#141 April 2002



THE MAGAZINE FOR COMPUTER APPLICATIONS

ROBOTICS

Cleaning Up With Rovervac

RC-To-Robot Conversion Kit

Robot Simulation Via Rossum

Design A Noncontact IR Bumper



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



Start programming BASIC Stamps with our enhanced Starter kits.



If you're new to the BASIC Stamp, try the BASIC Stamp 2 Starter Kit. We've published a wealth of applications for education, robotics, process control and interfacing to high voltage. A selection of development boards lets you achieve the project quickly often without custom PCB design. The BASIC Stamp 2 Starter Kit includes the BS2-IC module, programming board, BASIC Stamp Manual V 2.0, CD-ROM, serial cable and a component pack.

 BASIC Stamp 2 Starter Kit with Board of Education (#27203 - \$159)

To get more information and see additional BASIC Stamp Starter Kits visit our websit at www.parallaxinc.com.

Purchase a BASIC Stamp 2/2sx/2p24 Starter Kit between now and April 30, 2002, get a 9V 300 mA power supply free!



Want to get started with the ultimate BASIC Stamp? The BS2p24 module has an ample and accessible 16K EEPROM for datalogging and program storage. Interfacing with devices like GPS units, I²C sensors, and LCDs is easy with the BS2p. The BS2p Professional Starter Kit includes the BS2p24 module, the BS2p Demo Board, BASIC Stamp manual, CD-ROM, serial cable and component pack. With the component pack and documentation you will be able to build a time and temperature datalogger.

BS2p Professional Starter Kit (#27235 - \$199)

Order any BASIC Stamp 2/2sx/2p24 Starter Kit through the end of April 2002 and receive a free power supply (an \$8.00 value)!



BASIC Stamp and the Parallax logo are registered trademarks of Paralla







- LCD CHARACTER MODULES LCD GRAPHIC DISPLAYS LCD SERIAL DISPLAYS LCD VIDEO MONITORS LCD PC MONITORS LCD REPLACEMENTS LCD VEHICLE PC LCD TFT DISPLAYS LCD CONTROLLERS LCD BUYERS

EARTH CO. COM "The World Wide LCD Source"

32701 Calle Perfecto - San Juan Capistrano, California 92675 Ph:(949) 248-2333 Fax: (949) 248-2392 lcdking@earthlcd.com

Digital Oscilloscopes



- 2 Channel Digital Oscilloscope
- 100 MSa/s max single shot rate
- 32K samples per channel
- Advanced Triggering
- Only 9 oz and 6.3" x 3.75" x 1.25"
- Small, Lightweight, and Portable
- Parallel Port interface to PC
- Advanced Math options
- FFT Spectrum Analyzer options



DSO-2102S \$525 DSO-2102M \$650 Each includes Oscilloscope. Probes, Interface Cable, Power Adapter, and software for Win95/98, WinNT, Win2000 and DOS.

_ogic Analyzers



- 40 to 160 channels
- up to 500 MSa/s
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- up to 512K samples/ch
- Optional Parallel Interface
 Optional 100 MSa/s Pattern Generator

LA4240-32K (200MHz, 40CH) \$1350 LA4280-32K (200MHz, 80CH) \$2000 LA4540-128K (500MHz, 40CH) \$1900 LA4580-128K (500MHz, 80CH) \$2800 LA45160-128K (500MHz, 160CH) \$7000 All prices include Pods and Software







- 24 Channel Logic Analyzer
- 100MSa/S max sample rate
- Variable Threshold Voltage
- Large 128k Buffer
- Small, Lightweight and Portable
- Only 4 oz and 4.75" x 2.75" x 1"
- Parallel Port Interface to PC
- Trigger Out
- Windows 95/98 Software

LA2124-128K (100MSa/s, 24CH) Clips, Wires, Interface Cable, AC Adapter and Software

\$800

www.LinkIns4.com

Sectron Serial LCDs ...almost too easy!

LCDS have a well-deserved reputation for being a pain in the neck to interface. Their non-standard parallel busses and weird timing are enough to turn your hair gray—if you don't pull it out first. Our LCDs have an easy-to-program serial interface (up to 9600 bps) that makes them a snap to use, particularly with friendly micros like BASIC Stamps[®].

• *Easy hookup and mounting*. We don't stop at simplifying display interfacing and programming. For easy, quick connections, we offer a variety of ready-made wiring harnesses. Need mounting hardware? We've got that, too.

• In stock right now. Hate to wait? Our products are in stock and ready to ship the day you place your order.

• Online satisfaction. Our customers love our web site—it's fast, uncluttered, and jammed with tech data on our products. The online store runs weekly sales with savings up to 40%. Checkout is a breeze with a simple one-page form.



www.seetron.com

Scott Edwards Electronics, Inc.

1939 S. Frontage Rd. #F, Sierra Vista, AZ 85635 phone 520-459-4802 • fax 520-459-0623 web www.seetron.com • email sales@seetron.com

CIRCUIT CELLAR ONLINE

THE MAGAZINE FOR COMPUTER APPLICATIONS

Circuit Cellar Online offers articles illustrating creative solutions and unique applications though complete projects, practical tutorials, and useful design techniques.

This Month (Archive) About Us (Contact) Looking for More? **DO YOU EXCEL IN ELECTRONICS?**

Part 3: The Bottom Line

by Aubrey Kagan

This month, Aubrey concludes his series on Excel by looking at the Solver function, which allows models to be solved using several variables. As an added benefit, he includes a description on the use of regression, and shows us how to look at both linear and quadratic relationships and how the technique can be extended. Join him in this final segment of his series, which has provided insight to GO+ the available functions and shown us how to use them. March 2002

SOFTWARE IMPLEMENTATION OF THE I²C PROTOCOL

by Dariusz Caban

Thanks to the possibility of all-software implementation of the I²C protocol, microcontrollers can communicate with I²C devices. With this article, Dariusz presents us with an example implementation of the Standard mode of the I²C protocol for the popular 8031 microcontroller. March 2002 GOI

EXPANDING YOUR HORIZONS

Using CAD Mechanical Software in Design Projects Lessons from the Trenches—by George Martin

The end of every year, brings the end to many projects for George. So, with this expected slowdown, he fills his time with smaller, more unusual projects. One such project, this year, was an industrial packaging project, which gave him the chance to use the mechanical CAD software he had lying around awhile. March 2002 GO

THE DUST FLIES

Silicon Update Online—by Tom Cantrell

With each passing day, it seems the connection between what goes on in the classrooms of higher education and the real world gets harder to distinguish. Although Tom understands the necessity of fundamental research and development, he's lost on how some of their endeavors really benefit the average engineer. Take a walk with him through those hallowed halls, and see if you can't find something that's less like homework and more like fun. March 2002 GO+

ETHERNET TECHNOLOGY

Part 1: Frames, Collisions, and 10-Mbps LANs Technically Speaking—by James Antonakos

This month, James jump-starts us with a new series on the popular LAN technology, Ethernet. In this first part, he concentrates on the basics, examining all the different Ethernet specifications. But, the story doesn't end there. In the next two parts, he'll also take us through the Fast and Gigabit Ethernet technologies, then round out the series with a look at the operation of hubs, switches, and routers. March 2002

LOGIN/REGISTER

SITE NAVIGATOR

Customize

Knowledge 🕨 Centers

Product Reviews

Data Sheets >

Guides & 🕨 **Experts**

Ask Us

Circuit Cellar

Online Tools

App Notes

Webcasts

Careers

Resources <a>

FAQ

RESOURCES

Each month Circuit Cellar's Resource Links provide helpful links and information on a variety of featured topics.

 Pocket PC by Brant Schroeder

 Logic Gates by Rick Prescott



±

±

A Noncontact Infrared Bumper Tom Baraniak

The Rovervac—A Robotic Vacuum Cleaner Mike Rigsby

Design Logic

RoCK Specifications Part 1: Overview of a Winning Robot Kit *Joseph Jones & Ben Wirz*

Cyber Robotics Experimenting with Robot Simulation James Wilson

Racebot A Two-Degree-of-Freedom Stiquito Robot Scott Vu & James Conrad

APPLIED PCs Replacing Relays with Ladder Logic Part 2: The T100MD-1616+ Fred Eady

- ABOVE THE GROUND PLANE
 Switches and Glitches
 Ed Nisley
- FROM THE BENCH You Too Can Design with SoC A Design Challenge 2002 Primer Jeff Bachiochi
 - SILICON UPDATE Home Is Where the Plug Is Tom Cantrell



6	Task Manager <i>Rob Walker</i> On the Move
8	New Product News edited by John Gorsky
11	Test Your EQ
94	Advertiser's Index May Preview
96	Priority Interrupt <i>Steve Ciarcia</i> Those Insidious Pop-Ups

TASK MANAGER CIRCUIT CELLA

On the Move

'm sure there are plenty of interesting robotics systems involved with factory automation and other stationary tasks, but add locomotion to a robot and the interest factor doubles. It could be a Stiguito that shuffles along a tabletop, a wheeled vehicle that scurries toward the darkest place in a room, or a treaded fire extinguisher that navigates through a model home, robots that move attract attention. Even more attention-getting are robots that move like living creatures in the natural world. Search the Internet and you'll find video clips of everything from robot jellyfish to walking humanoids.

The spindly legs of insects convert well to robot hardware, as you can see by our cover models. (Special thanks to Jim Frye at LynxMotion for providing the yellow H3 Hexapod, Steve Richards at Acroname for sending the large metallic Brainstem Bug, and Jim Conrad for the Stiquitos and nitinol-winged butterflies.) I'd guess that bugs are probably the most common subject for lifelike movement projects, but that doesn't mean that designing a robot version of your favorite insect or arachnid is an easy task. Just ask anyone who has tried!

Of course, not all robots move with the agility of a house cat (as much as I prefer dogs, anyone who has ever seen the tail of an excited Labrador retriever knock over cups, vases, or miscellaneous coffee table items will tell you that it's a natural movement pattern that should never be implemented in Aibo). Some robots are built as practical solutions to everyday problems. The Trinity College Home Fire-Fighting Robot contest provides dozens of examples of just such an approach (see ad on page 45). If you're interested in robots that make life easier, be sure to read about Mike Rigsby's Rovervac project on page 22.

Maybe a vacuum cleaner with a mind of its own isn't for you or maybe you prefer real fish in your garden pond as opposed to robo-carp, either way you have to admit that robotics is an interesting discipline. The excitement of robotics enthusiasts never ceases to amaze me and is evident by the number of submissions that we get for the Robotics issue. I've said it before, but working with the authors and pulling this issue together each year is really quite fun.

I guess it worked out well that such an enjoyable issue should be my last as the managing editor of Circuit Cellar. Thanks to the input from all of you, putting together a quality engineering resource each month for the last three years has been a pleasure. Congratulations to Jennifer Huber on her promotion to the position of managing editor and best wishes to all of you.

rob.walker@circuitcellar.com

MONTHLY ROBOTICS FEATURE!

There's no question that readers enjoy our annual Robotics issue, so we've decided to add more robot-oriented editorial to our lineup! Starting next month, every issue will have a full-length article featuring one of your robot projects. As always, you the readers are invited to submit your project abstracts or articles to: editor@circuitcellar.com.

