TX/RX buffer memory. Program flash memory capacity ranges from 64 KB in the PIC18F66J60, PIC18F86J60, and PIC18F96J60 to a maximum of 128 KB of program flash memory in the PIC18F67J60, PIC18F87J60, and PIC18F97J60. If 64 KB is not enough flash memory and 128 KB is way too much, you can opt for the PIC18F66J65, PIC18F86J65, or PIC18F96J65, all of which carry 96 KB of program flash memory.

For program flash memory or SRAM requirements outside of the standard PIC18F97J60 family limits, you can choose to hang some external memory from the pins of the PIC18F96J60, PIC18F96J65, or PIC18F97J60. The only other real advantage of using the high-end PIC18F97J60 part is an extra serial port and a few more ADC inputs.

There's one thing I haven't called

out about any of the PIC18F97J60 parts. Take a look at Photo 1. You can't plug any of them into a socket, because every PIC18F97J60 family member is packaged in a TQFP. That means you will just have to build up a PIC18F97J60 platform of your own. However, before you start the design and build process, let's find out why we want to do this.

PIC18F97J60

Designed as a logical extension of the Microchip Technology ENC28J60 stand-alone Ethernet engine, the PIC18F97J60 is a single-IC combination of the ENC28J60 and a PIC18 microcontroller. Everything you've come to know and love about PIC microcontrollers is part of the PIC18F97J60. As you would expect, the PIC18F97J60 parts can run a bit faster than most other PIC18 parts, with a maximum clock speed of 41.667 MHz.

The PIC18F97J60's main reason for existence is to solve some of the original ENC28J60's problems that stemmed from simple physics. There's only so much Ethernet data that you can pump over a SPI in a given amount of time. The PIC18F97J60 eliminated the SPI bottleneck by incorporating the ENC28J60 on-chip with the PIC18 microcontroller. In the case of the PIC18F97J60 family members, the integrated ENC28J60 engine's memory area and SRAM buffers are directly accessible by the PIC18F97J60's PIC18 microcontroller subsystems.

"Problem" may be a bad word, so

maybe I should use the term "design trade-off" instead. If you have designed around the ENC28J60 stand-alone part, you know that a logic level translation is required if you want to run some of the PIC microcontrollers as full-speed hosts to the ENC28J60. ENC28J60 devices at revision levels below B5 have a SPI clocking restriction that requires a minimum SPI clock speed. To meet the SPI clock speed requirement, many of the PIC microcontrollers must run at a voltage level that enables the PIC system clock to generate a SPI clock fast

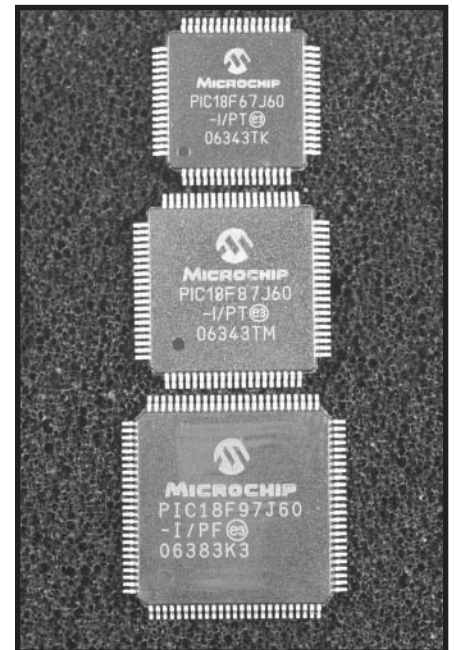


Photo 1—Here's a reconnaissance satellite view of all of the current PIC18F97J60 family members. I was tempted to design around the 64-pin part, because it is much easier to handle. However, that would not help those of you who may wish to use the extended functionality of the full-blown 100-pin PIC18F97J60.

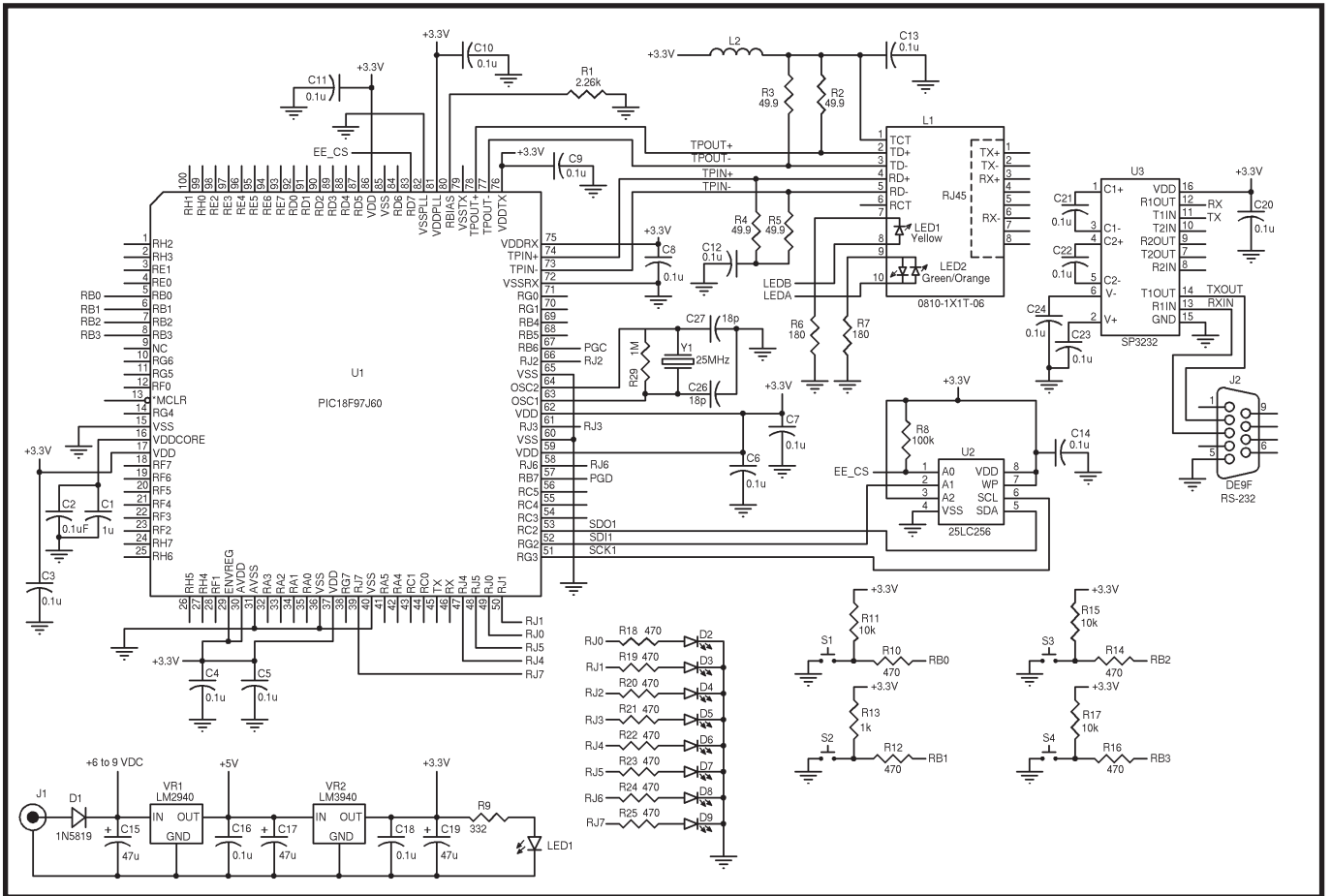


Figure 1—This is a boilerplate that I'm sure you'll be embellishing with your own set of peripheral devices.

enough to provide reliable data flow between the ENC28J60 SPI and the host PIC's SPI portal. Most of the current crop of standard PIC LF devices cannot acquire their full clock speeds at voltages less than 4.5 VDC. Since the ENC28J60 is a 3.3-VDC part with non-5-V-tolerant digital outputs, the ENC28J60 designer is forced into a design trade-off, which runs the PIC microcontroller at 5 VDC and buffers the ENC28J60's digital outputs with an external piece of buffer silicon.

All of the ENC28J60's SPI design trade-offs have been addressed in the new B5 ENC28J60 silicon. There is no longer a clocking minimum-speed requirement for the ENC28J60. However, if you choose to use a PIC microcontroller that cannot attain full-speed operation at 3.3 VDC, you will need to buffer the ENC28J60 digital outputs if you run the PIC host at 5 VDC.

The ultimate workaround to all of the ENC28J60 design trade-offs is to design your embedded Ethernet appli-

cation around the PIC18F97J60 family of devices. Don't get me wrong; there are still plenty of applications in which the new ENC28J60 silicon can provide a best-fit solution. Everyone in the world doesn't use PIC microcontrollers in embedded Ethernet applications. However, no matter which microcontroller host you prefer, it can be attached to an ENC28J60. You'll also need to incorporate an ENC28J60 if you are working on embedded Ethernet applications with the Microchip dsPIC line of microcontrollers. So, even though the PIC18F97J60's grass looks greener, there is a reason Microchip still makes the ENC28J60 available.

The PIC18F97J60 is actually very easy to bring up in the hardware sense. Figure 1 shows that the microcontroller personality of the PIC18F97J60 needs no more than any other PIC microcontroller requires. Each of the PIC18F97J60's power pins is bypassed with a 0.1- μ F capacitor. A standard PIC microcontroller crystal-controlled sys-

tem clock circuit is attached to the PIC18F97J60's OSC1 and OSC2 pins. The PIC18F97J60 requires a single 25-MHz crystal if the Ethernet functions of the PIC18F97J60 will be utilized. I decided to go ahead and include the 32-kHz clock circuitry as well. I've found that having a real-time clock available to the embedded Ethernet application is very handy.

There are five PIC18F97J60 oscillator options that are common across the PIC18F97J60 family of devices. PIC18F97J60 high-speed (HS) crystal modes include the use of either a standard crystal, as we have done in our PIC18F97J60 design, or a ceramic oscillator. To mix things up, you can enable the PIC18F97J60's phase locked loop (PLL) circuitry and invoke HSPLL mode. The PIC18F97J60's PLL subsystem consists of a 5 \times PLL that is sandwiched between a PLL prescaler and a PLL postscaler. Both the prescaler and postscaler can be configured to divide by two (1:2) or divide by three (1:3). The postscaler can also

be configured out of the circuit (1:1).

As I mentioned earlier, you must supply a 25-MHz clock source to the PIC18F97J60, because you are intending to use the PIC18F97J60's Ethernet capabilities. Disabling the 5× PLL and the postscaler will automatically disable the prescaler and provide a 25-MHz clock frequency to the PIC18F97J60's internals. To obtain the maximum clock frequency of 41.6667 MHz, you must enable the 5× PLL, disable the PLL postscaler, and configure the PLL prescaler for a 1:3 operation. If you do the simple math ($5 \times 25 \text{ MHz}/2$) with the prescaler set for 1:2 and the postscaler disabled, you'll see that the resultant frequency is well beyond the PIC18F97J60's operational capabilities.

The minimum clock frequency that can be sustained with the 5× PLL enabled is 3.8889 MHz (i.e., $5 \times (25 \text{ MHz}/3)/3$). Disabling the 5× PLL and maxing out the postscaler and prescaler values, yields a minimum clock frequency of 2.7778 MHz. External Clock (EC) and ECPLL modes, which accept the externally generated incoming raw-clock signal through the OSC1 pin, will work in the clock frequency domain exactly like the HS and HSPLL modes. The fifth PIC18F97J60 clocking mode is Internal Oscillator mode (INTRC). One of the five PIC18F97J60 clocking modes, which becomes the primary clock, is selected using the FOSCX trio of configuration bits. All of the PIC18F97J60's configurable oscillator components (5× PLL, prescaler and postscaler) are set up by the bits you code into the PIC18F97J60's OSC-TUNE register.

There's nothing new in the PIC18F97J60's MCLR circuitry. The PIC18F97J60's ICSP programming/debugging interface is no different than any other PIC. Outside of the PIC18F97J60's Ethernet interface, there is only one other design that you must take care of. The PIC18F97J60 contains an internal 2.5-VDC voltage regulator to provide power to the PIC18F97J60's microcontroller core from the PIC18F97J60's 3.3-VDC V_{DD} supply. The PIC18F97J60's internal 2.5-VDC voltage regulator is controlled by the ENVREG pin.

To enable the PIC18F97J60's internal 2.5-VDC voltage regulator, you must tie the ENVREG pin to V_{DD} . Otherwise, you must provide a source of power for the PIC18F97J60's core externally. Since you will be running the PIC18F97J60 at 3.3 VDC, I prefer tying the ENVREG pin, a 1- μF filter capacitor, and a 0.1- μF bypass capacitor to the PIC18F97J60's V_{CAP} pin over the incorporation of an external 2.5-VDC power supply circuit.

To keep the cost of the project down, the PIC18F97J60 and associated circuitry will be mounted on a simple double-sided PCB. When I do these types of projects, I try to put myself in the reader's shoes. I try to include enough flexibility in the project to allow the reader to be able to adapt the project to his or her needs. My original idea was to include pads for both the 64- and 80-pin variants of the PIC18F97J60 on the same PCB. That would have allowed the reader to use either of the parts and it would have eliminated the need to produce multiple PCBs for each PIC18F97J60 family device.

If you've ever played around with 64- and 80-pin TQFP pad layouts, you know that you can place the 64-pin pad layout neatly inside of the 80-pin pad layout. That works only if both the 64- and 80-pin TQFP pads are pitched identically. All of the PIC18F97J60 family members' pads are pitched identically at 0.5 mm. The difference in the 64- and 80-pin TQFP packages is simply four extra pins per side. If you were laying down a standard PIC microcontroller in the 80-pin package and did not hardwire any of the I/O, you could easily place a 64-pin pad package within the 80-pin pad package and just ignore the two general-purpose I/O pins at each flat corner when the smaller package was installed.

Unfortunately, in the case of the PIC18F97J60, the Ethernet differential inputs and outputs ride on two of the corners in the outermost pin positions. The PIC18F67J60's 64-pin package's Ethernet-differential pins shift into the general-purpose I/O pins of the PIC18F87J60 80-pin package. That alone should have been enough to squash the dual-IC pad idea for the PIC18F97J60 project board. Of course,

thinking big, I decided to try to lay the PIC18F87J60's 80-pin TQFP pads down inside of the PIC18F97J60's 100-pin TQFP-pad layout. That too was a "no go," as the PIC18F97J60 and the PIC18F87J60 don't match up in other places as well. They also have the same incorrect transposition of the Ethernet differential signals over general-purpose I/O signals between the intertwined pair of pad layouts. So, I scrapped the idea of multiplexing the PIC18F97J60 family of devices on a single PCB.

After a short moment of reflection, I decided that if I wanted to design for the PIC18F67J60 64-pin part or the PIC18F87J60 80-pin part, having the PIC18F97J60 100-pin part on the board would work out just fine. The PIC18F97J60 designer does not have to use the capability of the 100-pin part that does not exist in the lower-pin-count devices. Thus, the 100-pin PIC18F97J60 was in and the PIC18F97J60 device family pad multiplexing was out.

The PIC18F97J60 Ethernet interface circuitry is identical to that of the ENC28J60. The PIC18F97J60 Ethernet interface design is built around the Bel Fuse BM0810-1X1T-06 10/100Base-TX AutoMDIX belMag, which includes integral indicator LEDs. The LEDs are simply indicators as far as the PIC18F97J60 is concerned. Remember: the ENC28J60 uses the indicator LEDs to determine portions of its duplex configuration at power-up. The PIC18F97J60 multiplexes the Ethernet indicator LEDs with general-purpose I/O pins on PORTA and you can forego the Ethernet indicator functions and use the PORTA, RA0, and RA1 pins for I/O purposes.

Another difference is the external bias resistor. For the PIC18F97J60, it is specified as 2.26 k Ω . The value of the external bias resistor varies from revision level to revision level on ENC28J60 parts. The bias resistor is essential to the operation of the PHY module's internal analog circuitry and it influences the TPOUT signal amplitude. Messing with the value of the bias resistor can potentially cause the Ethernet transmit waveform to violate the IEEE 802.3 specification. So, you want to use what Microchip Technology specifies.

I've pretty much nit-picked all of the PIC18F97J60 hardware particulars, as it pertains to putting things on a PCB with solder. So, let's turn our attention to the rest of the supporting circuitry.

ESSENTIALS

Some of you may remember that some time ago I built an ASIX Ethernet NIC development board that used 0603 passive devices. I swore that I would never ever again put my eyes and hands into so much trouble. However, I'm finding that I can squeeze 0603 packages into tight places where the 0805 just can't go. So, here we are, back in 0603 land.

The driving force behind the essential supporting circuitry is the firmware that will run on the PIC18F97J60. Reinventing the wheel is not something any of us want to do. Writing PIC18F97J60 driver code from scratch

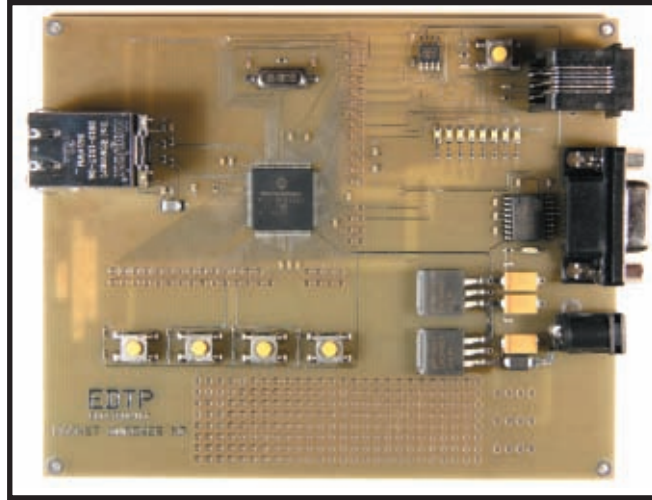


Photo 2—There's nothing more satisfying than bringing up a design from fiberglass and copper. I've included enough accessibility to the PIC18F97J60 to allow you to write your own drivers and application or use the prefab Microchip TCP/IP stack.

is effectively like chiseling on a round rock. A complete TCP/IP stack for the PIC18F97J60 is free on Microchip's web site. All you have to do is conform to some of the hardware conventions that are specified in the Microchip TCP/IP stack's compiler.h file.

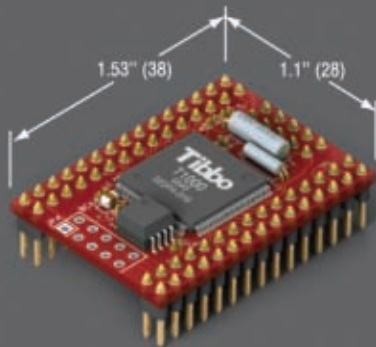
An RS-232 serial port is a must-have. The MAX3232 can be had as an

STMicroelectronics or Sipex part, which makes it easy to acquire from many sources in this magazine. Five 0.1- μ F capacitors plus one of the 3232 variants are all it takes to put the PIC18F97J60's serial port in place. I also wrestled with whether or not to replace the traditional nine-pin shell connector with header pins. I hate making up little cables for this and that, and I'll bet you do too. The nine-pin shell connector for the RS-232 port stays. I also went with the standard RJ-12 jack for the ICSP portal as well. That allows me to use the Microchip MPLAB ICD 2 right out of the box.

I have a lot of experience providing embedded Ethernet devices via my online store. I've noticed that many of you have the need for HTTP capability. The new Microchip TCP/IP stack has enhanced its HTTP firmware. To take full advantage of the TCP/IP

Tibbo
TECHNOLOGY

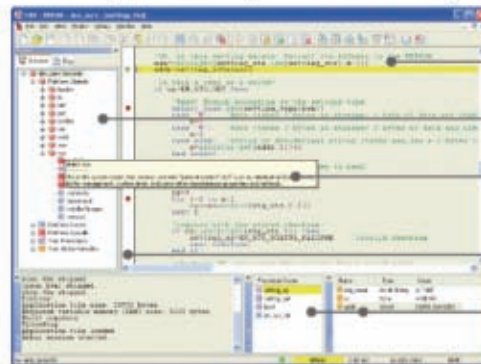
Build your next automation project around our EM1000 BASIC-programmable Embedded Module



- 50 MIPS CPU
- 100BaseT Ethernet port
- 512K flash disk
- 4x high-speed UARTs
- High-speed parallel slave port
- Real-time clock with backup power
- 49x general-purpose I/O lines
- Development kit available (EM1000-SK)

- Programmable – in BASIC!
- Optimized for real-time applications
- Rich object set
- Built-in webserver
- Event-driven operation
- Sophisticated development environment supports cross-debugging (no ICE needed)

Code and debug your Tibbo BASIC application using Tibbo Integrated Development Environment (TIDE) software



- Write in familiar BASIC language
- Inspect objects, procedures, and variables
- Code faster with auto-completion and code hints
- Set breakpoints, execute step-by-step, etc.
- Monitor the state of variables and stack

web: www.tibbo.com email: sales@tibbo.com

PC/104 Single Board Computers

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 256 M (with ARM Tool Chain), USB WiFi

200 MHz ARM9
Power as low as 1/4 Watt

- 5 boards, over 2000 configurations
 - Fanless, no heat sink
 - SDRAM - up to 128MB
 - 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
 - Linux, Real Time extension, NetBSD
- \$99**
qty 100
- \$129**
qty 1
- NEW!**
▪ SD card option
▪ VGA video option

Off-the-Shelf Solutions ready to design into
your project using DOS development tools



TS-5600 Shown with
optional flash modules,
A/D, RS-485 and
Merlin cellular modem

options include:
RS-485 Half and Full Duplex, A/D Converter up to 8 Channels at 12 bits, DAC up to 2 Channels at 12 bits, Extended Temperature

133 MHz 586

- Power as low as 800mA
 - Fanless, no heat sink
 - SDRAM - up to 64MB
 - COM Ports - up to 4 ports
 - Ethernet Ports
 - DIO Channels - up to 40
 - PCMCIA II adaptor
 - Compact Flash adaptor
 - USB Ports (Except on TS-5300)
- \$229**
qty 100
- \$259**
qty 1

see our website for 33 MHz 386 configurations

- 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

New Products and PC/104 Peripherals

Tiny WiFi Controller boots Linux in 1.1 seconds



\$249
qty 1

- 200 MHz ARM9
- Up to 128MB Flash
- Up to 128M SDRAM
- 802.11g WiFi
- SD Flash Card socket
- 1 external USB port
- 1 10/100 Ethernet
- 3 TTL serial ports

▪ Rugged aluminum enclosure
measures 1.1" x 4.9" x 3.1"

Intelligent Battery Back-up

NEW!