EDITORIAL DIRECTOR/PUBLISHER Steve Ciarcia

CHIEF FINANCIAL OFFICER

WEB GROUP PUBLISHER Jack Shandle

ACCOUNTANT Howard Geffner

Jeannette Ciarcia

ART DIRECTOR

KC Prescott

Cindy King

Mary Turek

Jeff Bachiochi

CUSTOMER SERVICE Elaine Johnston

GRAPHIC DESIGNERS

EDITORIAL PRODUCTION COORDINATOR Jennifer Huber

TECHNICAL EDITOR Jennifer Belmonte

MANAGING EDITOR

Rob Walker

WEST COAST EDITOR Tom Cantrell

CONTRIBUTING EDITORS Ingo Cyliax

Fred Eady George Martin George Novacek

NEW PRODUCTS EDITOR John Gorsky

PROJECT EDITORS Steve Bedford David Tweed

John Gorsky QUIZ COORDINATOR David Tweed

STAFF ENGINEERS

EDITORIAL ADVISORY BOARD Ingo Cyliax Norman Jackson David Prutchi

Cover photograph Ron Meadows—Meadows Marketing PRINTED IN THE UNITED STATES

ADVERTISING

ADVERTISING SALES MANAGER Fax: (860) 871-0411 Kevin Dows E-mail: kevin.dows@circuitcellar.com (860) 872-3064 Cell phone: (860) 930-4326

ADVERTISING COORDINATOR

Valerie Luster (860) 875-2199 ADVERTISING CLERK Fax: (860) 871-0411 E-mail: val.luster@circuitcellar.com

E-mail: sally@circuitcellar.com

Sally Collins (860) 875-2199

CONTACTING CIRCUIT CELLAR

Fax: (860) 871-0411

SUBSCRIPTIONS:

INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com To Subscribe: (800) 269-6301. www.circuitcellar.com/subscribe.htm. or subscribe@circuitcellar.com

PROBLEMS: subscribe@circuitcellar.com

GENERAL INFORMATION

TELEPHONE: (860) 875-2199 Fax: (860) 871-0411 INTERNET: info@circuitcellar.com, editor@circuitcellar.com, or www.circuitcellar.com EDITORIAL OFFICES: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066 NEW PRODUCTS: New Products. Circuit Cellar. 4 Park St., Vernon, CT 06066

newproducts@circuitcellar.com AUTHOR CONTACT

E-MAIL: Author addresses (when available) included at the end of each article.

For information on authorized reprints of articles, contact Jeannette Ciarcia (860) 875-2199 or e-mail jciarcia@circuitcellar.com.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) and Circuit Cellar Online are pub lished monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751, Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate USA and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. Two-year (24 issues) subscription rate USA and possessions \$39.95, Canada/Mexico \$55, 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 Celtar@ 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 @ 2001 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar and Circuit Cellar INK are registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

1Zh



Wireless Modems

XStream[™] Wireless OEM Modules 24 GHz Available Now

Long Range Low Power Networking

9XStream[™] 900 MHz Modules

-110 dBm 7 mi line-of-sight 20 mi w/ gain antenna 1/4 mi city-urban

Tx 150 mA, Rx 50 mA Power Down <10 μA

Point-to-point Point-to-multipoint Parallel networking 65,000 addresses 24XStream ™ 2.4 GHz Modules

-105 dBm 1.4 mi line-of-sight 4.5 mi w/ gain antenna 1/8 mi city-urban

Tx 240 mA, Rx 60 mA Power Down <10 µA

Point-to-point Point-to-multipoint Parallel networking 65,000 addresses

Same form factor allows your OEM products extended range domestically with the flexibility to deploy them internationally!



NEW PRODUCT NEWS

SMALL PACKAGE SIMPLIFIES MOTOR CONTROL

The ICON H-Bridge provides easy connectivity to a wide range of medium-power brushed DC motors. An on-board microcontroller monitors current and temperature to provide protection if either trip point is exceeded. Various configuration registers may be monitored and programmed via the serial interface. These registers contain information such as the system load current, board temperature, over-current trip point, over-temperature trip point, and status information.

In Serial mode, the H-Bridge is enabled/disabled via a serial command set. In Direct Drive mode, the serial input line becomes the H-Bridge enable control pin, while the serial output can be used to monitor the H-Bridge status. In both operating modes each MOSFET in the H-Bridge is controlled individually by a control input (four MOSFET control lines in all).

The ICON H-Bridge is designed to operate with motor voltages up to 40 V, and can carry continuous currents up to 12 A. This robust H-Bridge handles peak currents of 25 A and can operate to 85°C.

Connectivity is simplified to two receptacles capable of top or bottom insertion of $1'' \times 8'' \times 0.156''$ headers. With the small size and versatile interface capabilities of the ICON,

it meets the requirements of many DC motor interface designs.

Pricing in single unit quantities is \$100.

Solutions Cubed (530) 891-8045 Fax: (530) 891-1643 www.solutions-cubed.com

IF DIGITIZING IC

The AD9874 integrates the IF-to-digital conversion process onto a single chip that can be used to design a highly-sensitive, low-power receiver. The AD9874 MxFE integrates all the functional blocks (excluding the LO and VCO) needed for IF-to-digital conversion in a narrowband superheterodyne receiver. This partitioning reduces

IF filtering, enhances demodulation accuracy and delegates all modulation-specific functions to an external digital signal processor.

The AD9874 MxFE digitizes low-level IF signals from 10 to 300 MHz with a bandwidth of up to 270 kHz. Typical noise figure is 8.7 dB SSB NF and linearity, as measured by the third-order input intercept point (IIP3), is 0 dBm IIP3. The AD9874 is notable for its power efficiency, 21.2 mA versus 45 mA for previous genera-



Edited by John Gorsky

LOW-COST DATA ACQUISITION UNIT WITH USB

The Windmill 750 is a USB-based data acquisition system that provides 16 12-bit analog inputs, 16 bits of digital I/O, and up to eight counters. Up to eight 750s can be connected to one computer to increase the number of channels available.

The analog inputs accept ±10 V signals and have a conversion speed of 80 samples per second. The 750

uses an integrating ADC to reduce errors caused by noise. The digital I/O can be read and set individually or in groups of eight. The package



includes the Windmill 5 software suite and lifetime technical support. With the software, you can log data to disk, chart data in real time, switch digital outputs, use Excel or other Windows-based software for instant analysis, and add optional software modules as the need arises. No programming is necessary.

The eight 16-bit counters can each count up to 65535. You can reset a counter at any time from the software. The counters can be used in Resetting Count or Accumulating Count mode, and can set a scale and offset factor to the count from the software.

The Windmill 750 package costs \$420.

Windmill Software Ltd. 44 (0) 161 833 2782 Fax: 44 (0) 161 833 2190 www.windmillsoft.com

ANALOG DEVICES

AD9874

CLKOUT

tions. The signal chain consists of a low-noise amplifier, mixer, bandpass sigma-delta ADC, filters, and AGC circuitry capable of 12 dB continuous gain adjustment. The range and inherent anti-aliasing allow the AD9874 to cope with blocking signals 80 dB stronger than the desired signal. The AD9874 is fully programmable, providing

> register-based digital control of AGC and filter parameters, amplifier gain and bias currents through an SPI port.

The AD9874 is priced at **\$19.95** per unit in 1000-unit quantities. Production is planned for third quarter 2002.

Analog Devices, Inc. (800) 262-5643 Fax: (781) 937-1021 www.analog.com

NEW PRODUCT NEWS

WEB-ENABLED PC/104-PLUS PENTIUM SBC

The **PPM-TX** is a PC/104-Plus Pentium SBC with a -40 to 85° C operational temperature range. Included on the PPM-TX are four serial ports and a 10/100 Ethernet controller. The board is small, measuring only $3.6'' \times 3.8''$ (90 × 96 mm), and will run PC-compatible software for space- and power-limited embedded designs.

The 166-MHz Intel Tillamook Pentium CPU operates over the industrial temperature range without requiring fans or forced air-cooling. An internal floating point processor and 32 KB of on-chip cache improves it overall performance. The processor allows the PPM-TX to be

compartible with the installed base of applications for MS-DOS, Windows, Linux, and other PC-compatible operating systems and executives.

The companion Intel 430TX PCI chip set contains the core logic to provide PC-compatibility plus the I/O and bus interface logic including the keyboard controller, 16-channel interrupt controller, and real-time clock. Onboard peripherals include DMA controllers, three 16-bit counter/timers, two interrupt con-



trollers, keyboard controller, speaker port, and battery backed real-time clock. Addititionally, four independent RS-232 serial channels, USB port, parallel line printer interface, floppy disk controller, EIDE hard disk interface and mouse port are provided. A precision power-fail reset circuit and watchdog-timer are onboard for remote and unattended applications.

The PPM-TX supports 32, 64, 128, or 256 MB of SDRAM. It is field upgradeable since it is mounted in a standard SODIMM socket. An onboard 32-pin socket supports up to a 288-MB M-Systems DiskOnChip flash

memory device for use as a sold state disk. This offers a viable alternative for fragile floppy and hard disk drives in harsh environmental applications.

The single quantity price of the PPM-TX-166-0 is **\$625**. A 266-MHz version of the board is also available.

WinSystems, Inc. (817) 274-7553 Fax: (817) 548-1358 www.winsystems.com

Graphic Display Integration Made Easy!

"Easy GUI may be the answer to your prayers for a simple solution for middle-of-the-road [size] LCDs." — Tom Cantrell, Circuit Cellar Magazine



"Easy GUI saved me months of hardware design and programming effort..." - Mark Doble, President, MD Designs, Tampa, FL

Find out why people are talking about our technology... Order a Starter Kit Today!

©2002 Easy GUI is a Trademark of Amulet Technologies. U.S. and Foreign Patents Pending. Fully-integrated LCD Modules – with backlights, touch screens, and graphical user interface (GUI) chips

Dedicated GUI Chip – controls the LCD and manages the GUI— replaces traditional LCD controller and frees up your micro!

Includes Widgets – easily create bargraphs, lineplots, function buttons, numeric fields, text strings, and check boxes just to name a few!

Full VGA Support – chip and development tools work with any monochrome, single-scan LCD, up-to full VGA resolution (640x480)

Three Hardware Paths – LCD modules, controllerboards, and GUI chips – all with serial interface!

Now Available... 3.8" and 5.7" backlit modules with touchscreens!



Amulet Technologies GUI Engines For Embedded Systems

GUI Engines For Embedded Systems Visit us at www.GUIchip.com or call (408) 244-0363 today!

MORE POWER ...

The New Basic Micro Atom, offers more in the same package you are already familiar with. Built-in features like upgradeable firmware, Analog to Digital, Hardware Serial, In Circuit Debugging, 32 Bit Math, 32 Bit Floating Point Math and more !

The Atom includes powerful expanded BS2 syntax compatible software with commands such as; OWout, OWin, ADin, LCDwrite, If..Then..Elseif..Else..Endif, LCDread, HPWM, Hserout, Hserin, Do..While, Repeat..Until and more !





POWERFUL SOFTWARE

Easy to use multi document interface. Open as many files as you want all in the same interface. Easily navigate your files with the built-in explorer window.

IN CIRCUIT DEBUGGING

Watch your program run step-by-step as the Atom executes each line of code. Easily debug your code. Bring your programs to life quicker than you ever thought possible !

WATCH IT COME TO LIFE

No more strategically placing debug statements throughout your program! Not only can you watch your program execute live, you can keep an eye on every variable and ram location as they are affected by your program. Each program variable is displayed in binary, hexadecimal and its real values.

FEATURES

- Graphical Debugging (ICD)
- More Software Commands
- 32 Bit Integer Math
- 32 Bit Floating Point Math
- BS2p Compatible Syntax
- Expanded BASIC library
- Firmware upgradeable
- Unlimited Support
- Free Software Updates
- Plus More!

- Over 300 Bytes of RAM
- 256 Bytes of Data EEPROM
- External & Internal Interrupts
- 3 10 bit A/D Converters
- 2 Capture and Compare
- Internal Adjustable Vref
- + / External Vref
- 3 Timers, 2 PWM
- UARTS, I2C, SPI
- Plus More!

DEVELOPMENT KITS



Kit Includes: Development Board, Atom, Manual, Cable and power supply

Only \$149.95

BASIC Micro inC Microcontrollers Made Easy™ TO ORDER VISIT WWW.BASICMICRO.COM OR CALL US AT 734-425-1744 M-F 10 AM TO 6 PM EST.

The Basic Atom is a registered trademark of Basic Micro Inc.