\$119
qty 1

- Run your system for days
with no external power source



| | |
|-----------------------------|--|
| NEW! ZigBee Wireless | low power wireless, simple serial interface, range up to 1 mile |
| Modems | 33.6K baud, 56K baud, AT commands, caller ID, cellular using GSM and CDMA technologies |
| Non-volatile Memory | up to 2MB, 10 year lithium battery |
| Serial Ports | up to 4 serial ports with optional RS-485, opto-isolated available |
| 12 bit A/D, DAC | 8 channel 12-bit A/D converter, optional 2 channel 12-bit DAC, A/D jumpered for 0-2.5V, 0-10V or 0-20mA |
| CAN Bus Controller | Philips SJA1000, opto-isolated, up to 1 megabit/sec selectable termination resistor, Ocera Linux driver |
| 64 Digital I/O | 32 inputs, 32 outputs, 200 mA drive, optional 512 Kbyte or 1 MB battery-backed SRAM, stack up to four boards, RoHS compliant |

see our website for more boards and option details



Technologic
S Y S T E M S

We use our stuff.

Visit our TS-7200 powered website at
www.embeddedARM.com

stack firmware, you'll need to add a SPI-driven EEPROM in the form of a Microchip 25LC256. The EEPROM can be used to hold web pages. If HTTP is not in your application mix, the EEPROM can be used to store anything you want.

Every device on the PIC18F97J60 project board is powered by 3.3 VDC. However, since the PIC18F97J60's inputs are 5-V tolerant and you may need to use a 5-VDC-powered device, I threw in a dual-voltage power supply to

accommodate any 5-VDC devices you may need to incorporate into your design.

There are some portions of the Microchip TCP/IP stack that require you to enter a function block by pressing a button attached to PORTB on the PIC18F97J60. In return, the function block may return a status via the RS-232 port or an LED. So, to make sure you can fully use these functions, I added some blinking lights and push buttons to the design. I recommend

reviewing the Microchip TCP/IP Stack Version Log to get the lowdown on what the new version of the TCP/IP stack will do.

A NEW GENERATION

Folks who build boats like to name them. I build embedded Ethernet devices and I like to name them as well. So, the PIC18F97J60 project board from this moment on will be known as the "Packet Whacker NG" (see Photo 2). NG is short for "new generation."

The Packet Whacker NG is very easy to build. You may think that the 100-pin PIC18F97J60 will present an assembly problem. However, with the right soldering tools, you can mount the PIC18F97J60 by hand. If hand soldering fine-pitch TQFP parts is not in your job description, an assembled Packet Whacker NG can be had at www.edtp.com. I'll also kit the Whacker NG for those of you who have the tools to handle the mounting of the PIC18F97J60. Once you see the Packet Whacker NG TCP/IP stack's activity light blink, you'll realize that the Packet Whacker NG isn't complicated, it's embedded. ☺

Fred Eady (fred@edtp.com) has more than 20 years of experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design and communications.

RESOURCE

Microchip Technology, Inc., "PIC18F97J60 Family Datasheet," DS39762A, 2006, ww1.microchip.com/downloads/en/DeviceDoc/39762a.pdf.

SOURCES

Packet Whacker NG
EDTP Electronics, Inc.
www.edtp.com

Microchip TCP/IP Stack, PIC18F97J60 family of Ethernet enabled microcontrollers, and MPLAB ICD 2
Microchip Technology, Inc.
www.microchip.com

Professional Features – Exceptional Price

34 Channels sampled at 500 MHz

Sophisticated Multi-level Triggering

Transitional Sampling / Timing and State

**Connect this indispensable tool to your PC's
USB 1.1 or 2.0 port and watch it pay for itself within hours!**

- 500 MHz Sampling / Timing Mode (Internal clock)
- 200 MHz Sampling / State Mode (External clock)
- Multi-level Triggering on Edge, Pattern, Event Count, Group Magnitude/Range, Duration etc.
- Real-Time Hardware Sample Compression
- Qualified (Gated) State Mode Sampling
- Interpreters for I²C, SPI and RS232
- Integrated 300 MHz Frequency Counter
- +6V to -6V Adjustable Logic Threshold supports virtually all logic families
- Full version of software free to download
- Mictor adapter available

www.pcTestInstruments.com

Visit our website for screenshots, specifications and to download the easy-to-use software.

Intronix Test Instruments, Inc.
Tel: (602) 493-0674 • Fax: (602) 493-2258
www.pcTestInstruments.com



Servo Control

First used in remote-controlled airplanes, servo motors are not just built for flight. Dale explains how he used them in custom projects and award-winning robots.

From flying model airplanes to driving miniature robots, hobby servo motors have many uses, and they are simple to control with just about any microcontroller. In this article, I'll describe the basics of servo control and give you detailed examples of how I have used them in successful commercial designs, custom projects, and even award-winning robots.

Originally designed for actuating control surfaces on remote-controlled model aircraft, hobby servo motors have many applications in automation, robotics, and entertainment applications. They are available in many sizes, from many manufacturers, and they all operate in basically the same way. Reliability, low cost, and standardized packaging are the driving factors behind their success, and make these servos winning choices for a variety of other small motion control projects.

In May 2003, I made a technical presentation for the Dallas Personal Robotics Group (DPRG) on using low-cost hobby servo motors in robotics. I also presented the group with a method of "hacking" them for continuous rotation as drive mechanisms. I prepared a detailed "How To" document based on that presentation and it has been published online at www.dprg.org/projects/2003-05a. It goes into general detail about how the servo motors work and specifically how to modify one particular model, the TS-53 from Tower Hobbies, for continuous rotation. Later in this article, I will explain more about continuous rotation.

WHAT'S INSIDE?

There's much more than you might suspect inside that small plastic box. There would have to be a motor, naturally, some gears, and maybe some bearings

in the more rugged models. Stopping there would leave you with a very useful subassembly, but you'd have to design and build your own motor driver circuitry and some sort of control function.

Let's all wish really hard and poof! A magic chip arrives that does all that for us and does it really well. The chip, an integral part of every hobby servo, is electrically connected to a potentiometer that is mechanically connected to the output shaft and provides position feedback. This feedback along with motor back-EMF forms a control system that gives proportional error correction, thereby driving the motor to the desired position. If the motor is a little to the left of where it is supposed to be, the control system drives it a little bit back to the right until it gets close enough and stops. If it is a long way from being correctly aligned, it gives a proportionally greater push. Now you see why it is called a servomechanism, servo motor, or just "servo," for short.

Most common servos are powered by a small DC voltage, which was traditionally supplied by rechargeable batteries. Almost all servos are designed to operate from 4.8 to 6 V, corresponding to four- or five-cell NiCd or NiMH battery packs. Some higher-performance exotics are happy with higher voltages. The current draw can be surprisingly high, especially when a motor is stalled and can easily exceed one or more amps in very bad situations. A positive temperature coefficient (PTC) thermistor, which is also called a "resettable fuse," in series with the power lead can be a lifesaver in an unexpected or extended stall.

THE CONTROL SIGNAL

Let's look at a typical hobby servo,

right out of the box and before we "improve" it in any way. There are almost always three wires trailing away from the body of the servo. These provide power, the control signal input, and a ground line common to both. We've already covered the power requirements, leaving only the control signal to describe in detail. We will get to that shortly, I promise.

As shipped, the servo is mechanically constrained to rotate approximately 180° and no further. The normal range of operation is typically only 90°. Various attachments to the output shaft, which are also called "horns," are normally linked to wire rods and linkages that actuate various control surfaces of model airplanes, steering mechanisms of wheeled land vehicles, or even rudders of model ships. Just a little move this way or that is all that is required. Several servos are usually used in conjunction with multichannel radio units to accomplish the complete control apparatus. If your motion-control requirement is for a short, fixed amount of motion along a curve or a linear range, almost any servo can be used as is. But continuous rotation can only be achieved by physically modifying the servo, which most certainly will void any manufacturer's warranty.

To control the position of the servo-output shaft, a pulse signal is sent to the control input line at a regular interval. The width of the pulse determines the desired position of the shaft. The standard specification is for the pulse width to vary between 1 and 2 ms, with a resulting rotation to the right and left, respectively. This range usually represents a 90° arc. Most servos will also accommodate signals outside this range, up to the mechanical limits of the

motor (i.e., approximately 180°). A pulse width of 1.5 ms should put the output shaft near the center of its travel.

The interval between the pulses is traditionally around 50 Hz, but it is not critical. Many remote-control radio systems multiplex up to seven or eight servo control signals on a single carrier. If the interval is too short, the servos can start to chatter. Too long an interval can cause the servo to “go limp” between updates. A properly timed signal causes the servo to lock to the desired position and hold on tenaciously.

COMPUTER CONTROL

Generating an appropriate servo control signal with a microcontroller is not difficult. Using more advanced peripheral hardware, such as timers and PWM, makes it trivial.

The first example uses an Atmel ATmega8, which is a low-cost part based on the AVR RISC architecture. Any other AVR with an available 16-bit timer will work identically. Initializing the timer to the proper waveform mode, which can be either “fast PWM” or “phase and frequency correct PWM,” tells the appropriate output pin to “clear on match, reset on overflow,” and then set the PWM period to 20 ms based on the system clock and the timer prescaler used. That sounds like a lot of work, but it boils down to very little actual code (see Listing 1).

This works for a system running at 8 MHz and provides two channels of PWM signal for controlling two separate servos. To adjust the position of the servos, write to the PWM compare registers with the desired pulse width in microseconds and then go do more important things. The PWM peripheral hardware will do everything else. No interrupts or further updates are required.

```
OCR1A = 1000; //servo A = 1ms, move right
OCR1B = 2000; //servo B = 2ms, move left
```

This implementation gives positional granularity of 0.1% or 1,000 steps within the normal 90° range of motion. Expanding the range of 1 to 2 ms to 0.5 to 2.5 ms will increase the range of motion to 180°, or 2,000 steps, given the previous example. Note that not all servos can reach these mechanical limits

Listing 1—This C code snippet initializes the ATmega8's Timer/Counter1 for PWM duty using Fast PWM mode and configures the I/O ports as outputs. Servo output pulse widths can now be indicated using microsecond granularity, giving 1,000 steps of positional information in the standard 90° range of motion.

```
DDRD = 1<<PD5 | 1<<PD4;
TCCR1A = 1<<COM1A1 | 0<<COM1A0 | 1<<COM1B1 | 0<<COM1B0 | 0<<FOC1A
        | 0<<FOC1B | 1<<WGM11 | 0<<WGM10;
TCCR1B = 0<<ICNC1 | 0<<ICES1 | 1<<WGM13 | 1<<WGM12 | 0<<CS12
        | 1<<CS11 | 0<<CS10;
ICR1 = 20000;

DDRD = 0x30; // Configure OC1A, OC1B as outputs
TCCR1A = 0xa3; // Clear outputs on match, reset on overflow
TCCR1B = 0x1a; // Fast PWM mode, prescaler = 8
ICR1 = 20000; // Set PWM period to 20 ms
```

and could potentially develop a mechanical fault if forced beyond the extremes.

What if you have more servos to control than available PWM channels? Have no fear. I have a trick or two up my sleeve for that very situation. Shift a single digital one through an 8-bit shift register whose outputs are connected to as many as eight servos. The shift register can be in external hardware with only a clock and a data line coming from the microcontroller, or it can be implemented in software by using a hardware output port. A timer interrupt begins the sequence based on the servo update frequency. Another timer is preset for the width of the first servo pulse and when it expires a pulse is sent to the shift register, shifting the one down through the bits. The next servo's pulse-width is programmed into the timer and the process continues until all eight positions have been output. This is a much simpler algorithm for multiplexing servos than turning them all on at once and updating them all simultaneously. It also mimics the method used to transmit up to eight servo positions over a radio link. I used this method to control the two servo drive motors in my robot, “The Purple Prowler,” which won three awards in the DPRG Roborama 2005b: First Place in Quick Trip and T-Time and Second Place in Line Following. The additional servo outputs controlled a gripper mechanism for collecting cans, but it has never been as successful at that task as fellow DPRG member David Anderson's talented robot, SR04.

CONTINUOUS ROTATION

Most, but not all, servos can be mod-

ified for continuous rotation. Some vendors, such as Gordon McComb's Budget Robotics, sell premodified servos for a very reasonable premium. The basic alteration involves removing the mechanical connection to the position feedback potentiometer, effectively removing the rotational constraint, and either removing the potentiometer from the circuit or calibrating it so that motion correction stops when the null spot is reached. The null point is reached when the input control signal's pulse width is exactly 1.5 ms long.

The servo doesn't know it has been snipped and tries to faithfully fulfill its duties as best it knows how. This now consists of traveling to the left or the right until the input control signal jibes with the reported position, as revealed by the now-disconnected and centered feedback potentiometer. In some cases this will never happen and the servo goes merrily along, spinning forever, which is exactly what we want it to do. To cause the servo to cease spinning, we can either send out a control pulse that corresponds with the null setting of the position feedback potentiometer or simply stop sending pulses. The major difference between the two techniques is that sending the null position pulse width will cause the motor to hold its position if deflected, which is a form of braking, and cessation of control pulses will cause the motor to go limp or coast.

THE SERVO TESTER

In describing the hacking procedure to the DPRG, it occurred to me that to properly set the feedback potentiometer to a useful null position

would require an accurate reference signal. I suggested a simple “servo tester” circuit be built using a low-pin-count microcontroller, such as the Atmel ATtiny11, and I offered to make it available to DPRG members. Thus the Servo Tester Mark I was born (see Photo 1).

I hand-crafted several of these units using homemade PCBs and I distributed them to the servo hackers in the club. The main feature was a Center button that would reset the microcontroller to emit a calibrated 1.5-ms pulse. During the modification process and with the victim servo’s insides hanging out, the feedback potentiometer was adjusted until the motor stopped, indicating the precise center of the range. The prototype units also sported Left and Right controls that moved the servo in each direction. These would let the user test the operation of both hacked and unmodified servos. The ATtiny11 has a timer peripheral but lacks a PWM subsystem, so the output pulses were generated in code and not exactly certification-lab quality.

THE SERVO TESTER II

The Mark I became popular enough to deserve a second thinking about and this led me to develop the artfully named Servo Tester II (see Photo 2). As its name might suggest, it had two of just about everything.

I based the system on the Atmel ATtiny26, a 20-pin AVR microcontroller with a versatile PWM subsystem. Since two channels of PWM outputs were available, two distinct servo channels were included. Each had two connectors, so that a total of four servos could be attached at once. I don’t know if anyone besides me has ever found a use for the “four-servo” configuration, but it’s there if anyone ever thinks of a good application. The

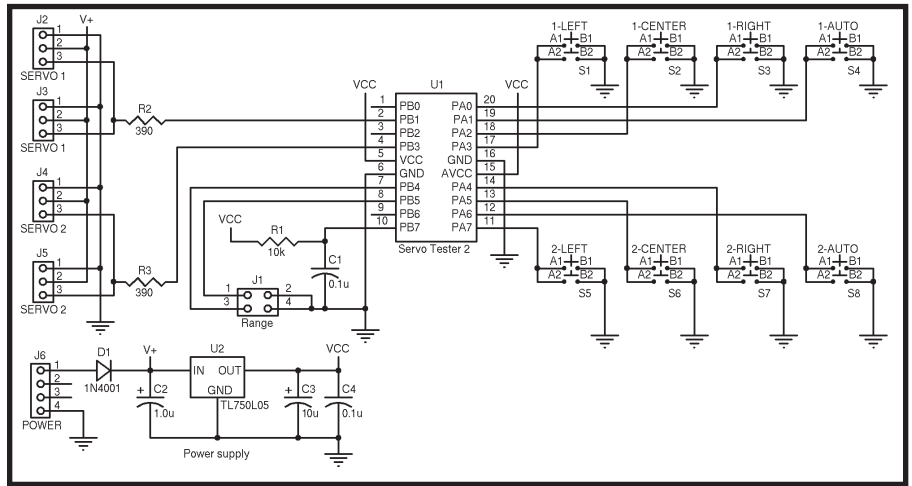


Figure 1—Using the versatile PWM peripheral subsystem of the ATtiny26, the Servo Tester II provides two channels of rock-steady servo control pulses. Each channel can be moved left, right, or back to the center, as well as set to Auto Sweep mode.

system is shown in Figure 1. The full source code is available on the *Circuit Cellar* FTP site. Assembled and tested versions are available from several retailers.

One nice feature of the Servo Tester II is the Auto Sweep mode, where the servo is exercised back and forth continuously. Jumpers enable operation in normal and extended range of motion modes. The Center button survives from its predecessor and performs the same function.

The ATtiny26 does not have the 16-bit timer resolution available to the ATmega8, which was used in the previous example’s code. It turns out that the humble 8-bit timer is perfectly capable of putting out precise and rock-steady servo control pulses, albeit at a lower granularity. Since the update interval period is reduced from 20,000 to 256 counts, a corresponding increase in step size is realized. The total range of possible positions declines from 1,000 in normal range to 17. This is plenty for the Servo Tester II.

SPEED CONTROL

One interesting characteristic of

hacked servos is that the speed of rotation is directly, if not linearly, proportional to the variance of the control signal from the center null position. If a properly calibrated hacked servo is given a series of pulses of exactly 1.5 ms, it doesn’t move and resists strenuously if forced off its mark. Should the pulse width widen ever so slightly, the output shaft will begin to rotate slowly in a counter-clockwise direction, as seen from the face of the motor. Correspondingly, a narrower pulse width will cause the shaft to begin to rotate clockwise. The farther from the middle position it gets, the faster the output shaft will rotate, up to a point about a quarter of the way from each limit, where full speed is achieved. This varies nonlinearly with different models of servos. Without some sort of velocity feedback, however, it is impossible to determine or predict the exact speed.

ROBO SERVO

Servos have many uses in robotics, which is one of my areas of interest. Unmodified servos are excellent for moving arms and actuating grippers, while continuous rotation servos are perfect for small robot drive wheels. Even a robot of little intelligence can drive around quite elegantly using servos.

I put this to the test in the fall of 2006, while preparing an entry for the DPRG Roborama 2006b, a series of robotics contests held at the Museum of Nature and Science in Dallas, Texas.

Listing 2—This is a very short but complete PBASIC program for the BASIC Stamp BS2 that drives a two-wheeled robot in a forward direction. Variations in motor characteristics, wheel diameter, and battery voltage will conspire against linear perfection.

```
main: PAUSE 16
PULSOUT 15, 250 ' left wheel counter-clockwise (< 1 ms)
PULSOUT 14, 1500 ' right wheel clockwise (> 2 ms)
GOTO main
```

| Value | Pulse length | Resulting motion |
|-------|----------------|------------------|
| 1 | Short (< 1 ms) | Clockwise |
| 0 | None | Stop |
| -1 | Long (> 2 ms) | Counterclockwise |

Table 1—Simple motion control for driving small robots can be implemented using these values to represent forward, reverse, and stop. Servos can accept control signals that are significantly outside their normal operational range. This characteristic makes it very easy to control them using almost any microcontroller.

Using only donated parts from various DPRG supporters (including Jon Williams formerly of Parallax and now EFX-TEK, Mike Dodson of Modern Assemblies, and others), I put together a theoretically simple robot. I used a prototype Parallax robot chassis, the venerable Board of Education sporting a vintage BS2-IC, some bumper switches, and a line sensor of my own devising. I originally wanted to prove some mathematical concepts concerning navigation using differentially configured drive systems, but I got caught up in the competitive whirlwind that happens before every big robot contest at the DPRG.

Now that the trophies are on the mantle (yes, the humble DPRG Yellow Robot placed in three of the four competitions in which it competed), I can reveal some of the surprisingly simple techniques used to run the servos with no sophisticated timing capabilities available whatsoever.

KEEP IT SIMPLE

First, let's greatly oversimplify the requirements. I find it always helps to implement expectation-management practices early on in a project. Keep those expectations down there—way down. Now for the most part, you can assume that the robot is either moving or not. If it is moving, it is going straight ahead, backwards, or spinning left or right in place. This requires three speeds for each of the two motors: full forward, full reverse, and a complete stop. There are logically nine possible combinations, four of which involve pivoting on one wheel.

Starting with the simplest option, the famous play dead trick, recall that a servo will stop doing anything if you cease to provide control pulses. By not emitting any sort of signal, you are telling the servo to stop. While not exactly mandat-

ing a Velocity Equals Zero command, you will find that on a flat surface, the effect is decidedly similar. So you have a velocity setting of zero for both the left and the right motors to accomplish the goal of not going anywhere. Next, you consider all the other possibilities, most of which involve detectable motion on the part of the robot.

To go forward, it would seem the thing to do would be to make the wheels roll forward and the robot should follow. What you have to remember is that due to the orientation of the drive wheels and the location of the servos with respect to the wheels, the two control signals must be exactly out of phase to produce forward motion. Some roboticians would solve this "problem" by simply reversing the sign of the velocity on one wheel; or worse yet, some would swap the wires on one of the motors and be done with it. I find that it makes more sense mathematically to treat them as 180° out of phase with respect to each other. For example, to move forward, the left wheel must rotate counterclockwise, while the right wheel must rotate in a clockwise direction. Do the opposite, of course, to go backwards. If both wheels are spinning clockwise, the robot spins in place to the left or counterclockwise if viewed from above.

A SIMPLE SOLUTION

This implies that each motor must be able to be directed to spin in either a clockwise or a counterclockwise manner. This is mechanically possible with servos, and it turns out to be easy to do with even the most simplistic of microcontrollers. Forget precision timing pulses. All you need is a short pulse for clockwise motion and

a long pulse for counterclockwise motion. The short pulse can be ridiculously short, as it need only be long enough to trigger the internal one-shot timer within the servo circuitry. Toggle an output line high and then immediately low again and you've completed the task. The long pulse is similarly crude but must not last overlong—"over" being defined in terms of nearing the expiration time of the control signal repetition rate. Anything over 2 ms and less than around 10 ms will suffice. These values, along with stop, are summarized in Table 1.

A complete, if brief, PBASIC program for the Parallax BS2 that will drive a two-wheeled robot forward in a mostly straight line is presented in Listing 2. This example assumes the left servo motor is controlled by P15 and the right by P14. These happen to be the lines I chose for the DPRG Yellow Robot because they were conveniently brought out to servo headers on the Board of Education. The magic numbers in the PULSOUT command are derived from the peculiar granularity of the BS2-specific PBASIC interpreter, resulting in 0.5- and 3 ms-pulses, respectively. The constant 16 in the PAUSE command is the remainder of the time that needs to be eaten up in the update interval after the pulses have been emitted and a little more for program execution. Your homework is to figure out how to drive in reverse, with extra credit for discovering the secret of spinning in place.

TRICKERY

One side effect of the simple drive-forward code previously listed is that both motors are triggered at almost the same time. By staggering the triggering

Listing 3—This C loop is a complete FreeRTOS task that generates a forward-reverse-stop control signal for a servo without using any special timers or PWM hardware. The value of the `servo_speed` global variable is used to determine the direction (see Table 1).

```
for(;;) {
    if(servo_speed) {
        PORTC = 1; // start servo pulse
        if(servo_speed < 0) vTaskDelay(mainSERVO_LONG_PERIOD); //
    extend it by 2 ms
        PORTC = 0; // end servo pulse
    }
    vTaskDelay(mainSERVO_UPDATE_PERIOD);
}
```

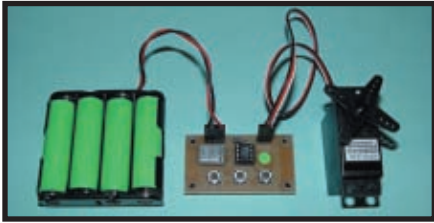


Photo 1—The Servo Tester Mark I was a simple design to provide a calibration signal for modifying servos for continuous rotation. It can also be used to test unmodified servos for proper operation. I built several units until it became apparent that a more versatile and accurate version would be required.

of the motors, the peak power draw can be theoretically cut in half. This is a big deal for battery-powered robots, especially on game day. This was accomplished by implementing a state machine that kept track of which side's turn it was to trigger and running the loop at twice the normal frequency. I then added another layer of trickery by omitting every other control pulse to achieve half-speed operation. This was required in my solution to one of the contest challenges: wall following. The robot proceeds by arcing forward and to the right, until

either of the bump switches detects a wall collision, then rotating left in place for a fixed period of time, and finally resuming the forward arc to the right.

This works three out of four times but gets hung up on the outside corners from time to time. To arc to the right, the left motor goes forward at full tilt and the right motor goes forward at approximately half speed. Have a look at the source code and try to figure out how the other contestants fared. There are some obvious and some not-so-obvious improvements that can be made. I usually think of these on my way home from the contest, of course. The complete PBASIC code that was actually used during the competition is available for download.

MORE COMPLEX SOLUTIONS

The same long, short, no-pulse algorithm for servo control can also be easily implemented in more sophisticated systems, using multithreading techniques. I used Richard Barry's FreeRTOS real-time kernel to create two independent tasks on an Atmel ATmega32 based on the

"AVR_ATMega323_WinAVR" example provided in the FreeRTOS download. The first task merely toggles a flag back and forth every 2 s for testing purposes. The second task outputs a long, short, or no pulse depending on the current state of the flag (see Table 1).

Each task is implemented as a continuous loop and has no knowledge of the operation of any other task, except for the value of the global variable `servo_speed`. FreeRTOS supports

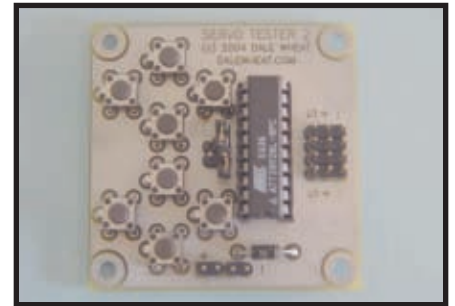


Photo 2—The Servo Tester II added a second servo channel and an "auto sweep" function. This device is available assembled and tested. Figure 1 and the full source code are available for download if you'd rather build your own, which will allow you to add or change the functionality to suit your needs.

Tight security...

...a [programmable] industrial network gateway for secure communication and network bridging



- Ultra-low power Intel IXP425 XScale processor
- Proven embedded Linux 2.6 kernel
- Fully user programmable
- Solid state, fan-less platform
- Two 100baseTx Ethernet channels
- Hardware accelerated encryption
- Four serial ports and four USB v2.0 host ports
- CompactFlash port for Wi-Fi LAN
- Tamper detection



a division of

VULCAN ICE Gateway... a versatile Linux platform powered by an Intel XScale communication controller for secure, high reliability network management and protocol conversion.

Customer support Product longevity Extended warranty RoHS compliance

888-941-2224

www.arcom.com



Think Embedded. Think Arcom.

more elegant forms of inter-process communication (IPC), but I just wanted a simple example. The servo update task is composed entirely of an endless loop (see Listing 3).

Bit zero of port C is the servo output, having been previously initialized as an output at some point not illustrated. The `mainSERVO_LONG_PERIOD` and `mainSERVO_UPDATE_PERIOD` constants are defined as 2 and 20, respectively. While this works and illustrates the latitude available with the servo update period, a more robust implementation would adjust the update period, depending on the length of the output pulse, to keep the overall update interval constant. Note that this works well enough for full forward and reverse on modified servos, but it lacks the resolution and repeatability needed for positional stability with standard servos.

GO ANIMATE

Now you have several methods at your disposal for animating your projects. Let's see some waving hands and

smiling faces on the next crop of robots! Even a little block that runs around and bumps into things all day long is more interesting than something that sits on the bench and displays π to a bazillion digits. Weld some wheels to the bottom of your PC and see what happens. How about a truly "mobile" phone? I'd love to hear about your servo-based projects. I hope this little introduction helps get your projects moving. Drop me a line if you have any questions or comments, and thanks for reading this far! ☺

Author's note: Assembled versions of the Servo Tester II are available from BG Micro (www.bgmicro.com) and E-Clec-Tech (www.e-clec-tech.com).

Dale Wheat (dale@dalewheat.com) is an old-fashioned tinkerer, inventor, musician, philosopher, and student of the human condition. He has no formal professional education. He enjoys reading science fiction short stories and playing a variety of musical instruments. In the summer, he enjoys

mowing two acres of grass and in the winter enjoys not mowing it. In 2005 and 2006, Dale was the president of the Dallas Personal Robotics Group (<http://dprg.org>), one of the nation's oldest personal robotics interest groups.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

RESOURCES

D. P. Anderson, "My Robots," 2006, www.geology.smu.edu/~dpa-www/myrobots.html.

R. Barry, "FreeRTOS," 2006, www.freertos.org.

Dallas Personal Robotics Group, <http://dprg.org>.

SOURCE

ATmega8/32 and ATtiny11/26
Atmel Corp.
www.atmel.com

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

Tianma LCDs



Tianma Microelectronics specializes in the manufacture of all LCD technologies from TN to TFT. With more than 22 years of LCD experience, we have grown to be China's largest LCD manufacturer. Tianma has teams of dedicated and experienced engineers and sales people worldwide to help you with all of your LCD needs. Our continued success in cell phone, automotive, consumer, and medical industries assure our ability to help you in whatever market you serve.

TIANMA

21138 Commerce Pointe Drive, Walnut CA 91789 USA
Tel: (909)468-2839 or 1-800-864-7789 • Fax: (909)468-0868
Email: sales@tianma.com • Web: www.tianma.com

USBee DX

Oscilloscope/Logic Analyzer

2 Analog and 16 Digital Signals
Up to 24Mps, 100+ Million Samples
Dual 8-bit ADC, +/-10V inputs

Bus Decoding

Click and Drag Instant Decode of Bus Transactions
I2C, SPI, ASYNC, CAN, USB Low and Full Speed
I2S, SM Bus, 1-Wire, PS/2

Digital Voltmeter

2-Channel, +/-10V, 8-bit ADC, Logging

Data Logger

2 Analog and 16 Digital Signals Plus Timestamp

Digital Signal Generator

16 Digital Outputs, Up to 24Mps
Playback of Logic Analyzer Traces

Pulse Width Modulator

16 User Controlled PWM Channels

Frequency Counter

16 Channel Counter to 24MHz

Frequency Generator

Generates Sets of Common Frequencies

I2C Controller

Control I2C Devices Using Simple Text Scripts

Remote Controller

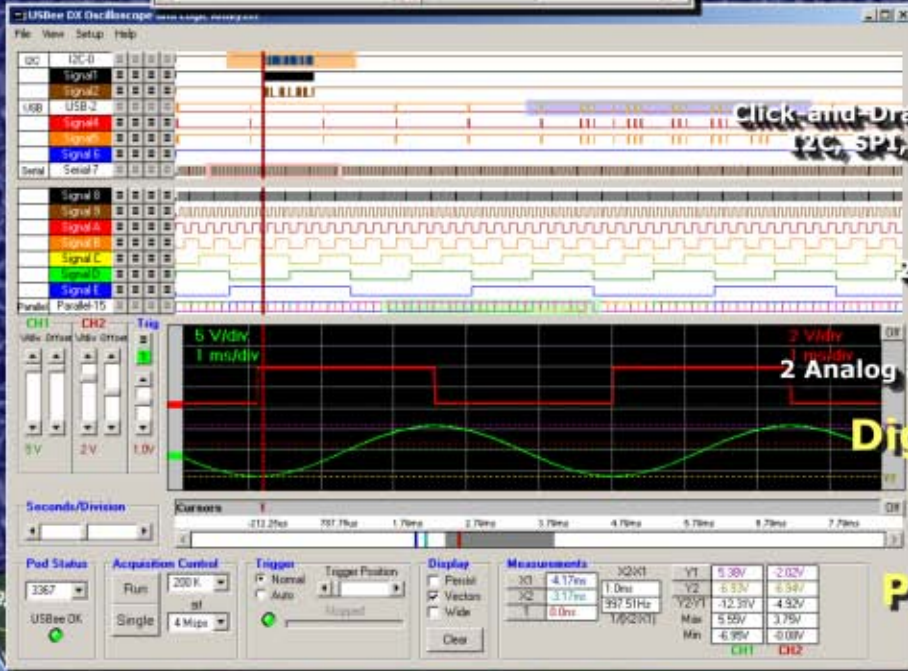
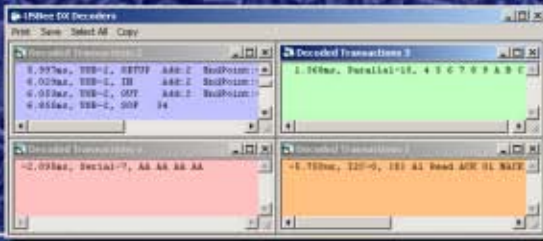
Easily Control Your Hardware Using Your PC

Pulse Counter

16 Channel Pulse Counter With Gating Control

plus the USBee Toolbuilder Source code and Library

Create Your Own Applications to Control the USBee DX



Actual Size

Data Extractors

Optional Software Modules for the USBee AX-Pro and USBee DX

Continuous Real-Time Embedded Bus Data Streaming

Store Bus Data to Disk or Send to Your Custom Application

Capture and Process Entire Test Sequences

Parallel, Serial, I2C, USB, ASYNC, CAN, SMBus, SPI, I2S, 1-Wire

USBee.com (951) 693-3065
support@usbee.com





PLATINUM SPONSOR



WITNESS THE POWER OF BRAINPOWER

WHEN BRAINS COLLIDE

EMBEDDED SYSTEMS CONFERENCE SILICON VALLEY
APRIL 1-5, 2007 MCENERY CONVENTION CENTER, SAN JOSE, CA

REGISTER BY MARCH 6TH FOR BEST CONFERENCE PRICING. ENTER PRIORITY CODE: U128

IN THE NOT TOO DISTANT FUTURE, IN A PLACE CALLED SILICON VALLEY, AN UNPARALLELED GATHERING OF GREAT MINDS WILL TAKE PLACE.

THE WORLD'S LARGEST INTERNATIONAL EMBEDDED EVENT AND TECHNICAL CONFERENCE

Be there at the **Embedded Systems Conference**. Join the power of brainpower in action. We're talking engineer brains here. On their own, they tackle the toughest problems. But when these brains collide, any challenge confronting humankind can be taken down. From global warming, to breakthroughs in cancer research, or vast leaps in communications technology. Heck, you can even solve that pesky power dissipation problem back at your cubicle. **ESC is access: to other brains, to information, to vendors, to new products.**

At ESC in Silicon Valley, you can put your brains against the best in the world. Including the mighty mind of **Al Gore**, champion of the planet who will be delivering the keynote address. Is this a must-attend conference? Let's just file that under *no-brainer*. Register for the best value All-Access Conference pass for full access to the event.

©2006 CMP Media LLC. All rights reserved.
WWW.EMBEDDED.COM



More “Hello World”

Moving What You’ve Learned to the Hardware

As promised, George continues showing you how to use C language for embedded system design. This time he delivers cleaner code, more functional requirements, and interrupts.

Last time, I started exploring how to use the C programming language for embedded system design (“Hello World ... Want Cookie,” *Circuit Cellar* 198, 2007). I created a program that turns on light on when an input switch is pressed and turns it off when it’s released. It used no interrupts, just brute force. I also promised a cleaner, faster version. Well, this month I’d like to add more functional requirements, introduce interrupts, and implement all of this on a real CPU. Oh yes, I’ll also clean up that code as promised. If you’re new to C, I recommend that you start with that previous article.

Using the C language, we covered declaring variables, sizing variables, procedures, #defines, pointers, if-else statements, true and false, bitwise logic operations, and for(; ;) loops. Also, I recommended B. Kernighan and D. Ritchie’s *The C Programming Language* as your first language reference. We ended that article with a program that operates a light whose output state is derived from a switch input.

REAL TIME

Before we get to real hardware, let’s add a real-time requirement. Say output light one is to blink at a slow rate when no switch is pressed and at a fast rate when any of the switches are pressed. How many switches are there? Marketing will get back to us on that issue, but the completion date for this project can’t slip.

The class of processors we’re dealing with has some sort of timer/counter hardware. Perhaps only one or several. Timers and interrupts go hand in hand with our real-time requirements. These CPUs also have interrupt capa-

bilities. Typically, when an interrupt occurs, normal program execution stops after the completion of an instruction, and the address of the next instruction is saved. The status of the CPU is also saved (typically on a stack), and then program control is also transferred through a memory location known as an interrupt vector or an interrupt vector table. The address of the interrupt routine is fetched using the table entry that is assigned to the interrupt that just occurred. Interrupt processing is started at the location pointed to by that vector. Interrupt processing continues until a return from interrupt instruction is encountered, and finally the status of the CPU is restored and normal program execution continues from the program instruction whose location was saved on the stack.

That’s the two-cent version of interrupts. There are a few more details (actually many more) that we’ll get into in future articles. There are several types of nonmaskable interrupts (NMIs), hardware, and software. NMIs are just that. They can’t be ignored or masked. Hardware interrupts can have several sources and they each typically have a priority level. Higher priority interrupt routines can interrupt lower priority interrupt routines. There is also an interrupt enable (IE) flag in one of the CPU’s registers. Once this flag is set, all interrupts are active. If this flag is cleared, only the NMI is active. Software interrupts are just that, generated by a software instruction. These are useful for debugging or implementing features in software if no hardware support is provided. The multiply instruction is a good example. Some versions of the CPU will have dedicated hard-

ware (more expensive), while other versions will only have software routines for multiply (slower, but cheaper). The code for both looks the same. When a multiply instruction is encountered, it’s either executed in hardware or a software exception (interrupt), or it is generated and the multiply is carried out in software.

The next level of detail necessary to understand interrupts is that when an interrupt occurs, the address of the next instruction is saved on the stack, the CPU flag register is also saved, and typically the IE flag is cleared. So you the user can execute the interrupt-routine code. When interrupt processing is complete, you execute a return from interrupt (RETI) instruction. The flags are restored and that saved next instruction is executed. Any of the shared processor (or system) resources that you touch during interrupt processing must be saved and restored. A typical shared resource you are likely to use is a CPU register. If you write the interrupt code in C, then the language and the compiler will know which registers were used and save/restore them for you. If you write the interrupt in assembly language, then you’re on your own.

If you get an interrupt and would like other interrupts to interrupt your work then just reenables the IE flag. Or you can use the priority mechanism built into the processor and let the hardware sort it all out.

My model for working with interrupts is to think of the interrupt routine as a separate entity. Anything it needs, such as variables, cannot be shared with the noninterrupt portion of the code and vice versa. At first, it seems like a

major restriction, but as you will see, it's the safest way to get really stable operation out of a complicated system.

REAL MICRO

Well, it's time to get into a real micro. Let's start with the Texas Instruments MSP430. That's a family of low-power (really low) 16-bit microcontrollers. I'll let you search on Texas Instruments's web site for the MSP430 and get showered with information. The project I'm going to use to describe my code uses a part in the MSP430x11x2 family. Search the site for the datasheet if you want all the details, but you won't need them for a bit. You will also find that manufacturers have a range of offerings and alter the content and pin count to match up to specific user requirements.

Let's start by looking at code examples for an interrupt that I'm using in a digital filtering project. Listing 1 is the timer/interrupt code. The keyword `__interrupt` (double underscores) defines the procedure `void TimerISR(void)` as an interrupt routine to the compiler. I'm using the IAR compiler and we'll talk more about that later. Just 20 lines of C code and that's it for the interrupt routine. The variable `input` is defined as an INT16 (16-bit integer) and is read each time this routine is executed. The A/D converter is set up to automatically start another conversion each time it's read. I use the ADC input for digital filters that I won't cover in this article. The `Msec10Timer`, `timer2`, and `StateTimer` variables are also 16-bit integer entities. The constant `MSEC10_TIMER` will be explained shortly. The C operator `++` in this listing, simply increments the variable one count. Look for more on this operation in later articles.

Every time this routine is executed, the `Msec10Timer` variable is incremented. When that value is greater than some constant, the value is reset and other variables are incremented. When the `Msec10Timer` variable is greater than the constant `MSEC10_TIMER`, that means that approximately 10 ms in real time have transpired. So the `Msec10Timer` variable is reset to zero and other timers

Listing 1—This routine reads the ADC, restarts the ADC, and manages the timers.

```
//*****[ interrupt ]**** ( 1)
// ( 2)
// Timer interrupt service routine ( 3)
// ( 4)
// This should be at a 3640 Hz rate = 0.274725274 usec ( 5)
// ( 6)
//***** ( 7)
__interrupt void TimerISR(void) { ( 8)
( 9)
input = ADC10MEM; // Read & restart the A/D (10)
// another conversion (11)
(12)
Msec10Timer++; // Increment a counter (13)
if (Msec10Timer > MSEC10_TIMER) { // 10 ms (14)
Msec10Timer = 0; (15)
timer2++; (16)
StateTimer++; // inc the State Timer (17)
} (18)
(19)
} // end of __interrupt void TimerISR(void) { (20)
```

are incremented. The details of this timing are explained a little later.

The next thing you need to do is insert the location of this interrupt routine into the interrupt vector table. Listing 2 is the code fragment for that table. This is from an assembly language routine. Hey, wait a minute. Isn't this all about using C and not assembly language? Yes it is, and for those with at least brown belts in C, Dan Saks wrote about how to do interrupt vectors in C.^[1] Since we are just getting our feet wet in the shallow end, it's easier to use the assembly language table. The original code had all unused vectors pointing to RESET. This is a good solution for production, but I wanted to trap any unexpected interrupts while testing. So, I changed them to point to a routine that just loops on itself. This way I know I have a problem with a particular interrupt source. It is also better than just getting reset and then trying to figure out why. The last point we need to cover is setting up the timer.

Listing 3's routine sets up all the hardware used in this system. The variables that appear in all caps are defined (constants). Now do you see why we adopted that convention? You know to look elsewhere for their meaning. And when we're dealing with all the registers and bits in a

modern CPU, it's the only way. Some of the constants are defined for the particular CPU and some of the constants I defined. The code fragment in Listing 4 is at the beginning of each file that referenced these defines.

The C header file is a file that is included right where it's found in the C source file. It's just as if you cut and pasted the header file into the source file. These header files contain information that you want to make available to the source file. By including statements that have a header file surrounded by `"` characters, the search path begins where the source program is found.^[2] If the header file name is enclosed with `<>`, the search follows implementation-defined rules. Line 1 in Listing 4 includes a header file defining all the CPU register and bit constants. While line 2 includes a file named `CCconsts.h` (*Circuit Cellar constants*), that file contains all the constants and definitions for this particular project.

Listing 3 contains a lot of information, let's look at it in more detail. Line 1 is a comment line. Line 6 is the procedure declaration. Line 8 stops the 430's watchdog timer. The watchdog timer is an interrupt source that's supposed to monitor your program's operation, and if it's not reset every so often, it will generate an interrupt

warning you that your program is out of control. Lines 10 through 12 set the internal oscillator to run as fast as possible (DCOCTL, DCO Clock Frequency Control and BCSCTRL1, Basic Clock System Control). Lines 14 through 18 define the pins as either inputs or outputs and set some initial output values. Lines 21 to 27 set up the ADC. And lines 30 to 35 set up the timer. All of that is done in C, and you must admit it's not that far away from an assembly language implementation.

Note, for the MSP430, PnSEL selects whether the pin is a normal I/O pin or dedicated to a special function, PnDIR sets the pin's direction as either in or out, and PnOUT sets the output value of the pin.

Let's spend a little more time looking at the timer. The 32,768-kHz crystal goes into the timer input. Line 30 divides that crystal input by nine. So $32,768/9 = 3,640.88$ Hz, or 0.2746 ms, for every interrupt. I needed that frequency for the original design this code was copied from. I would like to

count 10-ms intervals so the constant MSEC10_TIMER is defined as 35. So:

$$\left(\frac{32,768 \text{ Hz}}{9}\right) \div 35 = \frac{3,640.88}{35} = 104.025 \text{ Hz}$$

Thus, lines 15 to 17 in Listing 1 advance every 104 times per second, or 9.61 ms. Could I have come closer to 10 ms? Sure $32,768 \text{ Hz}/328$ is only off by $98 \mu\text{s}$. But as I said, I needed 3,640.88 for the digital filters I was running and the blinking of our lights can be approximate. The point is that you have some flexibility in how you structure what goes on inside a timer.