CIRCUIT CELLAR

Problem 1—The initial contents of the 4-bit serialin/parallel-out right shift register are shown in the given figure. Give the contents of the register after three clock cycles. Does this produce a maximallength sequence?



Contributed by Naveen PN

Problem 2—Consider a memory system with the following parameters:

Tc = cache access time = 100 nsTm = main memory access time = 1200 ns

If you would like to have effective (average) memory access time equal to or less than 20% higher than cache access time, what should be the minimum cache hit ratio?

Contributed by Naveen PN

Test Your EQ

Problem 3—In the circuit shown below, the switch (S) is open for a long time and closed at t = 0. What is the value of *I* just before and just after the switch is closed? What is the time constant of the current decay?



Contributed by Naveen PN

Problem 4—What is the difference between a microprocessor and a microcontroller?

Contributed by Dave Tweed



RS232/RS422/RS485 Converters

CMC's low cost converters adapt any

RS422 or RS485 including reliable high

any RS232 link up to 5000 feet.

Completely transparent to the system;

no software changes of any type are

RS232 TO RS422

between RS232 and RS422

Use as a short haul modem

Plug in and go. No software

WWW.2CMC.COM

Converts bi-directionally

changes required

AD422L signal powered

AD422-1 for 110VAC

CmC

necessary.



- Supports up to 40 RS485
- devices RS232 port for RS422 or RS485 • Automatically determines data direction. operation. These converters provide your
- Signal powered version RS232 device with all the advantages of available

speed operation (up to 200 kbaud) and ADA485 (requires 9VDC) \$79.00 data transmission distances up to 5000 ADA485-1 for 110VAC 89.00 feet. Two AD422s can be used to extend ADA485L signal powered 84.00

RS232 TO RS485 4 wire

- Converts an RS232 port for use with RS422 or RS485 devices
- Supports up to 40 RS485 or **RS422 multidrop devices**
- Adds multidrop capability to **RS232 devices**
- Automatically determines data direction.

AD422 (Requires 9VDC) \$79.00 89.00 ADA425 (requires 9VDC) \$89.00 84.00 ADA425-1 for 110VAC 99.00

> Mention this ad when you order and deduct 5% Use Visa, Mastercard or company purchase order



Connecticut microComputer, Inc. PO BOX 186, Brookfield,CT 06804 (203)740-9890

Fax:(203)775-4595

design and testing of vehicle network components CAN and more! The AVT-718 does it all! Supports all popular protocols.

Products to support the

- · CAN (2-wire and GM Single Wire CAN).
- · J1850 VPW (GM Class 2 & Chrysler). II850 PWM (Ford SCP)
- Keyword Protocol 2000 (and ISO 9141).
- · UBP (Ford).

Advanced Vehicle

Technologies, Inc.

· RS-232/422 to host computer.

See our web site for a complete listing of available products and information.

AVT's Products...

- AVT-715: Dual J1850 Interface, CCD optional (RS-232/422).
- AVT-716: Multiple Protocol Interface: J1850 VPW, PWM, ISO-9141, KWP2000,
- CCD, ALDL (RS-232/422). AVT-717: UBP & CAN Interface: Ford's UBP,
- CAN 2-wire & single wire (RS 232/422). AVT-921: Dual J1850 Interface (ISA Bus).
- AVT-931: Dual J1850 Interface (PC/104). AVT-932: Dual J1850 Interface to Ethernet
- (PC/104).

AVT's Services...

- Automotive Network Engineering.
- Customized firmware for our products.
- · Custom hardware and firmware development.

1509 Manor View Road • Davidsonville, MD 21035 (410) 798-4038 voice • (410) 798-4308 fax e-mail: support@avt-hq.com • home page: www.avt-hq.com



CIRCUIT CELLAR®

FEATURE ARTICLE

Tom Baraniak

A Noncontact Infrared Bumper

A robot that can't "see" where it's going isn't very helpful, so Tom took the common IR LED method of proximity sensing, added a PIC12C672-04 to the mix, and came up with an effective pair of eyes that works great for robots or any other objectsensing application. nless you want your robot playing bumper cars with its surroundings, you will have to include some sort of noncontact proximity sensor. One of the most common methods operates by detecting the reflection of transmitted infrared (IR) light from the surface of a nearby object. Various methods are available to implement this, including applications of television remote control receiver modules, LM567 tone decoders, new IR bumper modules made by Sharp, and a simple LED and phototransistor.

This article describes still another option, which takes advantage of some of the advanced features of an 8-pin Microchip PIC microcontroller and offers low cost, flexible mounting, high noise immunity, and low power consumption. Photo 1 shows an example of how the optoelectronics can be mounted in the plastic structure of a robot eye.

THE BASIC SENSOR

A simple IR bumper could consist of an IR LED, phototransistor, and comparator, as shown in Figure 1. Transmitted IR light reflected off an object or surface causes the phototransistor to conduct, generating a voltage drop across the 1-k Ω emitter resistor. When this voltage exceeds the threshold level of the comparator, the comparator's output toggles. One obvious problem with this is that the ambient light level may increase enough to cause the phototransistor to conduct without the reflected IR light. Light sources include sunlight, room lights, reflections, or television remote control IR.

For this method to be effective, the threshold must be set high enough to ignore all ambient light sources yet low enough to detect the reflected IR light. And this must be true at all times or momentary bright flashes would cause false transient bumper alarms. This, then, requires the LED to generate enough IR light to overwhelm the background light levels and overcome the high threshold of the comparator. Producing the continuous intense IR light requires considerable current, which also generates heat and reduces the lifetime of the LED, not to mention quickly draining the battery. As a practical matter, unless the reflecting surface is fairly close, this is not a viable solution.

DYNAMIC THRESHOLD

The problem with the above example is that the reflected IR must exceed all expected ambient light levels. The difficulty is determining what the ambient light level will be so the comparator threshold can be set.

But, what if you could monitor the ambient light and adjust the threshold based on its level? That is the basis of this PIC IR bumper system. The ADC of the PIC monitors the voltage of the phototransistor that's caused by ambient light with the IR LED off. The LED is then turned on and the phototransistor is measured again. If an object is nearby and reflects the IR, the phototransistor will see this as a pulse added to the background level (see Figure 2). When the reflected amount exceeds a software-controllable threshold, valid object detection occurs. An added benefit to this scheme is that because the IR LED is pulsed, the average power consumption is low even though the LED current pulse can be substantial (over 100 mA).

The phototransistor conducts in relation to the amount of light striking it. At some light level, the photo-

CIRCUIT CELLAR®

transistor becomes saturated and any additional light won't be detected. The point at which this occurs can be controlled to an extent via the phototransistor's emitter resistor or with an optical attenuator. However, as the value of the emitter resistor decreases, current through it and the phototransistor increases and can get excessive when the light is bright.

As a practical matter, proved after much experimentation, that doesn't happen except under certain rare circumstances. One such instance is if a bright light source such as an AC line powered incandescent bulb or a flashlight is within a couple feet of the phototransistor and lined up directly in front. If this is the case, there are no objects between the light source and robot. This condition is detectable in the software, which can then provide a warning signal because you don't want the robot blinded by the light or crashing into it. Shielding the phototransistor can also reduce the susceptibility of the phototransistor to a light source by blocking light that isn't nearly exactly lined up with the phototransistor.

Bright light reflected off surfaces hasn't been observed to be a problem with emitter resistor values of 1 k Ω or less. Also, the software compensates for any varying light levels such as caused by AC line frequencies, sunlight changes, television remote controls, and the movement of the robot itself.

In summary then, for most practical applications the sensors will be unaffected by external light sources. The device can detect the one unique circumstance when it is a problem. Then, the robot's main controller can determine how to proceed in concert with other sensors.

SHORT-RANGE IR BUMPER

Figure 3 shows the schematic for a complete two-channel IR bumper system with a sensing range in excess of 8" depending on the reflecting surface, ambient lighting conditions, and the IR LED drive current. Note that it consists of only the PIC microcontroller, two IR LEDs, two phototransistors, a MOSFET switch, and some ancillary resistors and capacitors. Given that most IR bumpers are low



Figure 1—When sufficient IR reflects off the surface, the phototransistor conducts enough voltage to trip the comparator. However, light levels may become high enough to cause it to conduct without reflected IR light.

to the ground and there aren't too many sources of bright lights at that level that could cause sensor saturation problems, this simple circuit will satisfy many bumper applications.

The phototransistors connect to analog input channels AN0 and AN1. The appropriate emitter resistor (R_{μ}) adjusts the light-controlled current gain (see Table 1). The two LEDs, one for each phototransistor, are connected in series and driven by the MOSFET switch. The series connection is a more efficient way to drive the LEDs because more power goes to light output and less to waste heat in a resistor. I used a MOSFET because it has a lower voltage drop and its gate drive requires essentially no current as compared to that of a bipolar transistor, and is therefore more efficient. It also can be directly connected to the output of the PIC, thus eliminating one resistor. You can replace it with an NPN transistor, such as a 2N3904, along with its current limiting base resistor.

Alternately, a PIC output port could directly drive the two series-connected IR LEDs with reduced output power and sensing range. At a forward voltage drop of about 1.3 V and a 47- Ω current limiting resistor, the current driving the LEDs is about 50 mA. The output port source and sink drive current is rated at 25 mA but can exceed this at the cost of out-of-tolerance voltage specifications. This doesn't matter because an LED is being driven as opposed to a logic gate. Heating effects on the PIC are minimal because the duty cycle of the LED is low. So, the PIC is capable of supplying in excess of 25 mA for short pulses. However, the

resistance of the output port's driver MOSFET and the forward voltage drop of the IR LEDs limit the output current to not much more than 50 mA. Using the directly driven LEDs as described with the resulting reduced sensing range the average current required is typically less than 10 mA. Note that it is necessary to eliminate the 22- Ω resistor in series with 5-V PIC power supply when driving the LED directly.

Similarly, an active-low output port can drive the LEDs if you change the output polarity in software. The LED anodes should connect to a 5-V supply separate from the PIC and have a bypass capacitor between the 5-V connection and the ground pin of the PIC.

Phototransistor saturation produces a constant DC level, typically greater than 4.3 V depending on the value of resistor R_E . This compares to approximately 0 V during normal, non-saturated operation. If the voltage remains constant at approximately 4.3 V even with the LED off during the four-pulse sequence (I'll explain in more detail later on), then it is assumed the phototransistor is saturated and will not respond to light reflected off an object. This anomaly is indicated as if an object were detected because the sensor has effectively been disabled.

Three I/O ports provide the interface to the IR bumper. Port GP3, which is input only, provides on/off control. This is useful as a power-saving mechanism and to turn off this bumper's IR LEDs so they don't interfere with other possible IR applications. Ports GP4 and GP5 go active high to indicate when sensor AN0 and sensor AN1, respectively, detect an object or sense an error caused by bright light. Their active-high output time in response to an object reflection is proportional and relative to the dis-



Figure 2—The reflected IR pulses add to the existing ambient light level. If the ambient light level is measured just before the pulse, it can be subtracted from the sum leaving only the effect of the IR light.

tance to the object. The main controller of the robot could use this information to determine if there's enough time to slow down first or if it's necessary for a quick stop.

There are large current spikes when the IR LEDs flash, so you must pay attention to the circuit layout and provide sufficient power supply bypassing. An RC filter is used to isolate the analog components-phototransistors and PIC-from the LED drive. If possible, the LED power and analog power should have separate connections to the power supply or at least should use separate wires to connect to a common source on the circuit board.

PERFORMANCE

Table 1 shows experimental results for different values of IR LED currentlimiting resistors (R_{LED}) with various phototransistor emitter resistor values and two different sensitivity values. An IRLD014 logic-level MOSFET drives the LED.

Notice that as the LED current-limiting resistor decreases the IR bumper circuit current increases, but so does the range. (The current shown is the average current of the entire IR bumper circuit. The instantaneous LED current is much higher.) To conserve power, the IR bumper needs to be on for only the direction of robot motion and only when in motion. For example, if moving forward, the rear sensors can be off, and vice versa.