MOVING ALONG

Well, learning is an action sport. Enough sitting around, let's install a compiler and get to it. I'm going to keep with the Texas Instruments MSP430 family. If you've got another, go ahead and use that one. The details of I/O system peripheral control will be different, but that's the point of using C. It should be easy to move designs

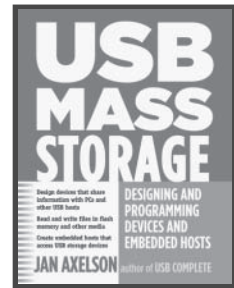
Listing 2—The *Trap9n(n)* labels will trap the CPU operation on any unexpected interrupts.

```

Trap1
    jmp Trap1
Trap2
    jmp Trap2
-- all vectors have a Trap location but are omitted here --
Trap15
    jmp Trap15
;-----
RSEG INTVEC ; MSP430x11x1 interrupt vectors
;-----
DW Trap1 ;RESET ;1 no source
DW Trap2 ;RESET ;2 no source
DW Trap3 ;RESET ;3 no source
DW Trap4 ;RESET ;4 no source
DW Trap5 ;RESET ;5 no source
DW Trap6 ;RESET ;6 A2D_
DW Trap7 ;RESET ;7 no source
DW Trap8 ;RESET ;8 no source
DW Trap9 ;RESET ;9 was Timer-AX
DW TimerISR ;10 Timer-A0
DW Trap11 ;RESET ;11 was Watchdog/Timer
DW Trap12 ;RESET ;12 was Comparator-A
DW Trap13 ;RESET ;13 no source
DW Trap14 ;RESET ;14 no source
DW Trap15 ;RESET ;15 was NMI, Osc. fault

```

ADD MASS STORAGE TO YOUR DESIGNS



USB Mass Storage Designing and Programming Devices and Embedded Hosts Jan Axelson

ISBN 1-931448-04-3 \$29.95
Lakeview Research LLC www.Lvr.com

From the author of **USB Complete**

USB/ETHERNET DAQ

LabJack UE9



Available now
for only ...

\$429 qty 1

USB/Ethernet Data
Acquisition & Control

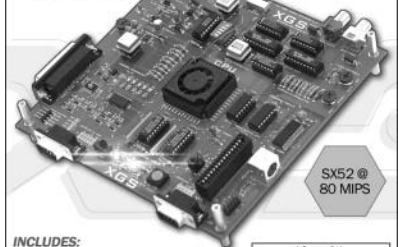
- * USB 2.0/1.1 and Ethernet
- * 14 analog inputs (12- to 16-bit)
- * Stream input data up to 50 kHz
- * Use with C, VB, LabVIEW, etc.
- * Includes DAQFactory Express
- * Operates from -40 to +85 deg C
- * 2 analog outputs (12-bit)
- * 23 digital I/O
- * Up to 2 counters (32-bit)
- * Up to 6 timers
- * Approx. 3" x 7" x 1"

LabJack Corporation, Colorado, USA
info@labjack.com, (303) 942-0228

www.labjack.com

ANDRE LAMOTHE'S XGAMESTATION DO-IT-YOURSELF VIDEO GAME SYSTEM

Inspired by the
Atari 2600, Apple II &
Commodore 64!



INCLUDES:

- Assembled XGS Micro Edition Unit!
- Complete Development Kit!
- Tools, Demos & Utilities!
- eBook on Designing the XGS Console!
- Cables and Power Supply Included!



WWW.XGAMESTATION.COM
(925) 736-2098 SUPPORT@NURVE.NET

among different processors. On TI's web site, under Get Design Support, enter Microcontrollers and Selection and Solution Guide. Then press Go. This gets you to the TMS430 technical documents. You'll also see TMS470 and DSP products. Don't go there! Seriously, there is just too much information available. Stick with the TMS430 for this installment. You will find selection guides, code examples, and development tools. Take some time to look through the information. I'm going to stick with one of the smaller, less expensive versions of the TMS430.

The header file is msp430x11x2.h and the key document is the "MSP430x1xx Family Users Guide," which gives a technical description of the microcontroller (actually, several in this subset of the TMS430 family). You'll see details about all the available peripherals, along with the instruction set. Now, if everything goes well, you will never have to look at the instruction set sections again. Actually, I look at them while I'm getting to know the CPU. The C compiler can output the C source lines mixed with the assembly language code that was generated. I keep an eye on this to make sure the code does not get bloated.

Next, look up Compilers/Assemblers/Linkers under development tools. You will see that Texas Instruments and IAR offer free (code limited) C compilers. I'm going to talk about the IAR because that's the one I use. Look up www.iar.com and then C compilers. Follow the links to IAR Embedded Workbench version x.xx for the Texas Instruments MSP430. You'll find a link to the 4-KB Kick Start edition. You will need to register with IAR and then you'll come across a download page. Download the compiler, install, and enter the keys as directed in the instructions. Ta-da!

Most of the modern IDEs have a project structure that they want you to follow. I use several and never take the time to quite figure out how to finesse my preferences on the IDE's directory structure. So, I just get it sort of close to what I want for a project structure and move on.

Let's try a directory structure something like this: D:/CircuitCellar/LearningC/MyFirstProject/Latest, where

D: is the drive containing your data files, CircuitCellar is the customer, LearningC is the product group, MyFirstProduct is the product, and Latest is the latest version of the design. This may seem a bit cumbersome, but as your projects and designs grow, you might find yourself thankful for this detailed structure. I've also found that it is difficult to point the IDEs to different versions of the project. So, I will copy the contents of Latest into V00-01, V00-02, and so on as I keep changing the code.

When everything is installed, start the IAR workbench and select "Create a new project in current workspace." Then select C and main. Notice the explanations as you make the choices. Then, name your latest project and

you should be placed in the IDE with the main.c file opened. From the tool bar, select Project/Rebuild All. Congratulations you have just compiled your first C program. You now have a working IDE that will compile C, assemble, link, and locate all for just some download time. I know this was a lot to follow and I bet you won't get the directory structure to your liking on the first try. So work with it and get comfortable.

Now, under Project/Options in the main tool bar, you can set up the tools to suit your needs. Some of what I do on small projects is as follows: Under General-Target, I select the correct Target CPU (msp430F122). Under C Compiler-Language, I allow IAR extensions. Under C Compiler-Code, I do not allow opti-

Listing 3—Configuring the Texas Instruments MSP430 hardware that we will use in our system.

```
//*****[ public ]**** (1)
// (2)
// Hardware Setup (3)
// (4)
//***** (5)
void SetupHardware(void) { (6)
(7)
WDTCTL = WDTPW+WDTHOLD; // Stop the watchdog (8)
(9)
DCOCTL |= DCO2+DCO1+DCO0; // set DCO to run (10)
// at fastest setting (11)
BCSCTL1 |= RSEL2+RSEL1+RSEL0; // XT2=ON (12)
(13)
P1SEL &= ~(BIT3+BIT2+BIT1+BIT0); // These PINS are Digital I/O (14)
P1DIR |= (BIT3+BIT2+BIT1+BIT0); // These PINS are outputs (15)
P1OUT |= (BIT3+BIT2+BIT1+BIT0); // Set outputs to 1 (16)
P2SEL &= ~(BIT4+BIT3+BIT2+BIT1); // These PINS are Digital I/O (17)
P2DIR &= (BIT4+BIT3+BIT2+BIT1); // These PINS are inputs (18)
(19)
(20)
ADC10AE = 1; // Analog input A0 enabled P2.0 (21)
ADC10CTL1 = ADC10DF; // Select ADC out data format as (22)
// signed left justified (23)
ADC10CTL0 = ADC10SHT_1 + ADC10ON; (24)
// ADC10 ON and mode set (25)
(26)
ADC10CTL0 |= ENC; // Enable conversions (27)
(28)
// Set up Timer A (29)
CCR0 = 8; // CCR0 for sampling rate of 3640Hz (30)
// from 32768Hz ACLK signal (31)
// (32768/9 = 3640) (32)
(33)
CCTL0 |= CCIE; // enable CCR0 interrupt (34)
TACTL = TASSEL0+MCO; // TACLK = ACLK, Count up to CCR0 (35)
(36)
} // end of void SetupHardware(void) (37)
```

Listing 4—The `Msp430x112.h` file is from TI, while `CCconsts.h` is one that I created for this design.

```
#include <msp430x11x2.h>    (1)
#include "CCconsts.h"      (2)
```

mizations. Under C Compiler-Output, I click to generate debug information. For C Compiler-List-Output, I list file-assembler mnemonics. These are probably the default settings, but you should understand what options are available.

Let's talk a little about style. I prefer to convert the tab character to spaces lined up every three. Whenever there is a `{}` pair, I put the first one (opening) on the line that started it all, and the last one (closing) on it's own line. Look at Listing 1, lines 14 and 18. And the last style issue for now is in a procedure. I do the `{}` as just mentioned, but I do not indent the first line of code. Look at Listing 1, lines 8 and 20. I feel this style gives me good visual spacing for indentation and brackets but keeps the code compact. We'll get into formatted source print utilities, and these style choices go well with those utilities.

ASSIGNMENT

Your assignment is to enter the code that I've described in these first two articles and compile it. Chase down the errors until you get a clean compile. Look at the code generated. Are you surprised at anything? Next time, we'll look deeper into the compiler and fire up another IDE.

If you want to run any of your code on real hardware, all the manufactures and other suppliers offer low-cost (less than \$100) evaluation boards. Check out the offerings on the Texas Instruments web site for examples of what's available.

Also, the IAR environment includes a simulator. So, even if you didn't buy any target hardware, you could still step through all the C code and observe its operation.

As usual, I'll be happy to address any of your questions on the *Circuit Cellar* bulletin board (<http://bbs.circuitcellar.com/phpBB2/>). ☐

George Martin (gmartin@circuitcellar.com) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his

own and co-founded a design and manufacturing firm (www.embedded-designer.com). George's designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Ciarcia Design Works Team. He's currently working on a mobile communications system that announces highway information. George is a nationally ranked revolver shooter.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/200.

REFERENCES

- [1] D. Saks, "Modeling Interrupt Vectors," *Embedded Systems Design*, 2006, www.embedded.com/showArticle.jhtml?articleID=192503651.
- [2] B. Kernighan and D. Ritchie, *The C Programming Language*, Second Edition, Section 4.11.1, Prentice Hall, 1988.

RESOURCES

S. Harbison and G. Steele Jr., "C a Reference Manual," Prentice Hall, 1995.

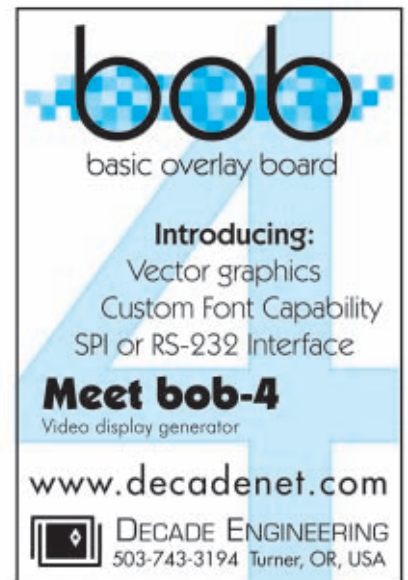
P.J. Plauger, "The Standard C Library," Prentice Hall, 1991.

Texas Instruments, Inc., "MSP430 Microcontroller Datasheet," SLAS439C, 2006, <http://focus.ti.com/lit/ds/symlink/msp430f2131.pdf>.

———, "MSP430x1xx Family Users Guide," 2006, <http://focus.ti.com/lit/ug/slau049f/slau049f.pdf>.

SOURCE

MSP430 Microcontroller
Texas Instruments, Inc.
www.ti.com



bob
basic overlay board

Introducing:
Vector graphics
Custom Font Capability
SPI or RS-232 Interface

Meet bob-4
Video display generator

www.decadenet.com

DECADE ENGINEERING
503-743-3194 Turner, OR, USA



**MSP430
CPU12
ARM
AVR**

Easy to Use!
Low Cost!
State of the Art!

Imagecraft compilers

starting at **\$199!**

high powered development, garage powered prices!

Fully Functional Demos!

Plus:
AVR, ARM & HCS12 Hardware Kits

www.imagecraft.com
info@imagecraft.com
(650) 493-9326 • FAX (650) 493-9329



PIC-SERVO
MOTION CONTROL

MOTION CONTROLLERS FOR
BRUSH, BRUSHLESS AND
STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com
JEFFREY KERR, LLC



A Plethora of Projects

Since 1988, Ed has written more than 120 articles for Circuit Cellar on everything from home control to programming. In this article, he covers his all-time favorite projects.

My wife, a budding author, persuaded me to join her in a “Write, Write, and Rewrite” class for aspiring essay writers. When discussion turned to the difficulties of finishing an essay and sending it off for publication, perhaps I wasn’t sympathetic enough when I quoted Dr. Samuel Johnson: “The gallows doth wonderfully concentrate the mind.”

Although *Circuit Cellar*’s editors don’t threaten me with the gallows very often, producing a regular technical column does concentrate one’s mind on getting the job done. All in all, though, when they’re happy, I’m happy!

Perusing the master Table of Contents on the *Circuit Cellar* web site shows that I’ve written 120-some-odd columns and articles for the last 200 issues, covering everything from home control to protected-mode programming to RF circuitry. Here’s a look at some of my all-time favorite projects.

WATER BOTTLE ROCKETRY

Steve presented the Circuit Cellar Strategic Defense Initiative in Issue 2, based on a gadget from my machine shop that was so much fun I still get an occasional e-mail about it. Photo 1 shows a launcher that fired 2-liter soda bottles out of sight: some bicycle brake components, a few brazed fittings, and three book-

shelf struts. The compressed air tank and enthusiastic bystanders don’t appear in the picture, but they were definitely present.

Blasting bottles into treetops wasn’t enough for the *Circuit Cellar* crew, though, because we wanted numbers. You’ll see that attention to detail in every issue: not only do you get construction information, but performance measurements and background to show why the information’s relevant in the real world.

The bookshelf struts supported thin wires that the rocket dislodged in passing. A real IBM PC recorded the elapsed times through its parallel printer port, using a reprogrammed hardware timer. In those days, of course, we told you how to modify a printer port to enable data input.

Although this wasn’t the world’s

first serious water rocket, it may have sported the first launchpad instrumentation. I found 2-liter bottles accelerated at about 400 G and departed the gantry at nearly 250 ft/sec. That’s 170 MPH, or 275 km/h for you metric folks. Not bad for a bottle of air, eh?

A followup in Issue 4 (1988) described a better launcher and you can find improvements on the web today after two decades of innovation. Oh, by the way, don’t believe any of the slanderous stuff Steve wrote; it’s mostly all true.

FIRMWARE AND HARDWARE

“Firmware Furnace,” my first column series, explored the low-level interface where software meets silicon. The now-classic Intel 8051 Microcontroller figured prominently in early columns, with applications ranging from simple LCDs, to power-line communication, to digital video, with a DIY in-circuit debugger thrown in for good measure.

Capturing and displaying analog video with an 8051 microcontroller wasn’t easy, but some precise firmware and a handful of cheap external parts did the trick. I think I was the first to explain why the usual software-intensive approach produced unacceptable jitter and show how to reduce it. My column in Issue 3 (1988) certainly raised some hackles with tightly interlocked



Photo 1—Combine some bike parts with a soda bottle, add some parallel-port electronics, stir in a bit of firmware, and we had the single most popular project ever: *The Circuit Cellar Strategic Defense Initiative*. Two hundred issues later, we still get e-mails!

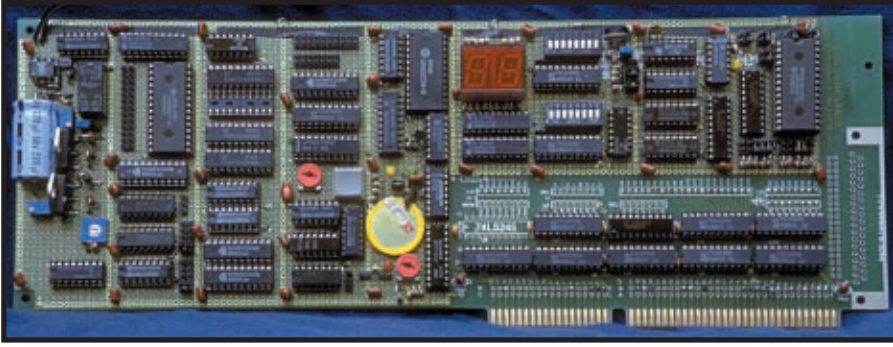


Photo 2—An ISA-bus board stuffed with everything from a serial number to a non-LSI graphic LCD interface lets us explore hardware design, bus and BIOS interfacing, and firmware control. Trivia quiz: What happened when you pushed that gray button?

mainline and interrupt handler code, because the handler didn't save any registers at all!

Although PCs weren't yet cheap enough or reliable enough for real embedded use, that trend was becoming clear, even when an Intel 33-MHz 80386SX microprocessor was a fast CPU. In Issue 11 (1989), I looked at how cache misses (in the few CPUs that had caches!) clobbered the precise timing we'd come to expect from microcontrollers. In Issue 19 (1991) I

traced instruction timing down through interrupts, past DRAM refresh and into the CPU's prefetch queue.

In Issue 31 (1993), I began exploring the intricacies of the ISA bus used in IBM PC-compatible systems, starting with a bare expansion board and culminating in a large bitmapped graphic LCD by Issue 46 (1994). Photo 2 shows the resulting board, built entirely with hand-wired TTL logic and interfaced to the PC bus through both memory and I/O ports. Unlike contemporary PCI and

PCI Express interconnects, the ISA bus was eminently hackable, primarily due to its relatively low speed, and I showed how the bus control and timing signals interacted with external hardware.

From the beginning, Intel's 80x86 addressing was widely regarded as baroque and protected-mode programming wasn't well-understood, so in Issue 48 (1994), I began my "A Journey to the Protected Land" series that built up a miniature multitasking kernel while examining some obscure PC hardware features. To put this in perspective, in those days the Linux kernel was at 1.1.

Unlike Linux, however, my code used Intel's full-throttle segmentation model, with privilege rings, memory protection, and task isolation. I even included a simple Virtual-86 mode "DOS Box" that could execute real-mode code. It wasn't a full operating system, but it did lay out the fundamentals at the hardware level.

Starting with Issue 1 (1988), Steve insisted that all *Circuit Cellar* articles have descriptive titles. Now, I ask you, doesn't "Serious CISC Meets the Tas-

EzPCB, Easy Way To PCB

Features:

- UL Proved High Quality
- ISO 9000 Certificated
- Accept Orders From 3pcs to 1M Pcs

Capabilities:

- Up to 40 Layers
- 2.5mil Track/Space
- 0.1mm Hole Size
- Blind And Buried Vias
- RoHs, WEEE Compliant

Prices:

- Only \$50 For 5pcs 4" x 4" 2L
- Only \$188 For 3pcs 4" x 4" 4L
- Only \$288 For 3pcs 4" x 4" 6L
- Free Solder Mask & Silkscreen
- Very Competitive Price For Bulk Orders

[Http://www.ezpcb.com](http://www.ezpcb.com)
sales@ezpcb.com

kettes" tell you exactly what that column was all about?

THE ANALOG REALM

Much analog circuitry these days is designed and cared for by somebody primarily responsible for digital hardware. Unfortunately, many "analog problems" arise from a misunderstanding of basic analog principles, so my current columns explore those issues with circuitry that illustrates how design equations become real hardware and what happens in the real world.

For example, in Issue 151 (2003), I showed how nonlinear components can generate unexpected signal frequencies. I then showed how filters can isolate or eliminate those frequencies in Issue 161 (2003). Some of those components may even lie outside your circuit boards, as the crystal diodes I measured in Issue 195 (2006) demonstrate.

Sampled-data systems, even those with "simple" ADC and DAC circuits, have peculiar properties in the frequency



Photo 3—Pointing out subtle design details sometimes requires getting into the guts of the hardware. Remember, what's inside the box still counts!

domain. My Issue 185 (2005) and 187 (2006) columns showed why you really

need those mysterious antialiasing and anti-imaging filters and what happens when they're missing.

From the very beginning, *Circuit Cellar* has always maintained that "What's Inside the Box Still Counts," and we've always been willing to do whatever it takes to get the story. Photo 3 shows a particularly dangerous mission from Issue 163 (2004), where a Lentz Launcher that fired pennies into the ceiling showed how high current interacts with fast switching.


When you must understand how things work, start by reading, then fire up your soldering iron. Simulation is good, but rosin smoke is even better!

That bottle rocket was so much fun, I might have to do it again. Hmmm. ☹

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

FlashPro430 GangPro430


USB Flash Programmers for Texas Instruments' MSP430 microcontrollers.



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

- ✓ 60 kB Flash can be programmed in about 2 seconds
- ✓ supports JTAG, Spy-Bi-Wire and BSL interfaces
- ✓ can assign unique serial numbers
- ✓ up to eight programmers can be connected to one PC and program target devices simultaneously

New: FlashPro-CC and GangPro-CC
USB Flash Programmers for TI's Chipcon microcontrollers




One PC and 8 programmers

Elprotronic
Incorporated

www.elprotronic.com

HIGH TECH PROTOTYPE PCBs


Touch the Future



- MICROELECTRONICS—LINE WIDTHS DOWN TO 1.25 MIL (0.00125")
- HDI—STAGGERED, STACKED, AND FILLED MICRO VIAS
- HI-RELIABILITY LEAD-FREE BOARDS
- HEAVY COPPER BOARDS

SUPPORTING THESE MARKETS

• MEDICAL • MILITARY • SPACE • DOWNHOLE • HIGH RELIABILITY • SECURITY



SIERRA MICRO ELECTRONICS • SIERRA PROTO EXPRESS
800.763.7503
WWW.PROTOEXPRESS.COM



Embedded USB Breakthrough

Jeff recently got crafty with the new FTDI VNC1L embedded USB host controller IC. Now he doesn't need to carry around his laptop in order to log data from his designs.

I'm interrupt driven. By rights, this column should be one in the continuing series I began last month. I have a number of projects planned that coalesce around electric vehicles. In fact, the vehicle I began with last month is an electric bicycle. The first project centered on circuitry to monitor storage batteries up to 48 V (4 to 12-V lead acid batteries). Both charging current and operating current give a picture of battery capacity and efficiency. That circuit outputs a serial stream of ASCII data at 1-s intervals. I used a laptop to record the data. Since the operating data comes from riding the bike, I slung the laptop in a backpack while I rode a three-mile course typical of the topography of New England. Plans are underway to explore other areas (e.g., a LIN automotive bus, a vehicle dashboard, a course slope, etc.).