When tested around the house, some typical household items worked better than others and some not at all. Oak wood furniture and drywall were detectable in excess of a foot. Clear glass in a patio door was visible at 4", but smoked glass was not detected. The door of my black refrigerator was detected at 6", but the angled vent slats at the bottom were not detected. The lesson here: don't depend exclusively on IR bumpers. A complimentary bumper system might use ultrasonics, but that's a subject for another article.



Figure 3—The simple, infrared bumper isn't amplified. Looking at the complete system, you see that it has two sensors and is pulsed.

OPERATION & SOFTWARE

The OPTO routine controls the IR pulse emission and detection and provides compensation for varying light levels (you may download the code from the *Circuit Cellar* web site). This is accomplished by measuring the voltage of the phototransistor three times, as is demonstrated in Figure 4. The first measurement is at point *A*. This is followed by a delay equal to the time the IR LED will be on, but with the LED off. Then the voltage is measured at point *B*. Finally, the LED is turned on and the phototransistor is measured again at point *C*.

The time period from point A to point C is relatively short compared to varying light levels. It is assumed that the rate of change from A to B will hold for the change from points B to C.

Because the effect of the reflected pulse is added to the ambient light level, its shape changes in concert with ambient light. Thus, the reflected light pulse when measured at point C has been modified by the effect of the ambient light during the time from point *B* to C, which is essentially the same as from A to B. So, the effect of any variations in light levels during the time from point A to B is added to the start of the time from when the IR LED is on at point B generating a new baseline point *B*'. The phototransistor output with the LED on measured at point C is then com-

pared to *B'*, providing compensation for the effects of changing light levels.

The amplitude of the reflected pulse on the positive slope is (C - B) + (B - A) = C - (2B - A), where 2B - A is the new baseline. On the negative slope, the amplitude is (C - B) - (A - B) = C - (2B - A), where again 2B - A is the new baseline. These corrections occur in the routine FIXBAS.

The waveform crest is a special case because the slope changes direction. If point *B* occurs at the maximum of the crest, it is possible that point *A* will be slightly less than point *B* and point *C* will occur on the slightly negative slope. For this circumstance, the equation B' =2B – A will provide the opposite answer to what is expected. To account for this, if B – A = 1, then *B'* will be set equal to *B*. This conclusion implies that it can

R _{led}	W = 1 Average current	R _e =1 kΩ	Ra R _e = 680 Ω	nge R _E = 470 Ω	R _E = 330 Ω
4.7 Ω	34 mA	14.5″	12.5″	10.5″	9″
6.8 Ω	28 mA	13.5″	11.5″	9″	7.5″
10 Ω	22 mA	12″	10″	8.5″	7″
22 Ω	13 mA	9″	7.5″	6″	5″
33 Ω	11 mA	8″	6.5″	5″	4″
4.7 Ω	34 mA	11″	9.25″	7.75″	6.25″
6.8 Ω	28 mA	10.25″	8.25″	7″	5.75″
10 Ω	22 mA	9.5″	8″	6″	5″
22 Ω	13 mA	7″	5.75″	4.5″	3.75″

Table 1—An IR bumper without amplification responds as shown. The range is a function of the phototransistor emitter resistor R_{e} and the LED drive current.



For complete rules and entry form

www.circuitcellar.com/PSoC2002

Get paid to learn about PSoC[™] MCU

- The fastest growing MCU architecture
- The highest level of integration
- The next new thing in Embedded



Take your career to the next level.



Win a share of **\$15,000 in Cash Prizes!**

Brought to you by:





The Cypress logo mark is a registered trademark of Cypress Semiconductor and Cypress MicroSystems, PSoC, and Programmable System-on-Chip are trademarks of Cypress MicroSystems Inc. The Circuit Cellar name is a registered trademark of Circuit Cellar Inc.

occur only when *A* is one less than *B*. If *A* is two less than *B*, then the LED pulse period occurs early enough or late enough on the crest that the equation B' = 2B - A is still valid.

The phototransistor voltage caused by a reflected pulse measured at point Cis cpoint in the OPTO1 and OPTO2 routines. In theory, if cpoint is greater than the fixed bpoint, it would be because enough light was reflected off a nearby object to be detected. In reality, there may be some signal induced in the phototransistor circuit when the LED pulses. This will probably occur unless the power and ground connections to the LED drive are decoupled from the analog section of the PIC. To compensate for the induced voltage, a minimum offset value (W) is added to bpoint, and cpoint is compared to it. Then, whenever the reflected voltage exceeds the amount W + bpoint, it is because of a valid reflection off a nearby object.

Variable *W* also can be used to control the sensitivity of the sensor. Low values such as W = 2 give the greatest sensing range. Higher values reduce



Photo 1—Each of the two robot eyes is made out of PVC plastic plumbing pieces. There are two IR bumpers consisting of a phototransistor and an LED. One sensor is located above the photo-flash tube and looks forward, and the other sensor is mounted on the side near the Polaroid ultrasonic transducer. The optoelectronics mount in holes drilled using a no. 10 drill bit.

the range allowing you to change the sensitivity under computer control depending on the application.

To reduce alarms caused by false detections, four consecutive detections are necessary before the detection valid bit goes active. This also aids in identifying when the phototransistor is saturated. As stated previously, the saturated condition results in a DC voltage level above the normal oper-



ating level. Because the varying intensity of an AC line-powered incandescent lamp can exceed the saturated voltage level momentarily at the peak of its cycle, you have to measure the voltage at several consecutive points, not all of which would exceed the DC level as shown in Figure 4.

The analog voltage measuring routine determines the pulse width of the IR LED. Part of that routine includes a time delay to acquire the voltage. Changing the time delay changes the pulse width. For the components and time constants specified, the pulse width should be in the range of 250 µs, ±10 to 20 µs.

The range value also can be used to implement a relative distance range finder. Smaller values of range correspond to longer distances because the range value is proportional to the intensity of the reflected light. This is also because the further the object, the lower the intensity. After the object is detected, as the robot gets closer, the range value for that object will increase proportionally. For most applications, this is optional and just knowing that an object is in the path is sufficient.

The pulse width varies with distance, so the use of a simple RC filter generates a voltage that is also proportional to distance. By varying the timing loops to adjust the pulse width you can change the range of the filtered voltage to better suit your needs. An alternate method of monitoring the pulse width is with a timing loop.

LONG-RANGE IR BUMPER

Depending on the speed of the robot, a sensing range of 8" to 12" may not provide enough warning to stop in time to prevent a collision. For those applications in which additional sensing range is needed, you have to add an amplifier stage to the phototransistor's output. Then, depending on conditions and component selection, the sensing range can easily exceed 15".

Also, because you're using an amplifier, the intensity of the emitted IR light may be reduced to save power. The examples I'll discuss draw about 12 mA even with the extended range. Running the IR LED at higher currents will increase the range even more.

Achieve a Higher Level of Development in Robotics



The BrainStem architecture includes hardware and software to achieve higher level development in robotics.

With software drivers, libraries and applications for Java, C, C++, and most platforms, much of the code can be reused between platforms.

The BrainStem modules can be communicated with via Host computers or they can act autonomously using small TEA programs. TEA stands for Tiny Embedded Application which is a strict subset of ANSI C trimmed down for basic control loops and logic control.

Based on a 40 Mhz processor, the BrainStem has space for 10 TEA programs and can run 4 concurrently while still handling commands sent from the Host computer.

BrainStem[™] GP 1.0 Module

Features:

- 40 Mhz RISC Processor
- 5 10-bit A/D ports
- 5 Digital I/O pins
- 1 Sharp GP2D02 IR driver
- 4 High res servo driver pins
 PalmOS
- 1 1-MBit IIC Bus
- 1 Serial port
- Stackable modules

- Compatibility:
- Windows
- WinCE
- MacOS
- Linux
- Java

The Deluxe BrainStem[™] PPRK



- Use it with your Palm or WinCE device attatched
- Dowload programs from your Palm or WinCE device, remove the Palm/WinCE and let it go!
- Download programs directly from your Windows or Mac PC

For more information on the BrainStem and our complete product line, please visit our website at www.acroname.com and be sure to check out www.robo02.com



4894 Sterling Drive Boulder, CO 80301 Phone: 720-564-0373 Fax: 720-564-0376 www.acroname.com sales@acroname.com

TRANSISTOR AMPLIFIER

A simple, one-transistor amplifier can be constructed using a common NPN transistor, a couple of resistors, and a couple of capacitors (see Figure 5). The amplified and inverted output is taken at the transistor's collector. The transistor emitter resistor sets the DC operating point and gain, and the capacitor (C_E) controls the gain for AC signals. Table 2 shows different values for R_E and C_E and the resulting sensing range per experimental results. Your results may vary depending on, among other things, the IR LED drive current. Resistor R_E should not be less than 470 Ω .

Note that the collector resistor of the phototransistor is down to 220 Ω . Lower values will work but with decreased gain and higher current flow when the phototransistor is saturated. Higher values will give increased gain but also make the phototransistor more susceptible to saturation from ambient light. Thorough experimentation shows that values less than 700 Ω but greater than 200 Ω give the best balance of performance.



Figure 4—The ambient light level is subtracted prior to the IR pulse so its effect can be subtracted.

The phototransistor is AC coupled to the amplifier through a 0.47-µF capacitor. Together with the input impedance ($R_B \parallel R_A$) the coupling capacitor forms a high-pass filter. Lower frequency components such as from an incandescent lamp are attenuated while the reflected pulse is passed through. If there are no reflected pulses, the phototransistor outputs a constant DC voltage. This is blocked by the input capacitor resulting in the amplifier also outputting a constant DC level.

Similarly, when the light level is so great that the phototransistor saturates, its collector voltage is held to a constant DC voltage of a couple tenths of a volt and it, too, is blocked from the amplifier by the input capacitor. The output of the amplifier for the saturated condition is also a constant DC level—the same as the constant light level. The result is that the saturated light condition would look as if there were no object nearby to reflect IR pulses even if the robot were headed straight for a bright light.

Now, notice that resistor R_c (2.2 M Ω) is in parallel with the input capacitor. When in typical room lighting, under which the phototransistor isn't saturated, its collector voltage sits at nearly 5 V. This effectively puts R_c in parallel with R₄ causing the amplifier output to dip slightly (remember, the amplifier is an inverter). As the phototransistor begins to conduct because of bright light, the collector voltage decreases down toward zero, so R_s is now in parallel with R_B. This parallelism causes the amplifier output to rise slightly to a maximum, which may be about 0.1 or 0.2 V more than the no pulse condition.



The software checks for the saturated condition by measuring at points A, B, and C (see Figure 4). If they are at or near the maximum and all at about the same level for four consecutive measurements, then you assume it's caused by a saturated light condition. Note that you can't simply compare to see if these points are above the maximum level because under normal conditions (with a line-powered light source) there will be a sinusoid upon which the reflected pulses would be superimposed. This explains why it is instead necessary that all three points be at about the same level for the four consecutive measurements.

Pay attention to the circuit's wiring. Keep the analog parts of the circuit (amplifier) separate from the digital parts (IR LED drive). Any signal coupled into the amplifier will be amplified. This includes switching components for the LED that cause ripple on the 5-V supply or variations in ground. Connect the analog components, including the PIC, to 5 V through an RC filter comprised of a 47- Ω resistor and at least a 100-µF capacitor. Use separate power and ground connections for the LED drive circuit to help isolate the LED switching noise from the amplifier. However, if the sensor detects an object even when there is none, there still may be enough coupled noise after amplification to appear as if it were real signal. If attending to the wiring doesn't help, increase the reflected light sensitivity in the software until it does.

OP-AMP AMPLIFIER

A single supply op-amp stage also can provide signal gain (see Figure 6). Unlike the transistor amplifier version,

the phototransistor here is connected to a common collector and the noninverted signal is taken at the emitter resistor R2. A 0.22-µF capacitor parallel to R2 smoothes out the higher frequency noise components. The signal is capacitively coupled to the noninverting input of the op-amp through a 0.22-µF capacitor.

The microcontroller's ADC has a range of 0 to 5 V. To make best use of that range

for varying light sources such as incandescent lamps, the DC output of the opamp is biased to approximately 2.5 V This is accomplished with voltage divider resistors R4 and R6 and depends on the gain of the op-amp, which is determined by resistors R5 and R3. Divide 2.5 V by the gain to get the voltage divider's value. Note also that the input coupling capacitor (C3) in concert with the equivalent resistance of the voltage divider (the equivalent resistance is R4 parallel to R6, which approximates to R4 because R6 is much greater than R4) forms a high-pass filter. Choose R4 and C3 for a cut-off frequency of about 250 Hz. This will attenuate AC line sources such as from an incandescent lamp while passing the approximately 500-Hz reflected pulses.

The capacitor parallel to the feedback resistor takes the rough edges off the amplified pulse. The RC time constant should be in the 15- to 18-µs range—the exact number is not critical. With a gain of 40, the typical sensing range is 15". For a gain of 70 the range increases to 20".

As with the transistor amplifier version, some tricks are needed to detect phototransistor saturation. This is done two ways. The first way is with 1-M Ω resistor R7 across the input capacitor (C3). Then, as the phototransistor output becomes more positive with increasing conduction, the DC output of the op-amp will rise to a maximum of about 0.3 V above typical lighting conditions. At saturation, the phototransistor simplifies to a short circuit. Any noise present on the 5-V line appears at the input to the op-amp and is amplified. It turns out that when the IR LED is pulsed on, the 5-V line



Figure 6—The phototransistor amplifier uses a single supply op-amp.



Figure 5—The phototransistor amplifier uses a single transistor and a couple of capacitors and resistors.

dips slightly for the duration of the pulse. The amplified result is shown in Figure 7. Points *A* and *B* should measure within 1 or 2 bits of each other, but point *C* will be less than both of the others. This, combined with the higher DC level, provides a method for detecting saturation.

Even though some noise is beneficial for detecting saturation, it is still advisable to decouple the analog sections (phototransistor, op-amp, PIC) from the LED drive through an RC filter (R1/C4). Also, be careful routing the 5 V and ground for the LED and try to keep them separate from the analog power.

TRIPLE SENSOR

A third analog input channel could be used for an IR bumper that monitors three sensors (you may download the schematic from the *Circuit Cellar* web site). It operates essentially the same as the two-sensor circuit except in the way it signals that an object has been detected. Using the third port bit GP2/AN2 as an analog input leaves only two bits to communicate with the main controller. One of those bits, GP3, is input only, which leaves just GP5.

The solution is to make GP5 bidirectional and to send the sensor-detected

bit serially. As an input, GP5 monitors the on/off control for the PIC. It works as follows. The main controller's I/O bit connected to GP5 is a combination input/open collector output bit pulled high with a 10-k Ω resistor. At powerup, and as part of the sensor-checking loop, GP5 is configured as an input and checks the state of the I/O bit. If the main controller pulls it low by turning on its open

Resistor R _e	Capacitor C _e	Range
470 Ω	No capacitor	14.5″
680 Ω	No capacitor	12.5″
1000 Ω	No capacitor	12″
2200 Ω	No capacitor	8.5″
470 Ω	2.2 μF	27″
680 Ω	2.2 µF	26″
1000 Ω	2.2 μF	23″
2200 Ω	2.2 μF	20.5″
3300 Ω	2.2 μF	19″
4700 Ω	2.2 μF	16″
	·	

Table 2—When using the transistor amplifier and the given values of R_{ϵ} and C_{ϵ} , the IR bumper responds with the given values. Note that the range can exceed 2'.

collector output, the PIC is commanded off. The software loops while waiting for the I/O bit to return high. When the main controller's open collector is off, the I/O bit is pulled high by the 10-k Ω resistor, which is sensed by GP5 as the command to turn on.

The bumper's PIC then checks each sensor for a valid detection. At the end of the sensor-checking loop, if an object is detected, GP5 will be set low by the PIC. After a time period of, say, 100 µs, GP5 may or may not return high depending on which sensor has detected an object. During the next 100-us window, while sensor one detects an object, GP5 will stay low. Sensor two corresponds to the next 100-us time slot, and sensor three corresponds to the last. If any of the sensors has a valid detection, GP5 will be held low during its corresponding time slot after which the sensor-checking sequence is repeated (you may download a diagram of this from the Circuit Cellar web site).

Another option, depending on your requirements, is for GP5 to simply indicate when any of the three sensors detect an object. You may have three sensors in the forward direction, in which case the robot should stop if any of them detect an object, but which one of them detects the object isn't important. It's possible that all three will detect the object. For this application, GP5 could simply pulse low or remain low as long as an object is detected.

To implement bidirectional control using GP5, GP5 is switched between an input port and an active-low-only output port. It never drives the output high because that might run up against the main controller driving the I/O bit low. When switching from an input to an output, you must set the output low before converting to Output mode. Then, when GP5 transforms from input to output, it will already be set low and will output an active-low state. When it's done outputting the low state, GP5 switches back to an input to effect a high state. Thus, to send an active low pulse, GP5 starts as input and the I/O line is pulled high. Next, GP5 is set low and then switches to an output port, which effects an active low. At the end of the low pulse, GP5 switches back to input and the line is again pulled high by the resistor.

QUAD SENSOR

The 8-pin PIC has four analog inputs, so it's possible to monitor four sensors (download the circuit diagram from the Circuit Cellar web site). However, this is tricky because two output bits, one to drive the LED and one to signal an object detection, are needed. Of the two non-analog I/O bits, only GP5 has output capability. The solution is to have two analog inputs perform double duty and function as digital outputs and drive the LEDs. As with the triple sensor described earlier, GP5 performs bidirectional communication with the main controller and adds a fourth time slot for the fourth sensor. Or, if it isn't necessary to know which particular sensor detected the object, GP5 simply could be taken low to indicate that at least one sensor detects an object.

The LEDs and phototransistors are grouped as two groups of two, AN0 and AN1 as a group and AN2 and AN3 as a group. Port AN1/GP1 drives the LEDs for AN2 and AN3. Likewise, AN2/GP2 drives the LEDs for AN0 and AN1.

The LEDs are switched by logic-level MOSFETs driven by the port bits when in Output mode. The MOSFET begins to conduct current with a gate voltage around 1.5 V and is fully on at about 2 V. But, the output from an amplifier is around 2.1 V when in Analog Input mode, which would cause the MOS-FET to turn on if it weren't for the divide-by-two resistor divider connected to the gate of the MOSFET.

When the port bit is set high at 5 V in Output mode, even with the divider, the gate voltage is sufficient to fully turn on the MOSFET. Large amplitudevarying light sources, such as from an AC line-powered bulb, will also cause the MOSFET to conduct, but only on the most positive part of the cycle and only briefly. A 2.7-k Ω resistor between the amplifier and the port bit limits the current into the amplifier when the bit is in Output mode. When the PIC is commanded off and between activehigh pulses to fire the LED, the port bit should be held low. This prevents the MOSFET from switching, which, although harmless, does waste power.

OPTOELECTRONICS

One parameter of interest when choosing the LED is how much the light diffuses with distance from the source. It should be concentrated in a small spot size on the target. Likewise, the phototransistor should limit its field of view to that of the area of the emitted light. The optoelectronics specified have beam angles of $\pm 12^{\circ}$.

Sensing range can be extended with the use of optics external to the optoelectronics. As a practical matter, you can't use external optics with optoelectronic devices that already have lenses (e.g., dome-shaped LED), because you'll never get the alignment right. External optics should be used only with devices that have flat lenses. And, because these are typically clear lenses, I recommend some sort of additional IR filter.

The LED and phototransistor should be mounted near each other although the exact placement isn't critical. A typical mounting arrangement might have them mounted parallel on 0.5" mounting centers, as shown in Photo 1. They can be mounted right in the skin or frame of the robot or in separate fixtures. (Use a no. 10 drill bit for T-1 3/4 packages.)



Figure 7—Note how to detect phototransistor saturation when using the op-amp amplifier.

What is important is that they be optically isolated from each other. There is enough stray scatter in the plastic packages for the phototransistor to detect IR light from the nearby LED. This includes the back of the package where the leads protrude. Also, some plastics or heat shrink tubing that appear to be opaque may not be so to infrared light, so be careful what material you mount the optos in. To aid in isolating stray emitted IR radiation, you can use a TO-18 metal package IR LED like the F5D1QT in place of the QED123QT. The F5D1QT does cost five times as much, however, the small quantity that you need and ease of mounting for isolation concerns should be considered.

The phototransistor could be mounted recessed in a tube that allows the reception of the emitted IR pulse yet blocks most of the indirect ambient light. Then, unless a light source is directly lined up with the sensor, its effect will be limited to reflections off nearby surfaces, allowing use of a higher emitter resistor value for increased sensitivity.

A variety of optoelectronics were tested. The best results were achieved with the inexpensive QED123QT IR LED and QSD124QT phototransistor. In addition, the phototransistor has a built-in IR filter, which reduces the effect of most visible light. This is especially important in areas of high fluorescent lighting because this light has considerable noise associated with it in the visible spectrum, which is cut out by the IR filter.

I also tested L14F1QT photo Darlingtons. Their higher inherent gain results in longer range but at the expense of easier saturation. Photo Darlingtons are not recommended unless your application can tolerate their characteristics. They have some benefits in the simple, basic IR bumper in which no additional amplifier is used, but use them with caution. Choose the emitter resistor value that works best for you.

APPLICATIONS

You have several options to choose from to meet your robot's noncontact bumper needs. For most applications, the simple IR bumper that doesn't use additional amplifiers and has relatively few parts will work fine. And, the sensors can be mounted right in the skin of your robot, providing flexibility in the shape and construction of your robot. There are no modules or clumsy little metal boxes to mount.

If you need more than two sensors, you can easily expand this system by using a microcontroller with multiple A/D input channels. And, you are not limited to the Microchip PIC series because several microcontrollers come with a built in A/D converter.

Although this sensor is intended for robotic uses, there are many other applications that would benefit from its simplicity and immunity to external light sources. These include the ubiquitous hands-free water faucet or toilet flush control, noncontact and electrically isolated switches, sensing whether or not an object is where it should be, security measures, and numerous others.

Tom Baraniak attended University of Wisconsin, Madison. Presently, he runs the Electronics Shop for the Chemical Engineering and Materials Science Department at University of Minnesota. As a kid he built his first robot out of orange crates and coffee cans. Today his materials are a bit more sophisticated. You may reach him at baraniak@spexotics.com.

SOFTWARE

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2002/141/.

SOURCES

IR LEDs, phototransistors, MOSFET switch Digi-Key Corp. (800) 344-4539 (218) 681-6674 Fax: (218) 681-3380 www.digikey.com

PIC12C672-04 Microcontroller

Microchip Technology Inc. (480) 786-7200 Fax: (480) 899-9210 www.microchip.com



FEATURE ARTICLE

Mike Rigsby

Rovervac—A Robotic Vacuum Cleaner

What do you get when you attach an ultrasonic detector and a PDA to a handheld vacuum cleaner? It may not be lean and mean, but Mike's mobile cleaning machine project provides ongoing enjoyment as he continues to develop his household helper. have long been interested in moving machines—a Lionel train when I was six, Mr. Machine when I turned eight, and so on. I was an early adopter of a first generation Aibo, the robot dog from Sony. Still, it seems to me that robots ought to do something worthwhile. From the persistent desire to create a machine that could do something useful came the idea for the Rovervac, the robot vacuum cleaner. Photo 1 shows the Rovervac.