I am delaying all this to sneak in a project that uses a brand new chip to improve the task I ended with last month. I came across this device and instantly knew I needed to make use of it. I think you will feel the same way. Future Technology Devices International (FTDI), the guys that were first to bring us a USB-to-serial interface, have released the VNC1L-embedded USB host controller IC. The 48-pin flash memory-based controller is preprogrammed to interface twin host/slave USB ports with one of

three interfaces: serial, parallel, or SPI. This will eliminate the need to tote around my laptop in a backpack to log data from last month's project.

VNC1L

If you are like me, when you take a look at the VNC1L's block diagram bells and whistles will go off in your brain (see Figure 1). I ordered the VDIP1 prototyping module to get started. This module has the VNC1L and supporting SMT parts, along with a USB connector, all on a small 0.6" DIP-28 header. This way I can hand-wire some circuitry without messing around with any SMT stuff.

The VDIP1 module comes with Vinculum disk and peripheral interface (VDAP) firmware preprogrammed into the VNC1L chip (see Photo 1). VDAP firmware supports USB flash memory (or

other mass storage) drives. No need to think about FATs and file structures. All of a sudden today's most popular storage device, the USB flash/stick/thumb drive is accessible for use in embedded projects. Simple ASCII commands via the chosen interface control access to the USB storage device.

Although I will most likely add this as a peripheral to last month's project at some point, here it makes most sense to design a stand-alone logger. I want to be able to attach an RS-232 cable, open a file, and accept data until I close the file. To do this, I need to use a microcontroller between the data and the VNC1L. If the microcontroller has twin serial ports, then I could use the serial interface between the microcontroller and the VNC1L. However, I wanted to use the solitary serial port for data collection, which

left either a SPI or FIFO interface to the VNC1L. To achieve maximum throughput, I chose to use the FIFO interface.

Figure 2 shows how I wired up this project. The VDIP is plugged into a DIP socket made of headers JP1:2. Power comes into JP6, which can also have TTL serial on it. This eliminates the need for the MAX232 and DE9, but adding these parts makes the project more flexible for use in the future. LEDs 1 and 2 mimic the LEDs on the VDAP module (port 1:2 activity). These will

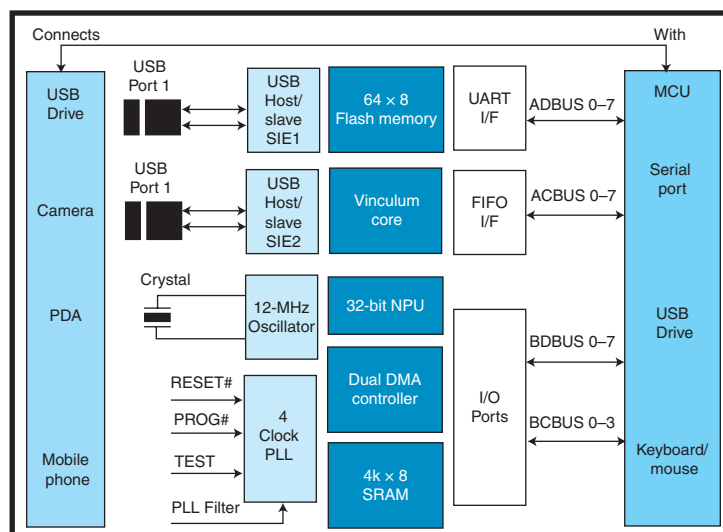


Figure 1—The VNC1L allows the designer to use many of the USB devices on the market via one of three available interfaces. The I/O ports are predefined as the full complement of asynchronous serial interface signals, a minimal SPI, or an eight-bit bidirectional parallel FIFO data port.

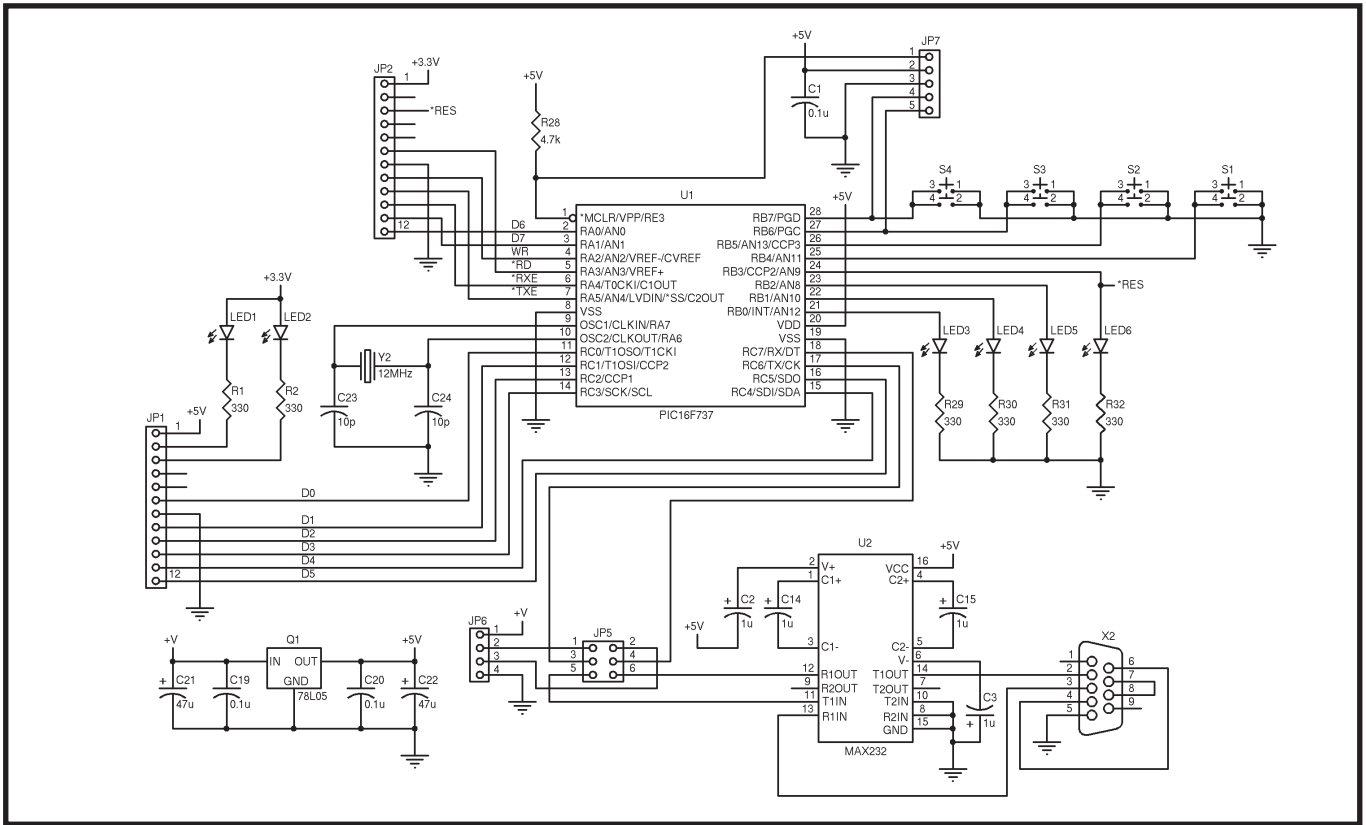


Figure 2—To aid in prototyping this application, I used the 24-pin VDIP module. This schematic shows how I used a microcontroller to interface with the VNC1L. The application detects an inserted USB flash storage device, opens a file, and stores <CR> delimited data to the file.

make more sense when I do a layout using discreet parts in place of the VDIP module. Four additional LEDs and four push buttons make up the user interface. I wanted to keep this design fairly simple, so I didn't use an LCD. An LCD would be necessary if you wanted to give the user the ability to enter file names or view the directory of the storage device. I thought that would make the device overly complicated. I opted for simplicity, and simple it is. I used only one push button in this application.

At reset (or at power-up) the VNC1L is programmed to check the ACBUS5:6 pins for user interface selection and send out a sign-on message through the selected interface connection. The message appears in the following format: Ver xx.xx, VMAP, On-Line. It also checks the status of the two USB ports and reports on any device. (A search is done on any storage device to check for ROM images, because the VNC1L can update its software via an automatic file transfer). Sending a carriage return (<CR>) reports the status of the port

with either a "No Disk" message or a D:\> prompt. Other commands work the same way. Most commands reply with the appropriate data, a "Bad Command," or a "Command Failed" (see Table 1).

As you can see in Figure 2, I tied LED6 to the VDIP reset line; therefore, LED6 is the status of the VNC1L. LED3 indicates when a device has been inserted. At this point you may request logging serial data by pressing button S1. The appropriate commands are sent to the VNC1L to search the directory for a file called FTB00.TXT. If the file is found, the file number is incremented (00-99) until the file is not found. It is then created and opened for writing.

A file can also be opened for appending data. LED4 indicates when a file is open. (At this point, as with any storage device, removing the device will close the file with a loss of data). The data stored is <CR> delimited, and I use this fact to save data. Data received from the microcontroller's serial port is stored in a RX data ring buffer. The microcontroller

watches this buffer for a 0x0D character. When one is found, it notes the number of characters in the string and commands the VNC1L to write that number of characters to the file. This command is actually two. The first is the WRF command containing the number of characters. The next command is the string of data. LED5 indicates a WRF is being performed. Pressing SW1 halts the logging process. You can begin another logging session by pressing SW1 again. No checks are made for free space; therefore, ample space must be ensured, which is not usually a problem.

VNC1L APPLICATION

To help debug and understand this device, I added a few options to this application. Both the SPI and FIFO interfaces are coded into the application. This is selected in the assembly code as the I/O definitions change with the selection. You can also choose to disable the logging application all together and have the microcontroller just pass serial port data directly to and from the VNC1L. This allows

you to connect to a PC running HyperTerminal, type in commands directly, and receive replies. Figure 3 shows a general flow chart for this application.

The major differences in the applications have to do with the type of interface you are using with the VNC1L. My initial reaction was to

use only the FIFO interface to get the fastest throughput. After wiring up the VDIP module to a microcontroller, I began developing some code for the

| ASCII | Binary | Function | Reply |
|---|--|--|---|
| <cr> | \$0D | Check if online | This will return the appropriate prompt or "no disk" message for the current command set. |
| Response to check if online for Extended Command mode | | If no valid disk is found | 'No Disk',\$0D |
| | | If a valid disk is found | 'D:>',\$0D |
| Directory operations | | | |
| 'DIR'<cr> | \$01,\$0D | Lists the current directory | A list of file names and directory names are returned. Each entry is terminated by \$0D. A directory entry has <sp>'DIR' after the name and before the \$0D. |
| 'DIR'<sp><name><cr> | \$01,\$20,<name>,\$0D | Lists the file name followed by the size. Use this before doing a file read so that you know how many bytes to expect. | \$0D,<name><sp><size> in hex(4 bytes) LSB first> \$0D |
| 'DLD'<sp><name><cr> | \$05,\$20,<name>,\$0D | Delete directory | Deletes the directory <name> from the current directory. <prompt>\$0D |
| 'MKD'<sp><name><cr> | \$07,\$20,<name>,\$0D | Make directory | Creates a new directory <name> in the current directory. <prompt>\$0D |
| 'CD'<sp><name><cr> | \$02,\$20,<name>,\$0D | The current directory is changed to the new directory <name> | <prompt>\$0D |
| 'CD'<sp>'..'<cr> | \$02,\$20,\$2E,\$2E,\$0D | Move up one directory level. | <prompt>\$0D |
| File operations | | | |
| 'RD'<sp><name><cr> | \$04,\$20,<name>,\$0D | Read file <name> | This will send back the entire file in binary to the monitor. The size should first be found by using the 'DIR' <sp><name><cr> command so that you will know how many bytes to expect. <prompt>\$0D |
| 'RDF'<sp><size in hex(4 bytes MSB first)><cr> | \$0B,\$20,<size in hex(4 bytes)>,\$0D | Reads the data of <size in hex(4 bytes)> from the current open file. | This will send back the requested amount of data to the monitor. <prompt>\$0D |
| 'DLF'<sp><name><cr> | \$07,\$20,<name>,\$0D | Delete file <name> | This will delete the file from the current directory and free up the FAT sectors. <prompt>\$0D |
| 'WRF'<sp><size in hex(4 bytes MSB first)><cr><data bytes of size><cr> | \$08,\$20,<size in hex(4 bytes)>,\$0D \$data,\$0D | Writes the data of <size in hex(4 bytes)> to the end of the current open file. | <prompt>\$0D |
| 'OPW'<sp><name><cr> | \$09,\$20,<name>,\$0D | Opens a file for writing to with 'WRF' | <prompt>\$0D |
| 'OPR'<sp><name><cr> | \$0E,\$20,<name>,\$0D | Opens a file for reading to with 'RDF' | <prompt>\$0D |
| 'CLF'<sp><name><cr> | \$0A,\$20,<name>,\$0D | Closes a file for writing. | <prompt>\$0D |
| 'REN'<sp><orig name><sp><new name><cr> | \$0C,\$20,<orig name>,\$20,<new name><cr> | Rename a file or directory | <prompt>\$0D |
| 'FS'<cr> | \$12,\$0D | Returns free space in bytes on disk. For disks of over 4 GB in size, this will return \$FFFFFFFF if more than 4 GB available. Otherwise, use 'FSE' command | <free space in hex(4 bytes) LSB first> \$0D |
| 'FSE'<cr> | \$12,\$0D | Returns free space in bytes on disk | <free space in hex(6 bytes) LSB first> \$0D |
| Debug commands | | | |
| 'IDD'<cr> | \$0F,\$0D | Identify Disk Drive. This will display information about the attached disk up to 4 GB. | Sends IDD data block and then <prompt>\$0D |
| 'IDDE'<cr> | \$0F,\$0D | Identify Disk Drive Extended. This will display information about the attached disk allowing for a disk capacity up to 2 TB. | Sends IDDE data block and then <prompt>\$0D |
| 'FWV'<cr> | \$13,\$0D | Get firmware versions | Displays the versions of the main firmware and the reprogramming firmware 'MAIN x.xx'\$0D 'RPRG x.xx'\$0D then <prompt>\$0D |

Table 1—This table shows just a few of the commands supported by the VNC1L. All commands end with a carriage return (0x0D), as do replies.

| Bit | Status signal |
|-----|---------------|
| 0 | *RXE |
| 1 | *TXE |
| 2 | — |
| 3 | — |
| 4 | RXEIRQ |
| 5 | TXEIRQ |
| 6 | — |
| 7 | — |

Table 2—Since the SPI has no status associated with the interface, the user gains access to status through an internal register. Four bits available in this status register mimic the operation of the FIFO's status I/O bits: *RXE, *TXE, *RD, and *WR.

interface. The FIFO interface is simple: 8 data bits, 2 output bits (*TXE and *RXE), and 2 input bits (*RD and *WR). *RXE indicates when the VNC1L wants to communicate with you. *TXE indicates when it's alright for you to communicate with it. Use *RD and *WR for controlling the passing of data.

I was having problems communicating with the VNC1L, so I read over the literature again. I single-stepped through my code to see where things were going bad. When I brought *RD low on the microcontroller, the signal wouldn't go low. Hmm, could I have had a bad I/O? No, when the signal was disconnected from the VNC1L, it wiggled fine. Short on the PCB? No, not that either. Hmm, it looked like the VNC1L inputs were being driven. I shot off an e-mail to the manufacturer and waited.

After days without a reply, I started getting antsy. I pored over all the information I could find on the web site, but I found no further documentation to help. I noticed that all of the applications on the site were using the serial interface. Nothing used SPI or FIFO. The chip was announced at the Embedded Systems Conference (ESC) in Taipei, Taiwan. I found that Don Prowie of DLP Design had given a presentation on the VNC1L in September 2006 at the Boston ESC. After visiting the DLP Design web site, I decided to dash off a note to Don. He immediately responded with, "Nope, no experience with the FIFO interface." I asked if he would kindly share a contact with me and he graciously forwarded my inquiry. Gordon Lund from the home office in the United Kingdom replied with a short note

that he would investigate this.

After looking at my schematic, I found I had the necessary connections in place to redefine the interface as SPI without any rewiring (as long as I didn't mind bit-banging an SPI.) So, while waiting for words from the "Empire," I wrote some code to handle SPI transfers. The SPI is a minimal interface so it is handled a little bit differently. Since there are no signals indicating the status of things, all communication gets channeled through one of two registers (inside the VNC1L). The SPI data is actually 12 bits long, a start bit, a RD/*WR bit, a Data/*Status bit, 8 data bits, and a reply bit. You supply the data bits on a *WR. The VNC1L supplies VNC1L the data bits on an RD. The Data/*Status bit chooses which internal VNC1L register the communications use. The reply bit is always supplied by the VNC1L. It indicates whether the RD data bits are Old/*New or if the *WR data bits were Rejected/*Accepted. Read the status register to get the status bits (see Table 2).

Beta ROM code arrived from the U.K. just prior to executing my application in SPI mode. Getting the new code into the VDIP temporarily took my mind off the rapidly approaching deadline for my column. Reprogram-

ming the VNC1L on the VDIP module with the beta code required one of the USB-to-serial chips FDTI Chip is noted for. Luckily, I had all that was necessary to make this connection (maybe more on this another time). The reprogramming utility worked fast, and I was quickly ready to try some of my code in Feedthrough mode. Having already set up for the SPI, I decided to begin there. This time when I powered up the project I got a sign-on message from the VNC1L. I stuck a stick (flash memory drive) into the USB port. Another message indicated the device was detected. I ran through a number of commands. Everything looked fine. I toggled the switch (in my assembly) to make adjustments in my code for the FIFO interface and reassembled it. Yep! The FIFO seemed to be working! Now for the logging application.

Since there is no terminal or LCD involved with the new application, code operation is verified by the LED activity. LED6 lit up, indicating the VNC1L was coming out of reset. LED2 flashed when the stick was inserted, telling me the VNC1L was interrogating the storage device. Then LED3 came on telling me my code saw the VNC1L's output. I was ready to plug in my data source. When I pressed button 1, LED5 let me know a file had been opened. Timely flashes of LED5 indicated data was being written to the file. I let a few seconds of data get captured before pressing Button 1 to close the file. Good thing I work alone, as I'm sure the sight of my strutting, fist pumping, and audible screams of "Yes!" were quite unprofessional. Hey, if my sports heroes can do their victory dances, why can't an engineer! I was as happy as a clam to see data scrolling on my PC after inserting the stick and opening the FTB00.TXT data file.

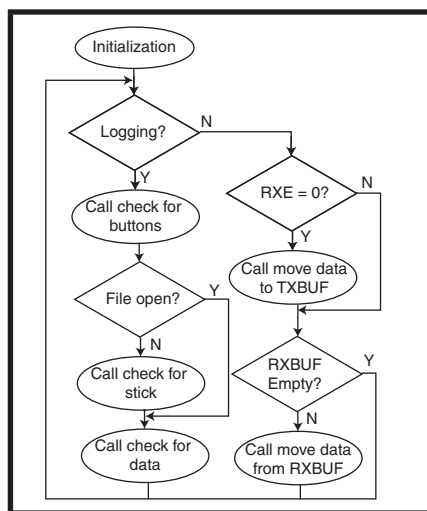


Figure 3—This chart shows the basic steps of this application's support of two modes of operation. Feedthrough mode allows a terminal to send and receive ASCII commands directly to the VNC1L. Logging mode makes use of those commands under the covers to create a stand-alone logging system for data. The VNC1L communication interface supports both serial (SPI) and parallel (FIFO).

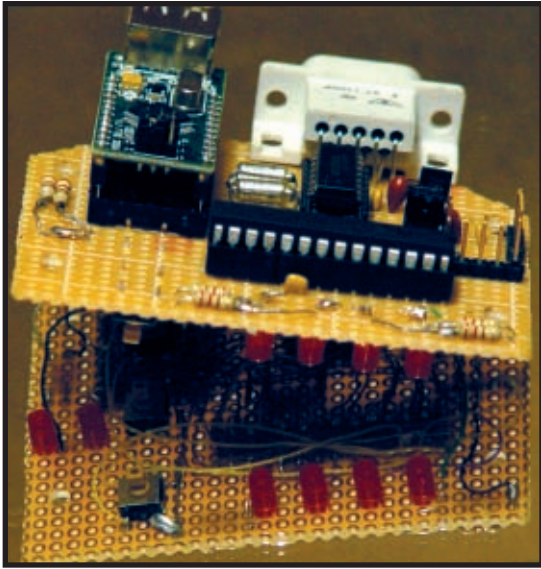


Photo 1—The VDIP module is used in this prototype. It conforms to a 24-pin 0.6" DIP socket. This allows prototyping without the need to handle any SMT parts.

varies; I set up 80-byte ring buffers in the third and fourth banks. This application requires data to be <CR> delimited and no string can exceed 80 bytes (or the head wraps around and destroys data). The application could use hardware handshaking to request that data be paused if the buffer gets full, but I didn't want any data source used with this project to have a handshaking requirement.

One thing you can't program for is the user removing the storage device prior to closing any open files. I suppose you could mechanically prevent the device from being removed or close the file between each store. (This lowers the odds, but it doesn't eliminate error.) Although this application doesn't check for or report free space, all the commands are here to do this. At some point, a real user interface would be necessary. I'd like to see a touchscreen integrated with some small LCDs. This would make integrating a simple user interface much easier. I wonder if the pressure of a touch on an LCD could be detected as some change in the driving signals.

VINCULUM

You can get the VNC1L and VDIP datasheets at www.vinculum.com. This web site (devoted to the VNC1L product) has other modules and applications available to aid the designer.

Interfacing with most modules occurs through serial connections. If you plan to use the FIFO mode, make sure the present revision level of the VDAP firmware is greater than V1.1 (the original version).

I get excited when a product opens up new design possibilities and is easy to use. However, there is a level of frustration that comes with exploring new products, especially when you begin to question your abilities. Manufacturers must be responsive with strong support, or they run the risk of having a design changed to eliminate a part that may actually be superior.

Now I can get back on task.

As spring approaches, there are plenty of paths to take in my quest of exploring electric vehicles. But first, I'd like to congratulate all of the past and present staff of this magazine for the blood, sweat, and tears they've shed to make it the success that it has become. Not many periodicals reach the amazing milestone of 200 issues. Steve Ciarcia knew what readers wanted back when he wrote his "Circuit Cellar" column in *Byte* magazine. Since then, a whole new generation of readers has joined the ranks. Now we all have the opportunity to share ideas and learn from one another, thanks to his understanding of what is important. Thanks for steering us on a straight course, Captain Steve. 🇺🇸

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 at jeff.bachiochi@circuitcellar.com.

SOURCES

VNC1L Embedded USB host controller IC

Future Technology Devices International
www.ftdichip.com

PIC16F737 Microcontroller

Microchip Technology, Inc.
www.microchip.com

TRI-M SYSTEMS

Proudly distributes:

TRI-M ENGINEERING

Battery Backup Module for HESC, V5SC & HPSC Series Power Supplies



Capacity 4500 ma-hr @ 8.4V nominal
Energy backup of 37.8 watt-hr.
PC/104 footprint size.
Available with built-in Real Time Clock

TRI-M ENGINEERING

IDE Flash Drive Carrier Board with Micro SD Interface



Available in 40 and 44 pin header configuration
Support PIO 0-4 and Ultra DMA 3 mode.
Bootable from Transflash/micro SD.
Low power consumption.

TRI-M ENGINEERING

Industrial Relay Board



5amperes @ 250VAC or 30VDC output
3V to 24V AC or DC input
PC/104 compliant, 3.75"W x 3.55"L
20 optoisolated input and 20 output relays

www.tri-m.com info@tri-m.com

1.800.665.5600

HEAD OFFICE: VANCOUVER
tel: 604.945.9565 fax: 604.945.9566



Code Like the Wind World Tour

Finding the right programming language is like selecting a hammer at a hardware store, it has to match your task. Tom introduces the latest version of Express. Perhaps it's the right "tool" for your next job.

Two hundred issues down the line and, with apologies to an old *Saturday Night Live* shtick, I can certainly say: "Silicon has been very, very good to me." I suspect it's been very, very good to you as well.

Software? Now that's another story. For instance, despite more than 10 years of upgrades to Windows, my PC still does the blue-screen boogie-nowhere from time to time. Sure, a PC crash is something folks have come to accept as "normal." It's just an inconvenience, like getting a flat tire on the way to work. But sometimes a flat tire can be a killer—just

ask Firestone.

Indeed, bad software can be a killer—literally. Many of you are probably familiar with the disturbing story of the Therac-25 radiation therapy machine that killed rather than cured. A fundamental design mistake made by the Therac-25 designers was "building a machine that relies on software for safe operation."^[1]

The Therac-25 story is a must-read for any engineer. You'll find a litany of mistakes, including poorly documented spaghetti code full of side effects, race conditions, etc. Other mistakes include piecemeal, ad hoc test proce-

dures and a failure to fully investigate problem reports because there haven't been problems before. An initial presumption that faulty hardware must be to blame is all too common, and usually wrong. Ultimately, as stated above, designing a machine that requires "bug-free software" (an oxymoron) to be safe is asking for trouble.

I made all these mistakes and learned my lessons long ago, thankfully suffering little more than excessive head scratching along the way. How about you? And remember, even if you aren't a software writer, you are a software user. For example, everybody who drives is likely to soon encounter "drive-by-wire" software under the hood of the latest vehicles. Knowing what we know about the myth of "bug-free software," reports of electronic throttles that don't, and "over-active" suspensions are not unexpected.^[2] I figure I'll wait for version 3.1 and let other folks be the beta site.

TOWER OF BABEL

While today's chips are a zillion times better than the originals, software tools have not enjoyed a similar improvement. In short, there's no Moore's law coming to the rescue of software. For example, the "C" language widely used today is little changed from the original defined more than 30 years ago.

Not that it's necessarily fair to blame the tool for the folly of its user. After all, the common household hammer hasn't changed much in cen-

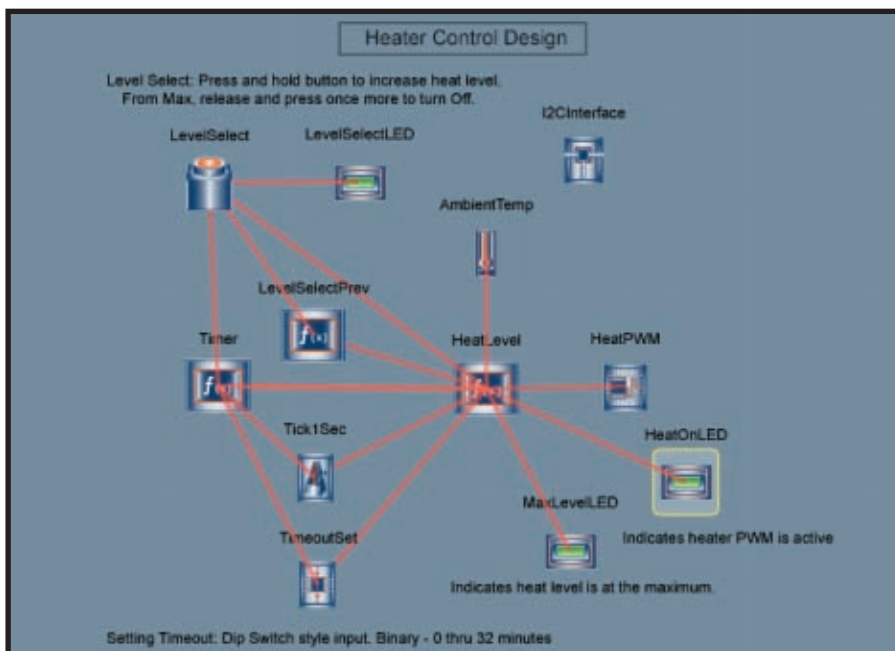


Photo 1—According to Cypress, an Express program (in this case a heater) is a picture that's worth a thousand lines of code.



Photo 2—With four PSoCs and all the trimmings (e.g., an accelerometer, USB and LCD interfaces, and LEDs), the Cypress World Tour demo board has something for everyone.

turies, but nobody blames the hammer when they whack their thumb.

It all raises the philosophical ques-

tion: What is the best programming language? Of course, reality-based imperatives sometimes dictate the answer. For instance, in the PC world, the answer to the “best” language question may simply be whatever Microsoft says it is.

Having dabbled in more than a few languages over the years, I’d like to propose that there is no single language that can be deemed “best.” Indeed, my experience is that languages that try to be jacks of all trades end up being complicated and bloaty masters of none. Rather, I’d say what language is “best”

depends on the task at hand. Next time you’re in the tool department at a hardware store, check out the selec-

tion of hammers. Besides the one you’ve got in your kitchen drawer, you’ll find all manner of specialized hammers (tack, mallet, sledge, nail gun, etc.) optimized for specific tasks.

The main problem I have with software is that it generally remains the province of a priesthood of full-time gurus. This was true of computer hardware long ago, but silicon changed all that with the PC, which gave everyone an entrée to the formerly insular cult of computing. Not so with software. Everybody has a PC, but I daresay few people ever actually program them, at least beyond tweaking a spreadsheet.

Actually, the odd thing about the situation is that it’s understood and accepted that only “computer experts” can do even the simplest thing. Does it make sense that the average person can drive a car, but they can’t program a one-line IF/THEN application?

SOCKETS, SEATS, SOFTWARE

All of these software ruminations

PCB-POOL®

SERVICING YOUR COMPLETE PROTOTYPE NEEDS

Price Example: 16 Sq-Inches (double sided pth)

2 Days: \$ 90.00

8 Days: \$ 22.50

Standard PCB-Pool Service
SIMPLY SEND YOUR FILES AND ORDER ONLINE!

New Service: **WATCH "ur" PCB®**

Save vital time on design errors in advance of receiving your Prototype. View high resolution photographic images of your PCB during each production stage. Be one step ahead, use our realtime PCB monitoring service.

WWW.PCB-POOL.COM

Tollfree USA : 1877 390 8541
sales@beta-layout.com

DOWNLOAD OUR FREE PCB SOFTWARE
www.free-pcb-software.com

INDUSTRY QUALITY
LEAD FREE
Pb, Sn, Ag, Cu, Ni

ROHS / WEEE
conform

Pololu Robotics & Electronics



Robot Kits
Line followers
Robot arms
Hexapods
Chassis

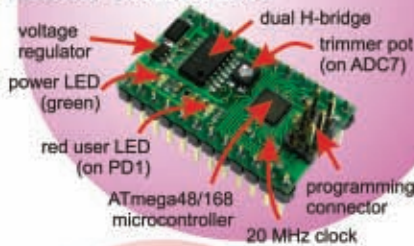
Mechanical Components

Gearboxes, servos
Wheels, ball casters



Motion Control
Servo controllers
Motor controllers

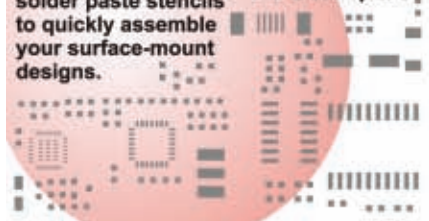
Robot Controllers



Solder Paste Stencils

Use our low-cost solder paste stencils to quickly assemble your surface-mount designs.

From \$25



Custom Laser Cutting

From \$35

Cut your own custom chassis, front panels, and more!

1-877-7-POLOLU
www.pololu.com

6000 S. Eastern Ave. 12D, Las Vegas, NV 89119

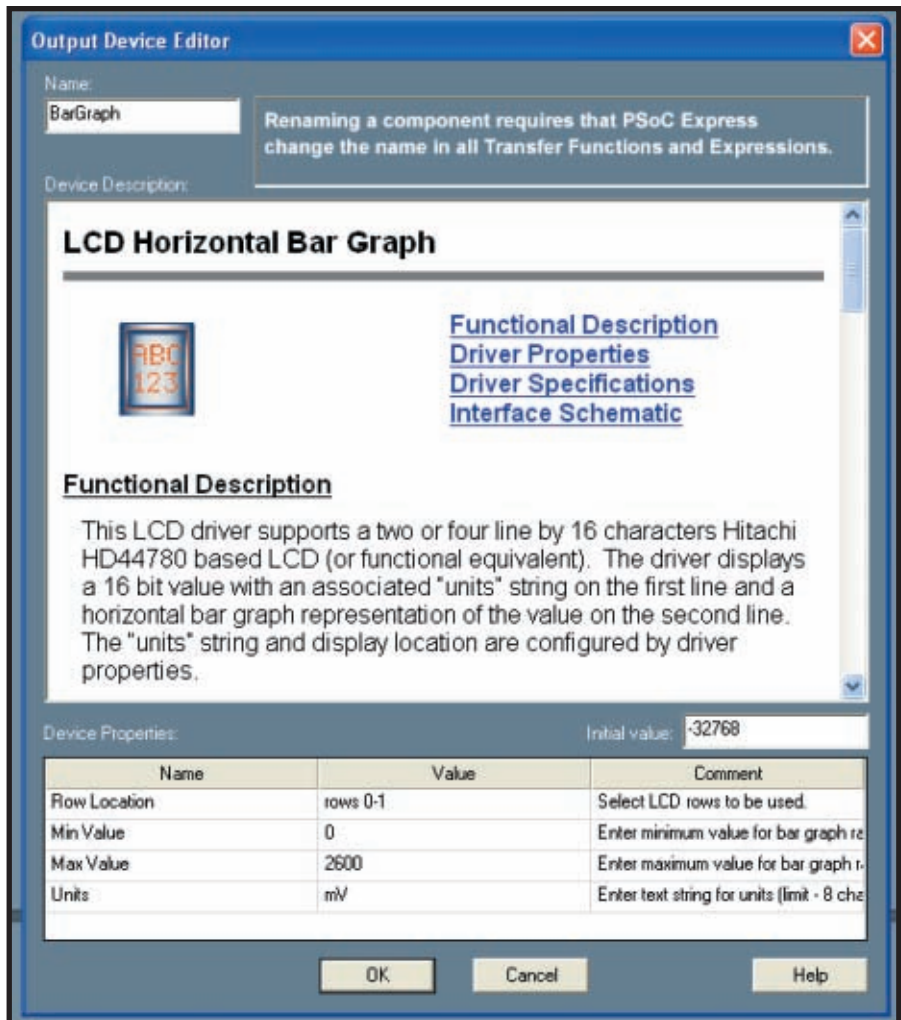


Photo 3—A notable upgrade with the latest version (2.1) of Express is the addition of a number of "smart" I/O elements. Shown here is the LCD Bar Graph that goes beyond simply interfacing the LCD to include a scalable bar graph function.

set the stage for a review of the latest version (2.1) of Express, the visual programming language for Cypress Semiconductor PSoC chips. I covered the PSoC chips when they were introduced more than five years ago (T. Cantrell, "SoC Hop," *Circuit Cellar* 128, 2001). At the time, I was impressed with the way Cypress

incorporated programmable logic to uniquely differentiate their otherwise middle-of-the-road 8-bit flash memory MCU. The icing on the cake is that the programmable logic includes analog capabilities and the PSoC price, typically a buck or two, is right.

The embedded market doesn't switch horses overnight. Even for a standout part, it takes a long time to build awareness, establish credibility, get designed in, and ramp into production. After five years, PSoC is finally getting the respect it deserves. According to Cypress's web site, PSoCs are now shipping at a run-rate approaching 100 million units per year. Sure, that represents just a few percent of the gigantic 8-bit market. But, unit shipments, "sockets" if you will, are only part of the story. Ultimate success and staying power is



Photo 4—The LCD bar graph in action. Note that the function also supports four-line displays, a feature that allows two parameters and bar graphs to be displayed simultaneously.

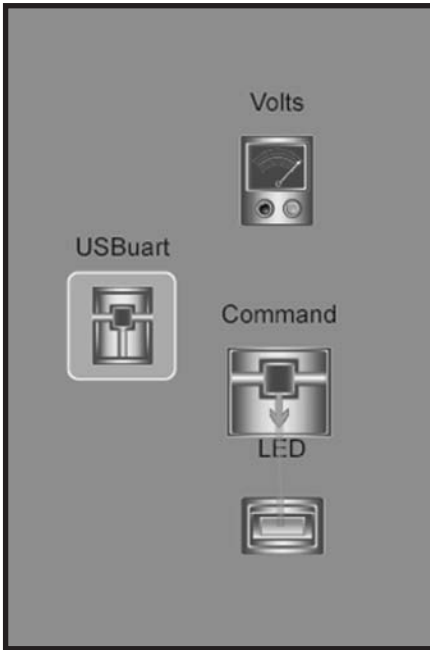


Photo 5—Another smart I/O function in version 2.1 is a USB interface. In conjunction with a Windows driver (generated by Express), it allows a PC to get into the action, in this case reading a voltage on a PSoC input and controlling an LED connected to a PSoC output.

as much determined by “seats,” (i.e., the number of designers using a part).

Cypress says they have 2,000 PSoC customers, and more importantly, the number is growing at an impressive rate, having doubled in just six months.

A couple of years ago Cypress introduced the Express visual programming language for PSoCs, which I covered in “World Beyond Ware” (*Circuit Cellar* 180, 2005). Let’s start by reviewing The Express basics to set the stage for this month’s evaluation of the new version.

Concept-wise, Express uses an “input,” “computation,” “output” model that reminds me of a schematic. A “program” is created by choosing elements from a library and then configuring and connecting them (see Photo 1). The irony that Express uses “pictures” to create software, even as hardware designers are switching to “words” (e.g., VHDL and

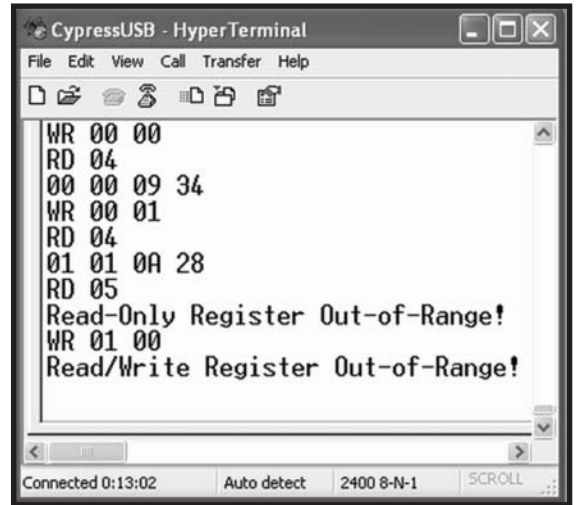


Photo 6—Here’s an example of the USB interface at work using HyperTerminal on the PC side. The “WR” commands toggle the LED, while the “RD” commands sample the voltage input (which I changed between readings). Note also the trapping of error conditions, a situation your PC-side software must deal with.

Verilog) is noted. Have it your way.

The predefined I/O and processing elements are enhanced with a measure of “parameterization” (i.e., the ability to configure certain aspects of their behavior). The more they do, the less

Create your Future at Camosun College



Read about our Students' Success Stories in this Issue!

- Digital Signal Processing
- Embedded Ethernet
- CPLD and VHDL
- C and C++
- Digital and Analog Wireless Communications
- Real Time Operating Systems
- CCNA and A+

Small classes, dedicated teachers, proven reputation. That's the Camosun advantage!



www.elex.camosun.bc.ca inted@camosun.bc.ca

A new era of secure RF

The ultimate in security for RKE and remote control applications. The HS Series transfers the status of up to eight buttons via a highly encrypted transmission.

CipherLinx™ security technology
 Never sends the same data twice
 Secure data authentication
 Up to 8 inputs
 Never loses sync
 80-bit key
 ISE evaluated
 Easy user setup

CipherLinx Technology

"In short, the CipherLinx™ protocol in the HS Series is well-designed and is an excellent choice for applications requiring a secure unidirectional link."
HSF (Independent Security Evaluators)

800-736-6677
 159 Ort Lane
 Merlin, OR 97532

Visit www.CipherLinx.com for more detailed information.

| Offset | Name | Read/write access |
|--------|---------|-------------------|
| 0 | Command | RW |
| 1 | LED | RO |
| 2 | Volts | RO |

Table 1—When a design using the USB interface is built, Express enables PC access with a register map showing the address and access rights (e.g., read-write, read-only) for variables in the design.

you have to. “Building” an Express program not only creates the software, but also a complete schematic, a bill of materials, and a datasheet for the underlying hardware. Development and debug takes advantage of a built-in simulator, reducing the need to “blow and go” every time you change something.

WORLD TOUR

Cypress is embarking on a world tour of seminars to roll out the new version of Express. For those of you who can’t make a seminar in person read along and I’ll take you on a guided tour of some of the highlights.

The world tour board is a fun little gadget with the equivalent of four demo boards built-in (see Photo 2). Starting in the top left, the first PSoC shows off Cypress CapSense capacitive (i.e., touch) sensing scheme, which Fred Eady covered in his column last month (“Embedded Capacitive Touch Applications,” *Circuit Cellar* 199, 2007). Heading clockwise to the upper right, there’s a USB demo. Cypress has been big in USB chips for a long time, and now Express can speak a bit of USB too. The USB port also acts as a handy power supply connection for the board (no wall wart needed) and there’s a 9-V battery option as well. Continuing to the lower right, there’s a PSoC driving a four-digit, seven-segment LED display. Finally, on the lower left, we find a tilt demo using a low-g accelerometer and also an interface for a commodity (i.e., HD44780-type) two- or four-line LCD. Each demo has a potentiometer driving an analog input, a switch, a five-pin header for the cute MiniProg USB flash memory programmer, and a Tokyo-by-night complement of LEDs.

Compared to the first version of Express I covered last time, there have

been a lot of additions and improvements to the I/O and computation building blocks.

For instance, the I/O options before pretty much extended only as far as the pins on the chip. Now, the list has grown rapidly with support for a variety of external devices including sensors such as temperature, pressure, humidity, optical, distance, along with the aforementioned acceleration and CapSense. On the output side, there are a bunch of PWM building blocks supporting different applications (i.e., motor, heater, power supply) with a variety of configurations and power levels (i.e., from V_{DD} at 10 mA, from the PSoC itself to 48 V at 10 A, using an external power transistor).

The new LCD bar graph in Photo 3 demonstrates the value of a “smart” I/O function. Simply getting some data onto the display is one thing, but the function also includes a bar graph feature with built-in scaling capability (see Photo 4). Put it all together and the few seconds it takes to configure the object replaces hours or even days of conventional programming.

Another example of a time and effort saver is the newly added USB “interface” object. Let’s take a closer look at the world tour USB demo to see how that works.

As shown in Photo 5, an “interface” is different than a regular input or output device in that it kind of just sits off to the side without any connections. That’s because it works by exposing all the variables in the design, the so-called “register map,” for access.

One such variable is the “Volts” object, which is an analog (0 to 2,600 mV) voltage controlled by a potentiometer on the board. The fact “Volts” doesn’t appear to be connected to anything in the Express design is a clue that it’s accessed via the “USBuart” interface. The same goes for “Command,” which is a so-called “Interface Valuator.”

In summary, what’s going on is that the PC on the other end of the USB interface can read the “Volts” present on the PSoC analog input and it can also issue a “Command” to control the LED connected to one of the PSoC

outputs. Both actions rely on the PC accessing the corresponding locations in the PSoC, as defined in the “register map” created and shown in the datasheet generated when the project is built (see Table 1).

So just how does the PC do that? The answer is found in the USB driver on the PC side of the equation, which is also generated by Express. It includes a basic command processor that allows the PC to issue commands to write, read, or continuously read locations in the PSoC register map.

Since the commands and results are in ASCII, we can use a terminal emulator to see what’s going on. As shown in Photo 6, first we tell the terminal emulator (e.g., Hyperterminal) to connect to the virtual COM port (i.e., in this case COM7) created when the Express generated USB driver was installed.

Next, you can see I typed in the commands to turn the LED on and off by writing 1 and then 0 to address 0, which maps to the “Command” interface valuator in the design. Along the way I turned the potentiometer to change the voltage input. You can see the resulting change of value returned in the “Volts” register map location (addresses 2 and 3).

The command processor is a bit finicky, with strict rules about formatting. Everything has to be uppercase and all values must be two-digit hex. Furthermore, as shown in the example, your PC-side program needs to be able to deal with error conditions and messages caused by access to an undefined address or attempting to write to a register location defined as read-only.

Joining the I²C and USB interfaces, Cypress has added support for their proprietary Wireless USB 2.4-GHz radio. Although some of the target applications (i.e., wireless keyboard and mouse) and the brand name itself sound PC-centric, there’s no reason Wireless USB can’t be used for a variety of non-PC applications (i.e., wireless sensor) as well.