In its current configuration (the Rovervac evolves weekly), the robot finds its way around obstacles and vacuums the floor. An ultrasonic detector sweeps like a radar beacon to find obstacles. A BASIC program running on a Palm PDA provides the logic for operation.

A hand-held vacuum, the Deluxe Wet-Dry manufactured by DirtTamer, with high efficiency particulate air (HEPA) filtration does the cleaning. HEPA filtration prevents particles larger than one micron from being blown back into the room. The Palm PDA communicates with a Pontech servo controller board, taking input from the ultrasonic detector and providing output to the relay control board. The ultrasonic sensor module works by sending out a 40-kHz sound and timing the returned wave. Photo 2 displays the ultrasonic sensor. This process occurs 20 times per second. The output of the sensor is from 0 to 10 V, representing 6" to 6'. Zero voltage on the output also represents no return of the sound wave (distance greater than about 10').

The Pontech controller accepts analog inputs less than 5 V, therefore, it's necessary to reduce the ultrasonic output voltage (through a voltage divider network on the relay controller board) before it's digitized and sent to the PDA. Output of the ultrasonic transducer covers a 15° path.

The 15° path is not sufficient to detect everything in the forward path of the vacuum, so I tried alternatives. First, I placed baffles (razor blades) in front of the transducer. This deflected the sound waves and changed the area of detection, but it still left holes in the detection window. The "one sensor sees all" method also created another problem: An object in front of the vacuum may need to be detected at 12" from the sensor, and an object to the side becomes a problem when it's 8" from the transducer.

My next effort required using two ultrasonic modules angled slightly toward the center, as you can see in Figure 1. I thought this would cover all of the area in the forward path of the Rovervac. The uncovered triangle in front of the machine could be managed by software, because noth-



Figure 1—You can learn from my mistakes as well as my successes. I intended to cover the detection zone with two ultrasonic detectors properly multiplexed, but it didn't work. The modules weren't synchronized.



Photo 1—The Rovervac, a cordless robotic vacuum cleaner, is composed of electronic components mounted on a clear acrylic base.

ing could get into the triangle without first passing through one or both detection zones.

I discovered erratic behavior when I tested this setup. The problem was caused by the operation of the two modules that weren't synchronized. For example, the transducer on the left might send out a sound wave then shut down and listen for the return. The left module would correctly read a return from 6' away, but the right module might go into Listen mode as the sound wave returns. The right module thinks the wave it sent out came back instantly, so something must be inches away. To successfully use multiple transducers, different frequencies or multiplexing (meaning one transducer pinging at a time) must be used.

Another minor challenge with this module is that distant objects (greater than 6') give the same output (0 V) as close objects (less than 6"). I solved this by aiming the transducer slightly

downward (see Photo 3) so that a return from the floor always can be expected. Furthermore, I used software to eliminate readings of 0 V, because normally nothing will get within 6" of the transducer.

I finally settled on using a servomotor, which you can see in Photo 4 to rotate the transducer. First, the sensor assembly is rotated to the left, then forward, then right. At the same time, ultrasonic readings are taken in each position. This method has the flexibility of allowing a ping to be taken in any direction; it also supports the use of different actions based on obstacles in varying relative locations. For example, an object detected to the right but not detected in the center may require a minor course correction rather than a total rotation.

Something I learned from my experience with Aibo (which goes against my engineering ways) is that everything a robot does doesn't need to be useful or practical. My wife wanted wiggle eyes, a nose, and a mouth, so those features are attached to the vacuum to indicate which end is the front. Expanding on the almost useless features, I added a tail-a pipe cleaner attached to a servomotor. The tail is normally down but whips up when the Rovervac rotates. The tail can be programmed to wag up and down or do other useless tricks, but also is a beneficial programming tool. During operation, it's easy to test functions by having the tail rise when a line is reached in software or when a specific input condition has occurred.

Mechanical assembly of electrical devices always proves to be a challenge for me. I didn't want this vacuum to be an upright, top-heavy device, because that would create tipping problems, require slow acceleration, and prevent cleaning under tables and counters. Having settled on low height, shape was the next consideration. Although rectangular forms are desirable for moving a rotating brush close to a wall, rectangles can be more challenging to maneuver (sometimes you have to back up and angle out of a



Photo 3—The geometric structure of the Rovervac puts most of the weight behind the wheel axis allowing the front to be raised above the carpet. The ultrasonic sensor (upper right) is angled in such a way to guarantee a return ping.



Photo 2—The Senix ultrasonic sensor board has four terminals—two for power and two for output.

tight spot). Two wheels allow a round disk to rotate around its center so it never has to back up.

I based the diameter of the disk on the size of the cordless vacuum it had to carry. I cut a small hole in the front of the base and fastened the bendable attachment to the hole (see Photo 5). I considered pointing the business end of the vacuum down to reduce the circle diameter, but this quickly devolved into a tall, unstable machine. For the base, I cut 0.25" thick acrylic with a scroll saw. The task was easy. This was my first experience with a scroll saw, and it went well. The diagram in Figure 2 shows the base.

For me, mounting the airplane wheels to the gear motor shaft proved to be a great construction challenge. In principle, you need to pull off the foam tire, separate the wheel halves (they slide apart), drill a hole the same size as the motor shaft, drill in a set screw, insert the motor shaft, tighten the set screw, push the wheel halves together, and replace the foam tire. The motor shaft has a 5-mm diameter. Forget about English equivalents, you need a

5-mm drill bit (unless you like wobbly wheels). The 5-mm drill bit, if you live in the U.S., is best found on the Internet.

CONTROL

Controlling the Rovervac involves four elements, including the processor, software, cables, and interface electronics. The Palm m100 serves as the processor. I used HotPaw Basic for the software. A Palm serial cable and a null modem adapter were chosen. Lastly, I chose the

Tigers: powerful CPU modules from Wilke & JED Microprocessors

The BASIC, Tiny and Economy Tigers are sold worldwide, with W98/NT software and six JED, Australian-made boards.



Tigers are modules running true compiled multi-tasking BASIC in a 16/32 bit core, with typically 512K bytes of FLASH (program and data) memory and 32/128/512 K bytes of SRAM. All Tigers use the same low cost W98 or NT compile, debug and download software.

The Tiny Tiger has four, 10 bit analog, lots of digital I/O, two UARTS, SPI, I²C, 1-wire, RTC. The module is 60mm by 28mm. From \$US 89.

The **BASIC TIGER** is a larger unit, with more I/O pins, up to 4Mbyte of FLASH and 2Mbyte of SRAM. A CAN version is also available.

The latest is the 28-pin Economy or E-Tiger, which is only 40mm long, and is an economical way to apply the power of Tigers to cost-sensitive applications. From \$US 49.



TIG505 Single Board Computer

The TIG505 is one of six Australian Tiger boards, and uses the TCN4/4 Tiger processor with 512K FLASH & SRAM. With 50 I/O lines, two RS232/485 ports, SPI, RTC, LCD, 4x10 ADC, 4 (opt.) DAC, and FLASH mem. Various Xilinx FPGAs add special I/O, eg 3x 32bit quad shaft encoder, X10 or counter/timer functions.

Standard TIG505 with TCN4/4: \$US 210

JED Microprocessors Pty Ltd

173 Boronia Rd, Boronia, Victoria, 3155, Australia. (Visa card accepted). Ph. (+61) 3 9762 3588, Fax (+61) 3 9762 5499 (Freight and local taxes not included in prices) *www.jedmicro.com.au*



Photo 4—As you can see, I use a micro servomotor to rotate the ultrasonic transducer.

Pontech controller board (see Figure 3) and relay interface board (see Figure 4) as the interface electronics.

The Palm m100 comes with the required serial cable. It needs only a null modem adapter to connect to the Pontech board. HotPaw Basic software is available for a free 30-day trial basis, after which a \$20 registration fee is levied. The Pontech board has eight outputs that are configured in the software as servo outputs or digital outputs. Five 8-bit analog inputs (0 to 5 V input) are available on the Pontech board.

Digital outputs from the Pontech board control a transistor that switches a motor control relay on or off to control the DC gear motors that drive the Rovervac. Each motor is controlled by two double-pole, doublethrow relays. One relay is the on/off power switch for a motor. The other relay is forward-reverse switch for the motor. The relay control board also contains a voltage divider network to reduce the ultrasonic transducer output to a level that is acceptable to the Pontech control board.

The Pontech SV203 is a PIC16C73 microcontroller-based servomotor control board that takes serial data from a host computer and outputs pulse width modulated (PWM) signals to control up to eight servomotors (see Photo 6). The board also includes a five-channel A/D port for receiving 0- to 5-V inputs.

I used simple commands in this project. Let's go through a number of them. The BDn command selects the board; the n represents the board number. It is possible to chain up to 255 of these boards together. The next command, BD0, overrides the board ID number and allows any boards connected to the serial port (I have only one) to be enabled.

The SVn command selects the servo pins to operate on. Here, n is a number between one and eight. Mn moves the servo to an absolute position from zero to 255, which moves the servomotor through 180°. Before a servo pin can be used for digital (rather than PWM) purposes, Mn must be set to M0.

I use the PSn command to set a servo pin. PS1, for example, causes pin 1 to go high. Conversely, PCn clears a servo pin, hence PC1 causes pin 1 to go low. Another command, Adn, gets a digital value from zero to 255 from analog input pin *n*.

Now that you know the commands, I'll focus on the rest of the software. The programs for HotPaw BASIC are written in Palm's Memo Pad. To be a software listing, the first line of the memo must start with "#" and end with ".bas". For example, the program I wrote is titled #zzz3a2.bas (download it from the *Circuit Cellar* web site). A handful of details should give you a better picture of what's going on here.

Line five disables the power-saving auto-shutdown feature of the PDA. Lines 15 and 20 open up the communication port. Line 30 causes the motors to turn on for forward direction and the ultrasonic servo to turn on for the tail to go down.

With command BD0, you enable the board. SV7 M0 PS7 causes pin 7 of the Pontech board to go high. This turns on a transistor (relay board CR2), which activates relay MR1 (see Photo 7). SV8



Figure 2—The major cutout features for the 0.25" thick acrylic base are illustrated in this drawing.



Figure 3—From the top, you can see the wiring of the Pontech SV203 controller board. It is part of the interface electronics.

M0 PS8 causes pin 8 of the board to go high. This turns on a transistor (relay board CR1), which activates relay MR2. Relay MR2, when energized, allows power to go to the contacts of relay MR1. Relay MR1 is energized, so 12 V (+M) goes out on line M1B to the positive terminal of the first motor.

The line SV6 M0 PS6 SV5 M0 PC5 causes MR4 to be energized and MR3 to be not energized. This results in 12 V (+M) going out on line M2A to the negative terminal of the second motor. The first and second motors must turn in opposite directions to move forward. SV2 M250 causes the tail to go down. Line 70 moves the ultrasonic servo to the center position (pointing to the front). Line 100 retrieves the distance reported by the pinging of the ultrasonic transducer board.

The rest of the program evaluates ultrasonic readings and responds accordingly. If an object is detected, the Rovervac starts to rotate (beginning at line 430) and the tail goes up (line 465). If no objects are detected, control is returned to line 30 and the tail goes down.

SETTING UP

To mount devices to the acrylic base, drill holes slightly smaller than screws and thread in the screws (tap-



Photo 6—A null modem adapter and Palm Hotsync cable are attached to the Pontech controller board.

ping into the acrylic). I used hook and loop fasteners to attach the Palm m100 and the vacuum to the base. For the power supply, I used separate sources whenever possible to avoid noise problems. This simple approach resulted in 17 batteries onboard the device. They break down as follows: the Pontech controller board uses 6 V (four AA batter-

ies), the Palm m100 uses 3 V (two AAA batteries), and the ultrasonic detector requires more than 15 V (two 9-V batteries). An internal lead acid battery does the trick for the vacuum cleaner, which needs many amps at 6 V. Finally, the relay control board uses 6 V and the drive motors use 12 V for a total of eight C batteries.

Now that it's built, you have to return to the obligatory question: Is the Rovervac practical today? The answer is not yet. I installed the batteries to the Pontech control board because



Photo 5—The flexible attachment that comes with the vacuum is screwed into the acrylic base. Spacers can be used with the vacuum brush to obtain the brush height that you desire.

there is no on/off switch. To use the device, I turn on the Palm m100 and start the software. Next, I plug in the ultrasonic batteries. At this point, I turn on the vacuum and then insert the relay/motor batteries. The vacuum will run for about 12 min., at which time all of the other batteries need to be pulled unless you like purchasing 16 alkaline batteries at a time.

Fortunately, the future looks better. I am currently working on "find home" and recharge additions to the







shipping included!

expresspcb.com





Figure 4—The relay driver and analog interface board provide power control to the wheel drive motors and voltage reduction for the ultrasonic input.

Rovervac. The vacuum cleaner voltage can be easily monitored through a spare analog input on the Pontech control board. Another relay (FYI: there are outputs to spare) can shut down the vacuum.

When the vacuum is off, the Rovervac needs to find its way home. It can already get around obstacles, so wandering about isn't a problem. Adding infrared receivers to the ultrasonic rotating beacon would allow the Rovervac to stop and search for a homing beam. The Rovervac could then rotate to the beam position and head to the charging station.

Exactly how will the Rovervac connect to a charging supply and feed all of the multiple voltage sources? The answers may surprise you. Perhaps I'll write a follow-up to conclude this project after I tackle the additions.



Photo 7—The relay control unit is a soldered, handwired prototype board.

Mike Rigsby, PE, is manager of the design team for the Department of Transportation in Lee County, Florida. Mike is an electrical engineering graduate of Vanderbilt University. He wrote Verbal Control with Microcomputers as well as published numerous magazine articles. His interest in robotics landed him a spot on a French Documentary, "Robots, Man's Best Friends?" and a recent interview with the Miami Herald. You may reach Mike through his web site at www.misterengineer.com.

SOFTWARE

To download the code, go to ftp.circuitcellar.com/pub/Circuit_ Cellar/2002/141/.

SOURCES

HotPaw BASIC www.hotpaw.com

Palm m100 PDA Palm, Inc. (408) 878-9000 Fax: (408) 878-2750 www.palm.com

SV203 Controller board Pontech (877) 985-9286 (413) 235-1651 www.pontech.com



see what's new at www.saelig.com!



DrDAQ plugs into a PC for useful datalogging at school, college, industry. Built-in sensors for light, sound, temp. or add pH sensor and run one of the many suggested science experiments! - only \$99!

PC Instruments

ADC-10 8-bit \$79 through ADC-216 16-bit \$799—display scope, spectrum and meter simultaneously. Connect to PC



parallel port and start gathering/displaying data immediately!

- EnviroMon temperature logging/alarm system standalone or with PC.
 - TH-03 thermistorto-PC converter
 - TC-08 8x thermocouples

1-888-7SAELIG

• 1 ⁺ ′k

١k



Remote control & data RS232 to RS422 & Make PCs acquisition without power! **RS485 converters** talk I²C easily! BITlink Plug directly into PC - self powered! ISA PCI p-port 2-wire • Drive any RS422 control network or RS485 devices. power & information on 2-wires! Industry-standard Send control and data \$299 100s of feet! card for PC's Over 50 different DIN-modules for: analog i/p & o/p, thermocouple i/p, K422/K485, 25pin > 9pin ... \$69 Master, Slave or Bus monitor digital i/p, relays, on 2000m network! Control or program I²C devices K422-99/K485-99. 9pin > 9pin \$69 • Low volt ICA93LV for 3V ic's ... \$399 US Starter Packs Isolate RS232/422/485 signals • PARALLEL PORT VERSION "How to PC" Boards for PCI/ISA/PCMCIA/PC104/VME/compPCI Kxxx-ISOL 25pin www.saelig.com NEW - PCI VERSION! self-powered \$139 Drivers for WIn95/98/NT,VxWORKS, pSOS, Lynx, Linux, OLE, HPVee, LabView, etc. 2-Year **CANboards PCMCIA Dataloggers** Datalogger TDS2020D SMD PCB adapters **lowpower PCcard logging** Self-contained in for prototyping Store analog/ 2" x 3" plastic box, digital data on 2-year battery-<u>USB ic's</u> PC FlashATA cards powered analog and > 100 readvmade RS232 <> USB easily!! digital dataloggers software function for storing events, modules-finish quickly. voltages, currents or • 8ch 10bit A/D, 33 I/Os, I²C, 2 x pressures for days to weeks. 1988. Saelig RS232, interrupts, sleepmode, Download detailed time and data pre-emptive multitasking, easy to via RS232 port and review your attach LCD or keypad. results with graphic software or CANbus adapter—recompile or log PC spreadsheet. data over huge network! www.abidata.be for details. We are noted for our great after sales support.

Saelig Co. Inc. 716-425-3753 • Fax: -3835 www.saelig.com • saelig@aol.com



FEATURE ARTICLE

Joseph Jones & Ben Wirz

RoCK Specifications

Part 1: Overview of a Winning Robot Kit

Designing robots can be a daunting task that requires electrical engineering, mechanical engineering, and programming skills. To make the process easier, Ben and Joseph used an AVR AT90S8535 to help convert a radiocontrolled car into an autonomous robot. ave you ever thought of converting an RC car into a robot? Dusting off an old

toy from the back of your closet and adding a microprocessor and some sensors is an appealing and inexpensive route to building a robot. But the task may be daunting if your experience is limited. Robots are notorious for requiring expertise in diverse disciplines. Electrical engineering, mechanical engineering, and programming are just some of the relevant fields.

Still, maybe you'd take the plunge and actually build that robot if it were just a bit easier to get started, if you could discover the right electrical and software modules to simplify the job. It was with the thought that many other people find themselves in this exact situation that we set out to create our entry for the *Circuit Cellar* Design Logic 2001 contest sponsored by Atmel. We devised an electronics package called the Robot Conversion Kit (RoCK) that transforms an RC car into an autonomous robot.

In this and the next few issues of *Circuit Cellar*, we will describe how to convert a toy into an autonomous robot. Because our aim is to pass on a deeper appreciation of the RoCK rather

than a simple recitation of its features, schematics, and code, we will explain the goals we pursued, principles that guided us, and specifications we developed. In subsequent articles, we will describe how we realized our aims by probing deeper into the hardware and software of the RoCK. We will illustrate how modern methods of robot programming can extract great power from modest hardware.

DESIGN GOALS

Our desire as authors and robotics professionals is to promote wider understanding of robotics. As practitioners of the field, we know that understanding robotics requires building robots, writing programs, and observing robotic behavior. Unfortunately, few newcomers take the initiative to study robotics if the available learning tools are overly expensive, needlessly complicated, or insufficiently powerful to teach sophisticated lessons. To address these concerns, we required that our scheme be inexpensive, easy to use, and sophisticated.

LET'S RoCK

We will proceed with a more systematic exposition of our design, but to better understand the development of our story, let's first jump ahead and see what a RoCK-powered robot can do (see Photo 1). Suppose that you have already chosen a mobile base, removed the factory-installed receiver circuit, and attached a RoCK module. At power-up, your RoCK-enabled robot, call it RoCKbot, plays a wakeup tune. At the same time, RoCKbot displays a pattern on its LEDs that indicates which of the 16 possible tasks the robot will execute. You select the Chicken task.



Figure 1—A robot is a device that connects sensing to action in an intelligent way. Robot builders must be able to program the robot. Builders benefit greatly from access to the robot's internal state.

Before putting RoCKbot on the floor, command maximum speed by twisting the dial all the way to the left. When it's set free, RoCKbot races across the floor toward the wall. At the last second the robot beeps and swerves away. RoCKbot then slams squarely into a narrow chair leg that the infrared (IR) obstacle sensors failed to detect. After about a second of futilely pushing against the chair, the robot backs up, spins in place, and then moves forward in a new direction.

Although it tries to avoid you, grab RoCKbot as it speeds by. Press the user button and twist the dial until the LEDs display the pattern for the Roach behavior. In a darkened room, place RoCKbot in the middle of the floor. RoCKbot waits patiently. When the room light is switched on, RoCKbot races about the floor, homing on the darkest spot. Safely concealed in the shadows, RoCKbot comes to a halt.

By connecting a serial cable between RoCKbot and your computer, you can display readings from the sensors of the RoCK and issue commands to RoCKbot's motors. You create a new task by selecting new values for the internal parameters and stringing together the built-in behaviors of the RoCK. You can then select the newly programmed task, now stored in EEP-ROM, using the dial.

Comfortable with your understanding of the behavior-based programming scheme RoCK employs, you can dispense with the built-in code. An external programmer and compiler let you replace the code in the flash memory of the RoCK with custom code of your own. A built-in connector makes it possible to add new sensors and actuators and control them with the new code. The abilities of the RoCK grow as your understanding deepens.

DESIGN PRINCIPLES

How do we achieve our goals of low cost, ease of use, and sophistication of control in a practical device? We start with a careful definition of a robot and the elements essential to all autonomous robots.

A computer program inhabits a virtual world where the rules are certain and unchanging. This is not so for a



Figure 2—This bottom view shows the two major drive configurations. A typical drive/steer base (left) has two driven wheels pointing in a fixed direction and two passive wheels that can be steered. Differential drive bases (right) have two wheels whose velocities can be independently controlled. Differential drive robots often use a caster for balance.

robot. A robot operates in the physical world where noise is present in every measurement, where intended actions have uncertain results, and where the robot must compute a safe response even in situations you didn't anticipate. These cruel facts frame the fundamental realities of robot design.

Thus, to function in the real world, a robot, like animals and people, must perceive key aspects of its environment. The robot then must move or otherwise act. The robot must connect its perceptions to its actions intelligently. Hence, a useful definition of a robot is a device that operates in the real world and connects sensing to actuation in an intelligent way (see Figure 1).

This basic definition suggests some components that all robots share. First, a robot must incorporate one or more sensors through which it can perceive the world outside of itself. Second, a robot must have a means of locomotion or actuation so that it can interact with the world. And third, a robot must have a computing element to enable complex (and hopefully intelligent) connections between its sensors and actuators. The more sensors and actuators a robot has, the richer its potential interactions with the world.

A useful robotic system must address two practical matters. One, the system has to provide some method to program the robot. And two, you need a way to monitor the robot's internal state. Viewing the state of the robot, including sensor readings, can help to explain what's going wrong when the robot behaves in strange, unexpected ways.

Implementing a low-cost robot requires making many choices. Deciding which programming scheme to use is a critical choice. In recent years, behavior-based programming has become dominant in mobile robotics. This dominance has come about because behavior-based programs have demonstrated an ability to produce robust responses even when the robot encounters unanticipated situations. Further, these programs tend to be miserly in their consumption of computational resources. Using a behaviorbased scheme can help the RoCK achieve both sophisticated control and low cost. Because this scheme required little processing power, the RoCK can function well with only a modest 8-bit microcontroller, thus minimizing the cost of the computation system.

LED	Task	Description
0	Theremin	Differential light levels striking the photocells determine beeper frequency
1	Dance	The robot moves in a programmed pattern while playing a tune
2	Wimp	The robot beeps and retreats from anything that comes near
3	Schizoid	The robot drives around erratically but tries to avoid collisions
4	Pounce	The robot sits quietly until an object comes near, and then crashes into it
5	Moth	The robot homes in on the brightest light
6	Mouse	The robot follows the nearest wall
7	Chicken	The robot moves in a straight line but attempts to avoid collisions
8	Roach	In a lighted room, the robot finds a dark area and stops
9	Joystick	You control the robot's motors via the serial interface
10	User 1	User-programmable task number one
11	User 2	User-programmable task number two
12	User 3	User-programmable task number three
13	User 4	User-programmable task number four
14	DiffSel	Helps you correctly connect a differential drive base
15	SteerSel	Helps you correctly connect a drive/steer base

 Table 1—These 16 named tasks can be selected using the on-board user interface of the RoCK. The left-most column indicates the number displayed in binary by the LEDs during task selection.