TOTAL RECALL

One seemingly small addition to Express version 2.1 is a “Delay” valuator. In this day and age when per-

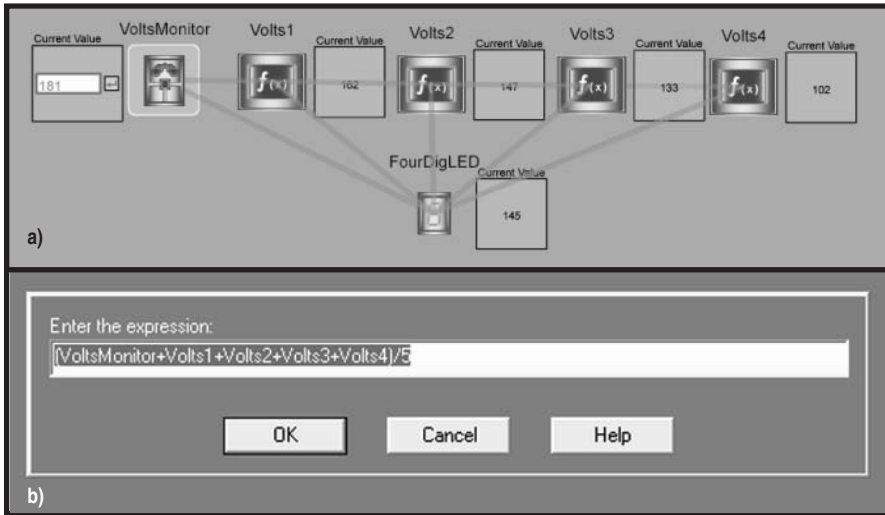


Photo 7—The Delay valuator (a) is a welcome addition to the latest version of Express. It gives designs a sense of history (i.e., data that persists across loop iterations) that's needed for many common functions, such as averaging (b) shown here.

formance is all the rage, it may seem odd that the ability to slow things down is a feature. But, due to the way Express works, the delay valuator is in fact a must-have feature for a variety of applications.

Why is the delay valuator so important? The answer lies in the way Express uses a “loop” structure to run your program. Inputs are captured, valuators are calculated, and outputs updated each iteration of the loop. However, that means there’s no easy way for an Express program to keep track of results from an earlier loop iteration.

That’s a problem, because a sense of history is key for many applications. DSP functions like filtering rely on performing computations on a time-ordered past-to-present stream of data samples. Another example is PID control, which needs to keep track of previous and accumulated errors (i.e., D and I terms, respectively) in order to calculate the new output.

The example in Photo 7 shows Delay at work. As shown in the simulation, the output of each loop iteration is calculated by averaging the last five readings. It’s kind of the software equivalent of a hardware designer’s FIFO or shift register.

EXPRESSION SESSION

I remain intrigued by the Express concept of visual drag-and-drop programming for embedded applications.

And Express 2.1 represents solid progress on a lot of fronts. In particular, the much larger I/O device library stands out for both expanding the application spectrum and simplifying your work. And although I didn’t get into it this time, Cypress now provides ways for users to create their own application-specific Express additions as well as integrate Express designs with conventional (i.e., C, ASM) programs.

Make no mistake: if it leads, it bleeds. Though much improved, Express still has, shall we say, quirks. For instance, going back to Photo 3, notice how some of the text at the bottom right is off the edge of the window, but the window can’t be scrolled or resized (Hint: You can adjust the column width). Similarly, the zoom function is kind of goofy, making it harder than it should be to see what you want where you want it. It seems like half the time I right-clicked on an object to do something with it, I got a pop-up menu from Adobe instead. (Express uses an Adobe-supplied graphics front-end.) Not to mention the fact there are still plenty of “gotchas” documented in the README file. But to be fair, many of the worst offenders from last time, such as the inability to edit a list without completely retyping it, have been fixed.

If anything, complaints about the implementation are fueled by the feel-

ing it’s holding the concept back. I have lots of ideas for improvements. For instance, why not add arrows to connections to make the direction data flow clear at a glance? It might also be helpful to visually indicate the data type (8 or 16 bits) of an object, lest you encounter an unexpected mismatch (which I did and, by the way, the simulator didn’t catch). In addition to a better pan and zoom capability, a macro-like encapsulation feature would ease working with larger designs by hiding complexity and reducing screen clutter.

On the other hand, the temptation to “fix it until it breaks” should be avoided. Express will never be able to handle giant, complex, high-performance, real-time apps. To do so, it would have to lose its identity and itself become a bloaty, complicated tool. Anyways, rocket science applications are kind of a moot point when the target is tiny PSoC chips.

If Express isn’t the perfect solution, so what? Remember the hammer example. No tool is ideal for every application. If you’re hanging a picture, you’re better off using the hammer in the kitchen drawer, not a nail gun. If a simple tool like Express can get your idea to silicon faster and easier, doesn’t that make it the best tool for the job? 📌

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.

REFERENCES

- [1] N. Leveson, “Medical Devices—The Therac-25,” <http://sunnyday.mit.edu/papers/therac.pdf>.
- [2] J. Scheeres, “Teched Out Cars Bug Drivers,” www.wired.com/news/autotech/0,63846-1.html?tw=wn_story_page_next1.

SOURCE

Express 2.1 visual programming language for PSoC
Cypress Semiconductor Corp.
www.cypress.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. Send your disk and digital submission sheet to: IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066 or e-mail adcopy@circuitcellar.com. For more information call Shannon Barraclough at (860) 872-3064.

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

PHYTEC

phyCORE[®] OEMable Single Board Computers

XScale: PXA270, PXA255

ARM: LPC3180 (ARM9); LPC22xx, LPC229x, AT91 (ARM7)

PowerPC: MPC5554, MPC5200B, MPC565, MPC555

ColdFire: MCF5485 **x86:** Elan SC520

C166/XC16x/ST10/8051 **CAN**

Blackfin: BF537

Faster-to-Market: Save time by integrating a PHYTEC Single Board Computer Module into your target circuitry.
Make - or - Buy: Why make your own when you can buy PHYTEC off-shelf solutions, cost-effective to 1000s units/year?

Integrated Support Services: Let PHYTEC assist you in the design of your end product: from tools and RTOSes to production. Our hardware is bundled with leading compilers (Keil, IAR, CodeWarrior), RTOSes (WinCE, Linux) and debuggers.
Immediate Support: Talk to PHYTEC technical staff with every call. No waiting for answers.
Your OEM solution: With 20 years design, production, and integration experience, PHYTEC is your OEM partner.

PHYTEC America, LLC ■ 203 Parfitt Way SW, G100 ■ Bainbridge Island, WA 98110 USA
www.phytec.com ■ (800) 278-9913 ■ www.phycore.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

FAT 12/16/32 FILE SYSTEM

- DOS/Windows Compatible
- USB Flash Disk
- USB Floppy & Hard Disk
- SD/MMC
- CompactFlash, ATA/IDE
- DiskOnChip
- NAND & NOR Flash
- 10KB RAM / 25KB ROM typical
- Low Cost, No Royalty
- Full Source Code

www.smxrtos.com

Micro Digital Inc
RTOS Innovators

800.366.2491 sales@smxrtos.com

USB-ICP & LCD-Demo Kit

USB-ICP Using a standard USB port, USB-ICP supports In-System Programming (both ISP and ICP mode) for Philips 80C51, LPC9xx, and LPC2xxx ARM7 microcontroller families. In-System Programming uses a two-wire serial interface to program and erase ISP enabled microcontroller devices without removing them from the system. This device is powered over the USB connection, so no external power supply is required! **\$69.00.**

LCD-Demo Kit Small footprint "pocket" LCD Demo board with 8-character alphanumeric LCD. Uses a highly integrated Flash based 80C51 device. A single CR2025 3V coin cell battery powers the entire board and the unit is User Re-Programmable via the ICP connector. **\$49.00.**

For XA Development Kits and I2C, MDIO and SPI tools, please visit our website.

www.teamfdi.com
VISA/MC/Amex

Future Designs, Inc.
2702 Triana Blvd
Huntsville, AL 35805
(256) 883-1240
Fax (256) 883-1241

High-speed USB 2.0 + Xilinx FPGA Complete Software API

75mm x 50mm
\$399.95

3.5" x 2.0"
\$199.95

XEM3010-1500P:

- 1,500,000-gate FPGA
- Programmable PLL
- Over 110 I/Os
- 32 MB SDRAM
- Configuration PROM
- 0.8-mm expansion
- \$399.95 / Qty 1
- \$349.95 / Qty 10

XEM3001:

- 400,000-gate FPGA
- Programmable PLL
- Over 80 I/Os
- 0.1" expansion headers
- Business-card size
- \$199.95 / Qty 1
- \$174.95 / Qty 10

FrontPanel Software API:

- Windows XP, Mac OS X, Linux
- C/C++, Python, Java
- Configure FPGA and communicate with your design
- The easiest way to integrate USB into your product
- Use for image capture, control, test equipment, etc.
- Up to 38 MB/s transfer rate!

5% off with Coupon Code: CKTCLR73*

* Valid for first order only. Void 30-days after issue publication.

Visit us online at:
www.opalkelly.com

Full Speed CAN USB Adapter

Simple configuration & use

\$235.00
Qty 1

+1 630-245-1445
Naperville, Illinois USA

www.c-a-n.com

www.can232.com

Only \$108 €89

CAN232 Features:
 Free sample programs
 8-15VDC supply via CAN
 Timeslamp 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 independent
 CE Approved

CANUSB Features:
 Free ActiveX component
 PC, MAC & Linux support
 Both 11 & 29 bit ID support
 Simple CAN logger included
 Free Threaded Windows DLL
 Firmware upgradable via USB
 Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.
 No need for external power
 Works up to 1Mbit CAN
 Supports RTR Frames
 USB 2.0 Full Speed
 Free USB drivers
 CE Approved

Only \$154 €129

www.canusb.com

Instant LCD

part CD-002 - \$189

- Versatile Programmable Module
 - Serial/PC/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

EARTH LCD.COM We Make **LEDs** Work™

Going Wireless is Easy!

Zigbee™

Wireless Mesh Network

NEW USB Zigbee™ Stick offers an easy way to Zigbee™ enable PC's

- ETRX2 USB Based on Ember Single Chip Zigbee™ 802.15.4 Solution
- No need for RF Design Experience
- 2.4 ISM Band

LEMOS INTERNATIONAL www.lemosint.com

Call Toll Free 1-866-345-3667 or email at sales@lemosint.com

New IP Products • \$129.95

IP Power

- 4 independent power outlets
- SDK for complete customization

IP Video

- NEW SDK tools for customization including telnet access, busybox utilities, wget, and more
- New simultaneous 4 channel DVR compatibility

Aruca electronics
www.ArucaElectronics.com

Solve complex signal acquisition problems...

- positioning & control
- environmental
- acceleration
- transients
- pressure
- vibration
- sonar
- GPS

- Linux Driver
- Guaranteed in stock
- Customization available
- 16-bit analog inputs and outputs
- Million sample FIFO eliminates interrupts
- Wide analog input and output ranges
- 40°C to +85°C Standard

www.stx104.com
Apex Embedded Systems
sales@stx104.com • 608-256-0767 x24

Get The Whole Picture

Camera Interface Application Kit for remote monitoring, security and event-capture via the internet.

Kit includes:

- RCM3365
- Serial Camera
- Servos
- Dynamic C®
- Dev. Tools

Buy Now \$499

RABBIT Semiconductor
www.rabbitappkits.com

PCBFABEXPRESS
 High Quality PCBs @ Low Impact Prices

Get 5 PCBs for \$13 each in 5 days

2 layer, 20 sq. in., FR-4, 0.062" thick
 FREE Tooling, FREE Soldermask, FREE Silkscreen

Visit website to see all offers on 2 to 8 layer PCBs

ORDER ONLINE
 408-522-1500
www.PCBfabExpress.com

Also see our fantastic PCB Assembly prices online

IR Buddy's

BACK!

irbuddystore.com

I²C/SMBus

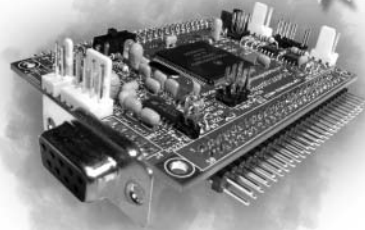
- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools

MCC
 Micro Computer Control

FC is a trademark of Philips Corporation

www.mcc-us.com

Start designing with
Freescale's
advanced S12X family!



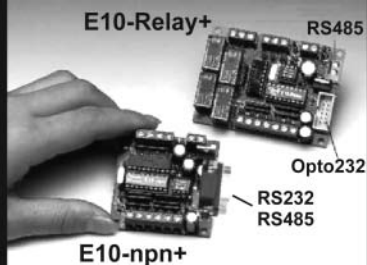
NEW!



Toll-free: 1-877-963-8996
www.technologicalarts.com

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!

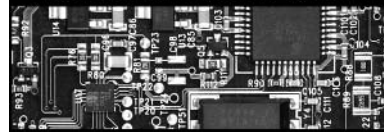
Get Free Ladder Logic Simulator:
www.tri-plc.com/ci.htm



Tel: 1-877-874-7527 - PLC specialist since 1993

ELECTRONICS MANUFACTURING SERVICES

PRO-TECH ELECTRONICS CANADA



IN BUSINESS SINCE 1988

- BOARD DESIGN / PCB LAYOUT
- PARTS PROCUREMENT
- SMT ASSEMBLY (BGA / FINE PITCH)
- CUSTOM WIRING HARNESES
- PROTOTYPES AND PRODUCTION
- WORLDWIDE CUSTOMER BASE

Tel: (905) 760-2185 Fax: (905) 760-2186

E-mail: sales@protechelectronics.com

www.protechelectronics.com

Discover
MCU Space,
a comprehensive
place of
PIC® MCU
tools,
articles,
and links.

visit www.MCUspace.com



PIC is the registered trademark of Microchip Technology Inc. in the U.S. and other countries.

CONTROLLER AREA NETWORK CAN to RS232 CAN to USB

Flash CAN232 - The original RS232 to CAN
adapter. Supports both ASCII and binary
commands. \$99

CANPIC CAN232 - Enhanced CAN to RS232
adapter just \$89

NEW! CANUSB - CAN to USB adapter \$99

Lowest! TrashCAN - Bare bones CAN to
USB adapter as low as \$69.95.

I2C232 - I2C to RS232 adapter only \$89.

I2CUSB - I2C to USB adapter only \$89.

Need a Software Guy?

Over 25 years experience from high volume
automotive components to one-of-a-kind test
boxes. Let EMICROS do your next embedded
software project.

EMBEDDED MICRO SOFTWARE
email: info@emicros.com

www.EMICROS.com

sponsored by ThePokerBug.com

WIRELESS RS-232

WCSC (Willies Computer Software Co)



- Extremely easy to install
and use
- Distances up to 100
meters (329 feet)
- Bluetooth Serial Port
Profile

- Communicates with Bluetooth enabled
PDAs, Laptops, Smartphones, & Computers
- No programming or special software needed

Other Products

- Professional serial communication
libraries & development tools.
- PCI, PCMCIA, USB, Universal PCI,
& ISA multiport RS232, RS422, &
RS485 cards, Bluetooth/USB
Dongles

<http://wireless.wcscnet.com>

sales@wcscnet.com (281)360-4232

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@teletybe.com

ValueCAN

The High Value
Tool For
Controller
Area
Network

- USB to CAN
- Simple software
analyzer included
- DLL with examples
for custom applications
- PC isolated from CAN
- 100% bandwidth at 500Kb

Only

\$295



www.intrepidcs.com

USB Host Stack

- USB 2.0
- EHCI, OHCI, UHCI,
ISP116x, ISP1362, ISP176x,
ARM, ColdFire
- Hub, Mass Storage, Modem,
Mouse, Keyboard, Printer,
Serial Converter
- Low Cost, Royalty-Free
- Full Source Code
- Standalone or RTOS
- Device and OTG Available



Micro Digital Inc
RTOS Innovators

www.smxrtos.com/usb

Big things come in small packages...

- Highlight key product developments
- Introduce new products
- Recruit the best engineers
- Create valuable brand awareness

Reach the right audience, right now with Circuit Cellar's Idea Boxes!



CHINA PCB SUPPLIER

- DIRECT SALE FROM CHINA
- PROTOTYPE TO PRODUCTION

instant online quote
shopping cart ordering system
China competitive prices
free Electrically test



web: <http://www.pcbcort.com>
email: sales@pcbcort.com
Tel: +86-571-87012818
Addr: No. 2 Haining Road,
Hangzhou, P/R China

WWW.PCBCART.COM

Link Instruments

Digital Oscilloscope 500 MSa/s

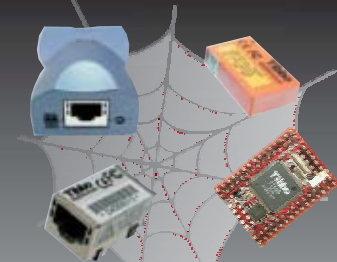
- 2 Channel Digital Oscilloscope
- 500 MSa/s max single shot rate
- 250 MSa/S (Dual channel) 512 Kpts
- 500 MSa/S (Single channel) 1 Mpts
- Advanced Triggering
- Portable and Battery powered
- Only 9 oz and 7" x 3.5" x 1.5"
- FFT Spectrum Analyzer
- USB 2.0

\$950



www.LinkInstruments.com 973-808-8990

Net Modules



programmable
revolutionary

SAVE 5% enter coupon code CCI at checkout

ABACOMdirect.com



tel: +1(416) 236 3858
fax: +1(416) 236 8866

Zigbee

watchout
here comes
Spider
wireless

auto-binding multidrop sensor / actuator
network pods

SAVE 5% enter coupon code CCI at checkout

ABACOMdirect.com



tel: +1(416) 236 3858
fax: +1(416) 236 8866

Wireless I/O

extend your digital I/O's
over 100's of feet up to
tens of miles - in both
directions - with the
16IO-SSRT RF transceiver
modules



SAVE 5% enter coupon code CCI at checkout

ABACOMdirect.com



tel: +1(416) 236 3858
fax: +1(416) 236 8866

MYLYDIA, INC.

Layout Gerber,
Prototype Making

QUICK TURN

PCB & Turnkey
@
The Best Prices

Tel: 800-695-9342

Sales@mylydia.com

WWW.MYLYDIA.COM

R-Box™

Low Cost 16-bit Data Acquisition
and Mass Data Storage



OEM \$99

- 3.0 x 4.0", 50mA standby, 160 mA, 9-24V DC
- Complete C/C++ programmable environment
- 8 16-bit ADC, 8 16-bit DAC, 4-20 mA outputs
- CompactFlash with FAT file system
- 80 MHz CPU, TTLs, Solenoid drivers, Relays, Opto-couplers
- 3 RS-232/485/422, timer/counters, PWM, RTC, EE
- Aluminum box with field removable screw terminals

50+ Low Cost Controllers with ADC, DAC, solenoid drivers, relays, file system with CompactFlash, LCD, DSP motion control, 18 UARTs, 300 I/Os. Custom board design. Save time and money.



1724 Picasso Ave., Suite A, Davis, CA 95616 USA

Tel: 530-758-0180 • Fax: 530-758-0181



www.tern.com • sales@tern.com

NEW

RS232 to TCP/IP

- TCP/com™ v2.0, RS232 to TCP/IP software. Plus TCP/IP to RS232.
- WinWedge™, RS232 or TCP/IP data direct into any Windows app. - Excel, Access, etc.

TALtech

Free 30 day evals at www.taltech.com

8/16-bit EEPROM | Ser. EEPROM | Flash EPROM | GAL / PALCE | Most MCUs | Low Voltage to 1.3V. | DIL dev. w/o Adapter.

Conitec's last generation Galep-4 employs ASIC universal pin technology for each pin of 40 pin ZIF-socket. **9000+ device library** / lifetime free updates. Programs 8/16 bit EPROM'S, EEPROM'S, 0-Pwr RAM, FLASH, Serial EEPROM'S, GAL, PALCE,

microcontrollers such as 87/89xxx, PIC, AVR, ST62, etc. Low voltage devices down to 1.3V. No adapter required for DIL devices. 8 Hrs. operation on battery (AC charger included). Runs **WIN 98,NT,ME, 2000, XP** with Hex/Fuse Editor.

Remote control from other apps, (e.g. VisualBasic). Substitutes high priced universal programmers e.g. ALL-11 (HILO) or LAB-TOOL-48 (ADVANTECH) **Provides virtually matching performance at 1/3 to 1/5 the price.** Info, orders, softwr: **619-462-0515**



9000

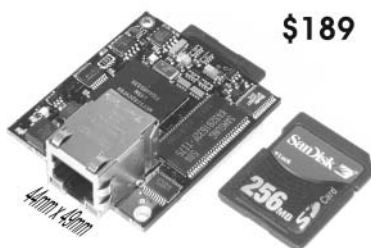
DEVICES



A PALM-SIZED PROGRAMMER HANDLES OVER 9000 DEVICES ?

SMALL PACKAGE. BIG FEATURES.

SALES, SUPPORT / EMAIL: CONTACT@CONITEC.NET / WWW.CONITEC.NET TEL: 619-462-0515. FAX: 619-462-0519



\$189

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

PROTOTYPE CIRCUIT BOARDS

\$85^U_S

 for two 5" x 6" 2-layer boards

Shipped **NEXT BUSINESS DAY** if data is received by **1:00 pm EASTERN**

www.apcircuits.com



AP Circuits

(403) 250-3406

VISA MasterCard staff@apcircuits.com

CAN-4-USB FX

USB to CAN Interface
USB 2.0 Hi-Speed 480Mbps!

Other companies may claim USB 2.0 but if it isn't Hi-Speed it is only 12Mbps.

2007 marks our 11th year of selling CAN interfaces in over 30 countries!

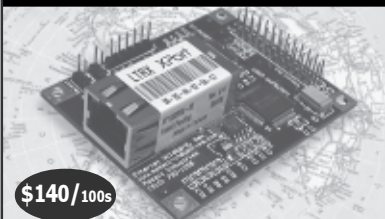
\$249

USD.



Zanthic Technologies Inc.
403-878-2202
www.zanthic.com

EtherSmart Wildcard™ Network - Enables Your Product



\$140/100s

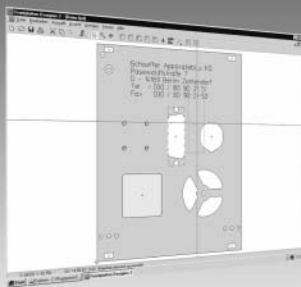
- Standard RJ45 jack hosts 10/100Mbit Ethernet
- HTTP, SMTP, TCP, DHCP, ICMP, and ARP Protocols
- Email program-controlled messages to a specified LAN IP address
- Establish a TCP/IP connection to exchange binary or ASCII data
- Serve software-controlled dynamic content to your web browser



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

Front Panels?

Download the free Front Panel Designer to design your front panels in minutes



Order your front panels online and receive them just in time

www.frontpanelexpress.com

Unrivaled in price and quality for small orders

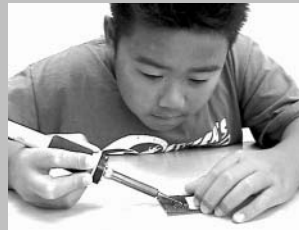
Serial • Ethernet Web Server

Simple, DTE, DCE
No SW Changes



+1 630-245-1445
Naperville, Illinois USA
www.gridconnect.com

gridconnect



ANYONE
Can Now Easily
Hand Solder Surface-
Mount Components!
**Even A 10
Year Old!**

www.schmartboard.com

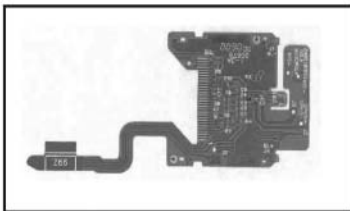


**Weather
Instruments**
for PCs

AAG
electrónica

www.agelectronica.com

*High Quality FPC
Competitive Price*



Start at

\$299
Qty10

WWW.EzPCB.COM

Flashlite 186

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

\$69
QTY 1

- \$99**
Development
System
- Development kit includes:
- Flashlite 186 controller
 - Borland C/C++ ver 4.52
 - FREE Email Tech Support
 - Serial Driver library
 - AC Adapter and cable

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

JK microsystems

C Programmable Controllers & ICs

- Onboard Tiny C interpreter for fast program development
- Multiple program storage
- Free IDE includes text editor, compiler, simulator, serial downloader and test terminal



Full documentation on website

www.ozitronics.com

ALL ELECTRONICS CORPORATION

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

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

PRINTED CIRCUIT BOARDS

QUALITY PRODUCT
FAST DELIVERY
COMPETITIVE PRICING

- 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

PROTOTYPE
through
PRODUCTION

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



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

HIGH PERFORMANCE SOCKETS & ADAPTERS



- Package Converters SOIC/DIP, BGA/QFP, BGA/BGA and more
- 10+GHz Bandwidth BGA/QFN Sockets
- Heat Sinks available up to 100W
- Adapters for probing/test/prototype
- High Volume Production Adapters-BGA, QFP and more
- SMT Package Emulation/Interconnect

Quick-Turn Complex Custom Adapters



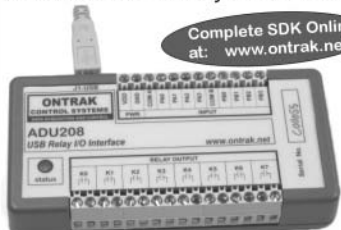
Ironwood Electronics

1-800-404-0204

www.ironwoodelectronics.com

USB Data Acquisition

ADU208 -USB Relay I/O Interface



Complete SDK Online at: www.ontrak.net

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

JStamp!

FAST 32-bit native-execution Java
Executes 3 million Java byte codes/sec
compact 1 x 2 inch DIP40 package
Real J2ME CLDC Java (threads, floats)
2048 KB Flash & 512 KB SRAM

www.jstamp.com

SYSTRONIX
Salt Lake City, Utah, USA
Java is a TM of Sun Microsystems, Inc. JStamp is a TM of Systonix, Inc.

CUSTOM MEMBRANE KEYBOARDS / SWITCHES



- 1 TO 2 WEEKS TURNAROUND
- VERY COMPETITIVE PRICING
- Ex.: (5) 4-switch keyboards for \$395.00
- PCB backed switches
- Custom metal backplates/assemblies
- Electronic assemblies/graphic overlays
- Electronic file transfer capabilities

Picofab Inc.

47808 Blvd. Henri-Bourassa
Charlesbourg, Quebec, Canada G1H 3A7
Tel: (418) 622-5298 • Fax: (418) 622-9996
Email: sales@picofab.net

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

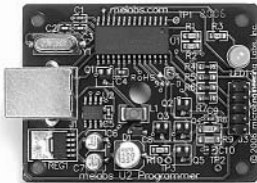
USB Programmer for PIC[®] MCUs

\$89.95
(as shown)

RoHS Compliant

Programs PIC MCUs including low-voltage (3.3V) devices

Includes Windows 98, Me, NT, 2K, and XP Software



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



EPIC[™]

Parallel Port Programmer starting at \$59.95

Serial Port Programmer starting at \$79.95

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

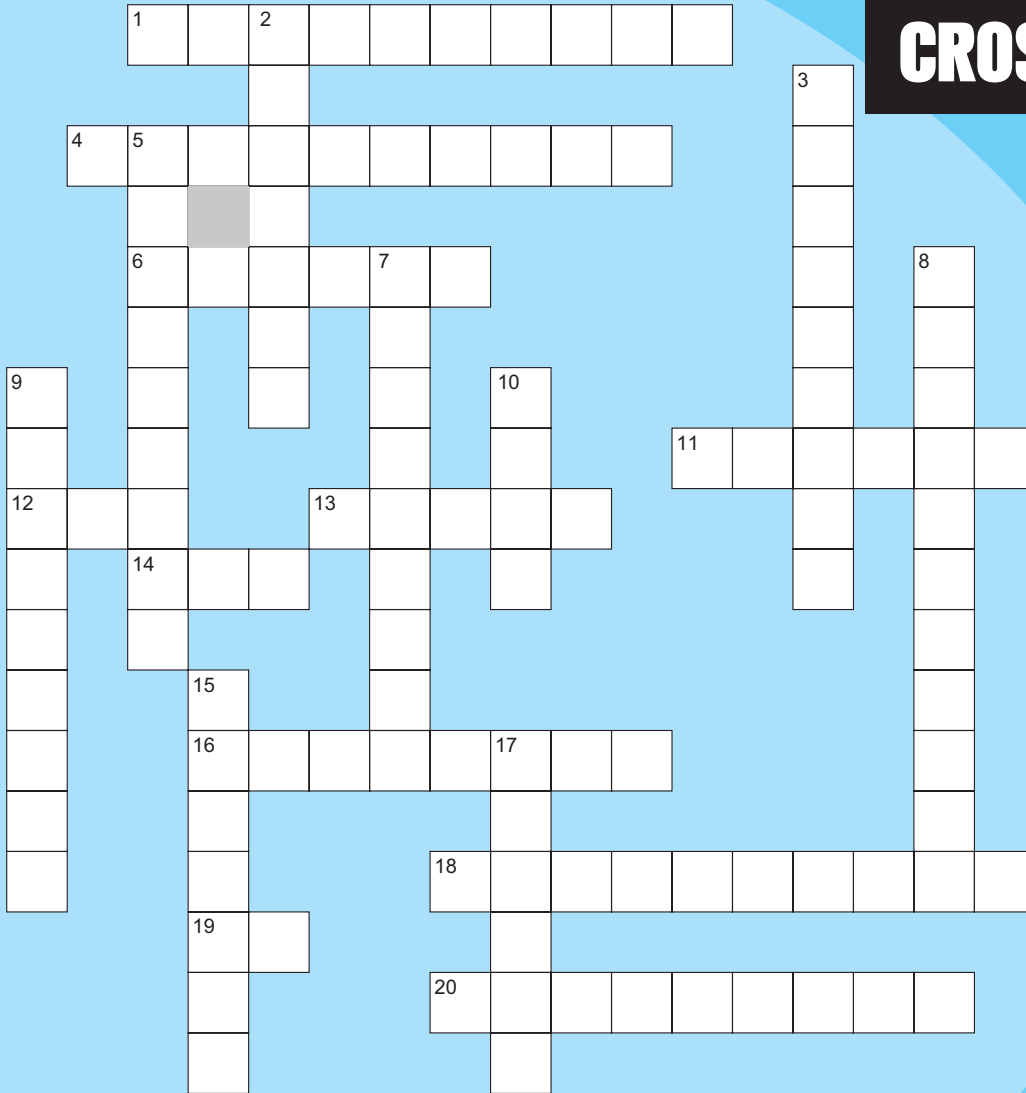
PICBASIC[™] Compiler \$99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO[™] Compiler \$249.95
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

1. A software glitch in this lunar explorer caused all of its fuel to vent in 1994.
4. In this code, common in telecommunications, each bit of data is represented by at least one voltage level transition.
6. Founded in 1998, this service enables you to send and receive online payments.
11. Malfunction
12. Object-oriented programming language developed in the 1980s for embedded real-time applications
13. A two-terminal semiconductor that passes current in one direction
14. Non-return to zero
16. Signals whose range of frequencies are measured from zero to a maximum bandwidth or the highest signal frequency
18. To drive around in search of a Wi-Fi network
19. Quadrature counts
20. American designer (1918–present) who developed coincident-current magnetic storage, which was a precursor to RAM.

Down

2. To convert a file into secret code
3. To put into mechanical action
5. &
7. The size of a signal
8. A pressure-sensitive display used to present and receive information
9. A ball in a socket used for pointing and navigation
10. An association charged with defining protocol standards for the short-range exchange of data over infrared light
15. Sloping; slanting
17. The opposite of digital

The answers are available at
www.circuitcellar.com/crossword.

Brought to you by:

LEMOS

INTERNATIONAL

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 | 73, 91 | ezPCB | 19 | Maxstream | 49 | R4 Systems |
| 90 | AP Circuits | 86 | FDI-Future Designs, Inc. | 87 | Merican Made | 39, 95 | Rabbit Semiconductor |
| 89 | Abacom Technologies | 90 | Front Panel Express, LLC | 86, 88 | Micro Digital, Inc. | 87 | Rabbit Semiconductor |
| 91 | All Electronics Corp. | 37 | Futurlec | 27 | Microchip | 23 | Reach Technology, Inc. |
| 87 | Apex Embedded Systems | 89 | General Circuits, Inc. | 92 | microEngineering Labs, Inc. | 13 | Renesas |
| 63 | Arcom Control Systems | 86, 91 | Grid Connect | 90 | Mosaic Industries, Inc. | 91 | Schmartboard |
| 87 | Aruca Technologies | 33 | HI-TECH Software LLC | 31 | Mouser Electronics | 28 | Sealevel Systems |
| 7 | Atmel | 71 | IMAGEcraft | 89 | Mylydia, Inc. | 11 | SEGGER Microcontroller Sys. LLC |
| 32 | BG Micro | 90 | Intec Automation, Inc. | C2 | NetBurner | 74 | Sierra Proto Express |
| 47 | Bitscope Designs | 88 | Intrepid Control Systems | 3 | Noritake Co., Inc. | 10 | Silicon Laboratories, Inc. |
| 65 | CWAV | 58 | Intronix Test Instruments, Inc. | 69 | Nurve Networks LLC | 92 | Systronix |
| 26 | CadSoft Computer, Inc. | 92 | Ironwood Electronics | 92 | Ontrak Control Systems | 90 | TAL Technologies |
| 83 | Camosun College | 64, 91 | JK microsystems, Inc. | 86 | Opal Kelly Inc. | C3 | Tech Tools |
| 90 | Conitec | 17 | Jameco | 91 | Oztronics | 56, 57 | Technologic Systems |
| 39, 88 | Custom Computer Services, Inc. | 71 | Jeffrey Kerr, LLC | 50 | PCB Design Confrence West | 88 | Technological Arts |
| 1 | Cypress | 5 | Keil Software | 87 | PCB Fab Express | 89 | Tern, Inc. |
| 86 | DLP Design | 69 | LabJack Corp. | 81 | PCB-Pool | 64 | Tianma Microelectronics |
| 71 | Decade Engineering | 69 | Lakeview Research | C4 | Parallax, Inc. | 55 | Tibbo Technology, Inc. |
| 37 | EMAC, Inc. | 87 | Lawicel AB | 86 | Phytec America LLC | 92 | Trace Systems, Inc. |
| 66 | ESC-West | 29, 87 | Lemos International | 92 | Picofab Inc. | 79 | Tri-M Systems, Inc. |
| 87 | Earth Computer Technologies | 2, 89 | Link Instruments | 88 | Pioneer Hill Software | 88 | Triangle Reasearch Int'l, Inc. |
| 74 | Elprotronic | 83 | Linx Technologies | 82 | Pololu Corp. | 32 | Trinity College Robot Contest |
| 88 | eMicros | 87 | MCC (Micro Computer Control) | 88 | Pro-Tech Electronics Canada | 88 | WCSC (Willies Computer Software Co.) |
| 22 | ExpressPCB | 21 | Matrix Orbital | 91 | Pulsar, Inc. | 90 | Zanthic Technologies, Inc. |

Preview of April Issue 201 Theme: Embedded Programming

Java-Based Earthbox Watering System

Reverse-Engineered ECP Bus

ATir Keyboard Interface

Slave Flash Trigger

Generic Modbus Simulator (Part 2): Create a Modbus Master

Build a Three-Axis CNC Mill Machine

ABOVE THE GROUND PLANE Battery Capacity: Charge

APPLIED PCs Uncomplicated RF Communication

FROM THE BENCH Local Interconnect Network

SILICON UPDATE USB: (Wire)Less is More

ATTENTION ADVERTISERS

May Issue 202 Deadlines

Space Close: Mar. 12
Material Close: Mar. 21

Theme: Measurement & Sensors

**BONUS DISTRIBUTION:
Sensors Expo**

Call Shannon Barraclough
now to reserve your space!

860.872.3064

e-mail: shannon@circuitcellar.com

RabbitFLEX

A New Way To Customize



- Click-to-ship in 5 days!
- Pay only for what you need
- Revision friendly
- Perfect for prototype and production



RabbitFLEX™ is an unique build system that gives you the power to develop custom boards without the hassle and the cost. The RabbitFLEX simple-to-use web interface allows you to choose from numerous options such as digital I/O, analog I/O, serial ports, and Ethernet connections on your custom board. Just configure and buy online and our patent pending manufacturing process will deliver your solution in a matter of days. With RabbitFLEX you will reduce design risk, manufacturing cost, and development time.

Start developing now by ordering the RabbitFLEX Tool Kit and your own custom RabbitFLEX board. Take your solution to the next level.

Configure and Buy Online
www.rabbitFLEX.com

Quick Turn Boards Range From
\$149-\$279

RabbitFLEX Tool Kit
\$199

Test Drive RabbitFLEX

Build your custom RabbitFLEX board online. Add a tool kit to your order for a complete development system including Dynamic C®.

www.rabbitFLEX.com



For a limited time with kit purchase.



2900 Spafford Street, Davis, CA 95616 Tel 530.757.8400
Solutions That Work



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Inside the Box Still Counts

This was the title of my very first *Circuit Cellar INK* editorial 19 years ago. Two hundred issues later I still believe it. Like solder being my favorite programming language, I still believe that however appliance-like embedded control and personal computing implementations become, we can't forget that the process of achieving that goal isn't instant. In order to create the sophisticated devices and technologies regarded as off-the-shelf, many people still have to maintain real expertise in rudimentary design skills. Basically, somebody always has to know what's inside the box.

Dave Tweed has done a wonderful job describing both the evolution in technology and *Circuit Cellar's* editorial course during the last 200 issues. Ed Nisley has supplemented it with musings about how wild and woolly some of his projects have been. What is not touched upon is how we all got here in the first place.

Perhaps you've picked it up from my sarcastic editorial comments over the years, but in my opinion the phrase "engineering appreciation" is an oxymoron to most of the business world. They love us when it comes to conceptualizing and designing profitable products and then complain about the costs of maintaining the engineering department after it's done. They love our geeky personality when it comes to working 16 hours a day to solve a design problem at the office, but they refer to us as "hackers" because we like to do the same stuff at home.

It's no secret that I wrote a monthly hands-on design series at *BYTE* for 10 years before *Circuit Cellar*. The great success that *BYTE* and I shared was because we had a mutual appreciation of discovery and invention. Unfortunately, *BYTE* was insanely profitable. I say "unfortunately" because when you start having the revenue from 700-page issues, management gets addicted to the money. The result was that it only took a minor market fluctuation and they all started "tweaking" things to keep their "fix." The biggest "tweak" was changing the entire editorial direction of the magazine from invention and design to "follow the bouncing PC." Seeing *PC Magazine* make lots of money was too much for them and they decided to make *BYTE* into a *PC Magazine* clone. It didn't mean an immediate end for me, but I saw the handwriting on the wall. Ultimately, I was given an opportunity to continue with the magazine, but only if I would review advertiser products instead of writing design articles. I'm sure you can understand my response to that.

I was already at a slow burn around that time, but the final straw for me was at a restaurant when I overheard someone at a nearby table describing how the PC was taking over the world and becoming a high-volume commodity and the embedded controller of the future. As I mentioned a long time ago, the point at which I almost jammed a piece of garlic chicken in the guy's ear was when he said that businesses didn't care what was in the box. The future was commodity computer users. All the rest of the engineers were just hackers (and not the good kind).

That was the last straw. It would be fun to say that I ran out of the restaurant and started a magazine that afternoon, but it actually took about a week. It would also be very egotistical of me to leave you with the impression that I started it alone. *Circuit Cellar* is and always has been a team effort. The people who started *Circuit Cellar* 200 issues ago were Ken Davidson, Ed Nisley, Jeff Bachiochi, Dan Rodrigues, Tom Cantrell, my wife Jeannette, and me. The important point to be made here is how things stand 19 years later. *BYTE* is gone now and just a dusty memory, but like the constancy of *Circuit Cellar's* editorial direction for the last 19 years, every member of this original team is still involved with producing *Circuit Cellar* magazine today.

I look back at some of our early articles and smile at their simplicity, but that's where everyone's experience level was when we started. Today's technology indeed seems more sophisticated, but it is really only the sophistication of the blocks we plug into a design and not the innate intelligence of the engineer that has changed. Blocks that were once just NAND and NOR gates have been replaced with integrated logic and programmable devices. Continually increasing the complexity and function of these design building blocks can only happen if there are people who know what's already there and where we are going. To describe how *Circuit Cellar* fits into that equation, I choose to use the exact same words that I said 19 years ago:

Without a continuous effort at understanding and improving present achievements, we cannot progress to higher levels of achievement. Circuit Cellar is a publication designed to increase that awareness. We must not ignore the fact that it is a combination of people and machines that create intelligent personal systems. Whether they be applied as toaster-like appliances throughout an industry, serve as the control system of a CAT scanner, or function as a video arcade game, the basic ingredients of computers are similar. It is the continual evolution of a computer's concept, design, and application that ultimately results in the perfection of the truly Intelligent Personal System.

Apparently, enough of you agree with this assessment that we're still here 19 years later. Thank you.

steve.ciarciac@circuitcellar.com



100 MHz, 18 Channel, Portable Logic Analyzer

Auto Hardware Compression Captures
from 128K to 30 Billion Samples
@ 10ns resolution

only \$499.00

- Automatic Real-time Hardware Compression can save up to 5 minutes of data @ 10ns.
- Display I2C, Synchronous (SPI), Asynchronous (RS-232), State, Boolean, Bus and Analog Data.
- Edge and Pattern Triggers.
- Pattern Searches with Match, Duration & Animation.
- Specialized Sequential Searches for Serial and State Mode Signals.
- Drag & Snap Markers.
- Dual Waveform View with signal edge snapping.
- Multi-Signal Data Tables with Link Groups.
- Specialized Exports and List Views for Serial Signals.
- Print or Save Images with custom comments.
- USB Powered for Portability.
- USB 1.1 & 2.0 compatible.



Now with Serial Signal Analysis

Download and try the Software!

Cables & Micro-Clips Included

Professional Capture & Analysis Software Included

EconoROM III™



EPROM & FLASH Emulation at its best!
from \$179.00

FlexROM III™



www.tech-tools.com

(972) 272-9392 • sales@tech-tools.com

Copyright © 2006 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

PICmicro® MCU Programmer

Multi-Function In-Circuit & Gang Operation

only \$199.00

- Supports 10, 12, 14, 16 & 18F Series PICmicro MCUs.
- Edit Code, EE Data, IDs and Configuration Fuses.
- Auto or Manual Serialization in Code or EE Data.
- Firmware auto synchronizes to Software version.
- Control Files protect Option Settings.
- Accepts Command-Line Parameters.
- Tests and Monitors Voltages.
- Control Signal for Self-Powered Circuits.
- Single and 4 Gang Adapters available.
- Includes Cables, Software & U.S. Power Supply.



PropellerTM Proto Board



Parallax's **Propeller Proto Board (#32212; \$19.95)** is a low cost, high quality solution for permanent projects using the Propeller chip. To save the user some expense, the USB programming interface is not included on the Propeller Proto Board; we suggest you purchase a Prop Plug (not pictured; #32201; \$29.95) for programming your Propeller Proto Boards.

This board is also ready for use with the Prop Clip (not pictured; #32200; \$29.95), and is also compatible with the USB2SER Development Tool (not pictured; #28024; \$29.95) with the addition of a 4-pin header.

Features of the Propeller Proto Board include:

- P8X32A-Q44 Propeller chip
- 64 KB EEPROM for program and data storage
- LM1086 5V and 3.3V regulators provide up to 1.5 Amps with an input power supply of 6-9VDC
- Accepts the optional VGA, mouse and keyboard interface available in the Accessory Kit – *coming soon!*
- Three-position power switch (off, logic power, power to logic and servo ports)
- Sockets for 4 servos
- Removable 5MHz crystal
- Access to all 32 I/O pins
- Large amount of prototyping area
- Unplated row of holes along perimeter to provide stress relief to off-board connections
- Same mounting holes as the Board of Education (3" x 4") for compatibility with the Boe-Bot[®] robot

Introductory Price
\$19.95
Order today!

Order the **Propeller Proto Board** online (#32212; \$19.95) at www.parallax.com or call the Parallax Sales Department toll-free 888-512-1024 (Mon-Fri, 7am-5pm, PT).

PARALLAX
www.parallax.com