Another key element to keeping the cost of our robotic system low is to incorporate the ubiquitous RC car. RC cars, manufactured by the millions, are a great bargain. They provide a mobility base that is rugged, inexpensive, and available in a wide variety of types.

To keep costs low and make a RoCKbot easy to build, we sought ways to simplify the interface between the RoCK and its mobile base. In our design, a total of only six wires are necessary to connect the RoCK to its base. Two wires link the RoCK to the motor battery of the RC base and two wires go to each of the base's two motors. All of the connec-

tions between the base and RoCK are implemented using screw terminals, eliminating the need for soldering.

In addition, wherever possible we reduced the complexity of the electronics package. We chose the Atmel AVR AT90S8535 microcontroller to support this effort. Analog-to-digital converters (ADCs), a comparator, and three timer/ counters are built into the 8535. These systems allowed us to use on-chip hardware and programming to implement features that would otherwise require costly external components.



Figure 4—The primitive behaviors run constantly in the software. Each uses the present sensor values and a set of parameters to compute an output for the motors. The arbiter decides which behavior is permitted to control the motors using a priority list supplied by one of the tasks via the task selector. The task selector also decides how the beeper is controlled. The user tasks (programmed by you) operate in a way identical to the other tasks except that their specifications are stored in EEPROM rather than flash memory. The scheduler runs one behavior after another and keeps the sensor values updated.



Figure 3—This block diagram illustrates the major subsystems of the RoCK. The sensors include light sensors, infrared obstacle sensors, voltage sensors for the logic and motor batteries, and a motor current-based collision sensor.

A connection to a host computer is required to program the robot and monitor its state. In implementing this connection, we chose to move the bulk of the interface code to the host computer. After simplifying as much as possible, we arrived at a host-initiated serial protocol plus a low-overhead scheme for writing data to EEPROM.

SPECIFICATIONS

The RoCK has 12 preprogrammed tasks. You can program and store up to four additional tasks of your own. The

piezoelectric beeper can play user-programmed tunes of over 100 notes. Robot trajectories composed of more than 100 path steps can be stored and executed. The RoCK has sensors to detect light, obstacles, collisions, and battery voltage. Pressing one button and adjusting a single dial lets you easily configure and control the RoCK. Connecting this device to a host computer enables you to monitor and program it. Despite its high level of functionality, the RoCK remains an inexpensive device.

Our goal was to ensure that the RoCK could work with as many different types of RC bases as possible. However, some bases make better robots than others (see Figure 2). One common type of RC base uses a drive/steer configuration. Such a base has one motor to move the robot forward or back and a separate motor to point two passive wheels in the desired direction of motion. The difficulty such a base creates becomes apparent anytime the robot wanders near confining obstacles. Lacking the ability to spin in place, a base using the drive/steer configuration can easily become trapped by even a sparse arrangement of obstacles (see Photo 2).

A second type of RC car incorporates differential drive. In this configuration, the differential

velocity of wheels on either side of the robot controls the motion. Tracked RC vehicles use the differential drive configuration almost exclusively. The capacity to spin in place gives the differential drive robot the ability to extricate itself from almost any situation. Significantly, a differential drive robot can use the same simple algorithm to escape from a trap that it used to enter the trap. By contrast, the algorithms that allow a drive/steer robot to escape tend to be arbitrary and complex. The differential drive configuration thus makes a more effective robot.

RC cars that have two or three functions have only a single motor. When this motor spins in one direction, the base moves forward. When the motor spins in the other direction, the base turns and backs up at the same time. The RoCK was not designed to use RC bases of this type. Photos 3a and b show the RoCK with the enclosure top removed and on, respectively.

The on-board user interface enables the RoCK to operate without being connected to a host computer. A DPDT switch controls both the 9-V logic supply incorporated into the RoCK module and the motor battery supply attached to the mobility base.

The illumination pattern on the four LEDs indicates the status of the robot. During task execution, LEDs generally display the state of the left and right obstacle detectors and the status of



Photo 1—*A* robot built on a differential drive tracked base approaches a chair leg.

the collision detector. During task selection, the LED pattern indicates which built-in tasks you've chosen.

The piezoelectric beeper can produce tones of arbitrary frequency, enabling the RoCK to play a built-in or userprogrammed tune. Additionally, the state of certain sensors can be mapped into the frequency played by the beeper.

The user potentiometer enables you to experiment with the parameters that control the robot's behaviors. For example, in a user task, the gain parameter in the control loop that enables the robot to follow a bright light can be mapped to the user potentiometer. You can then adjust the robot's light-following behavior from sluggish to neurotic. The user button works in conjunction with the user potentiometer to select the built-in tasks you wish to run.

The circuit board is mounted in a small, plastic enclosure. The enclosure includes a compartment for the 9-V battery that provides power to the logic. Mounting all of the sensors directly on

the board enhances reliability. The enclosure is open on one end so that the light sensors and IR obstacle detectors have a clear view. Four LEDs are on the top surface of the enclosure. Also on this surface are the power switch, user select button, and the user potentiometer.

The RoCK module includes an RJ12 connector for the serial interface. Multi-pin headers give access to unused microcontroller ports and allow connection of a programmer for the on-chip flash memory. The RoCK needs enough sensory inputs to enable a variety of interesting behaviors. We chose the set illustrated in Figure 3.

A dual-channel IR obstacle detection system is included. This system is composed of two series-connected emitters and two independently wired IR receivers. The receivers are sensitive to a 38-kHz modulation frequency. The emitter/detector pairs point diagonally outward from the RoCK to cover the area in front of the robot. Each receiver detects IR radiation reflected from nearby objects in the direction the detector is pointed, informing the robot of imminent collisions.

Dual diagonally mounted photocells measure the light level in front of the RoCK. The difference between the light measured by the left and right photocells enables the RoCK to home in on a bright light source or speed to a dark corner.

As the robot operates, the logic and motor batteries discharge, decreasing the voltage of each battery. The RoCK monitors the voltage of these two batteries allowing the robot to take action if the voltage falls too low.

The AT90S8535 processor contains 8 KB of flash memory program storage, 512 KB of SRAM, and 512 KB of software-programmable EEPROM. The chip uses the Harvard architecture with separate memory spaces for data and program storage.



Photo 2—The nonzero turning radius of this drive/steer robot can cause trouble. If the robot comes within a turning radius of an obstacle before the sensors detect the obstacle, a collision or a complex series of back-and-forth escape motions may be inevitable. The circuit board is mounted directly on the RC base.





32-bit native execution MCU J2ME CLDC w/options 512 KB to 2 MB Flash 512 KB SRAM 1x2 inch DIP 40 3.3 or 5-14 VDC www.jstamp.com

JStamp![™]

Experience all the benefits of Java technology with FAST 32-bit native execution and deterministic real-time capability. JStamp's JVM and RTOS are in silicon. Native execution means that Java is now your "machine language"!

Find out why native execution Java is the biggest news in embedded systems since C, and how it can slash your product development time and lifecycle cost by 50% or more. Get the facts at www.jstamp.com/reality.htm

> JStamps are under \$100 (10+). Development kits are about \$299. All are available now.



JRealTimetm

See the entire JRealTime family at www.jrealtime.com

SYSTRONIX[®]

Salt Lake City, Utah, USA Tel:+1-801-534-1017

Java is a TradeMark of Sun Microsystems, Inc. JRealTime, JStamp, SaJe, JStik are TradeMarks of Systronix, Inc The RoCK implements a pair of full H-Bridge motor drivers made from discrete NPN and PNP transistors. The H-Bridge provides bidirectional control for the two permanent magnet DC drive motors in the mobility base. Motor drivers are protected from short conditions by a PTC fuse, which resets

automatically. The fuse also serves as a low-side current sense resistor. Voltage drop across the fuse corresponds to motor current consumption. During a collision, motor torque increases substantially, thus increasing current draw, with a resultant increase in the voltage drop across the fuse. The built-in comparator of the AVR detects when the motor current draw exceeds the current reference, indicating a collision.

You may choose to connect the RoCK to any of a variety of different sized motors with differing current requirements. To ensure reliable collision detection, you can adjust the current trip point. This is accomplished by



Photos 3a and b-The circuit board and sensors are mounted in a plastic enclosure.

using a potentiometer to form a userset current reference voltage. Without the over current-based collision detector, you would be left the complex task of constructing a bumper for the RoCK, a process known to confound even experienced robot builders.

The RoCK uses a novel discrete component, half-duplex, RS-232 level converter to enable serial communication while reducing cost and board space. Given the RoCK's modest communication needs, the limitations imposed by half duplex are inconsequential.

In the behavior-based programming scheme used by the RoCK, all primitive behaviors are considered to be operating



in parallel. Each can compute an output for the motors, typically the output's conflict. An arbitration behavior decides which of the competing behavior wins control (see Figure 4).

Choosing a list of behaviors, ordering their priorities, and setting behavior-related parameters specifies a partic-

ular robot control program or task (see Table 1). The reusability of these components enables you to store many tasks in the robot. We'll explain this perhaps unfamiliar control scheme in more detail later on in the series.

The host interface operates through the serial interface. This gives you access to the microcontroller's static RAM (which includes the control registers) and read access to EEPROM. EEPROM write access is provided by a small amount of included code.

WHEN WE RETURN

Now that you've seen the big picture of the RoCK, we will take a closer look at some of internal details next month. Anyone interested in buying the RoCK should visit our web site at www.wirz. com/rock for more information.

Joseph L. Jones grew up in a small town in the Missouri Ozarks. He studied physics at MIT and received an BS in 1975 and an MS in 1978. He took a trip around the world, worked at the MIT Artificial Intelligence Lab, and is now senior roboticist at iRobot Corp. You may reach him at jlj@irobot.com.

Ben Wirz also grew up in a small town in the Missouri Ozarks. He studied physics and electrical engineering at Washington University in St. Louis, and graduated in 1997. He is currently employed as a senior electrical engineer by iRobot in addition to running his company, Wirz Electronics. You may reach him at ben@wirz.com.

SOURCE

AVR AT90S8535 Microcontroller Atmel Corp. (408) 441-0311 Fax: (408) 436-4200 www.atmel.com



Virtual System Modeling

Build It In Cyberspace

Web: www.labcenter-electronics.com

Develop and test complete micro-controller designs without building a physical prototype. PROTEUS VSM simulates the CPU <u>and</u> any additional electronics used in your designs. And it does so in real time.*

- CPU models for 8051/8052, Mc68hc11, PIC and Atmel AVR series micro-controllers available now. More CPU models under development. See website for latest info.
- Interactive device models include LCD displays, RS232 terminal, universal keypad plus a range of switches, buttons, pots, LEDs, 7 segment displays and much more.
- Extensive debugging facilities including register and memory contents, breakpoints and single step modes.
- Source level debugging supported for selected development tools.
- Integrated 'make' utility compile and simulate with one keystroke.
- Over 4000 standard SPICE models included. Fully compatible with manufacturers' SPICE models.
- DLL interfaces provided for application specific models.
- Based on SPICE3F5 mixed mode circuit simulator.
- Integrates with PROTEUS PCB design to form a complete electronics design system.
- VSM integrates with LabCenter's Schematic Capture tool ISIS, the Mixed Mode Circuit Simulator PROSPICE and the PCB design tool ARES.

<complex-block>

*E.g. PROTEUS VSM can simulate an 8051 clocked at 12MHz on a 300MHz Pentium II.

LabCenter products distributed in the USA, Canada and Mexico by R4 Systems Inc.



Write, phone or fax for your free demo CD - or email info@r4systems.com. Tel: 905.898.0665, Fax: 905.898.0683 Web: www.labcenter-electronics.com.

FEATURE ARTICLE

James Wilson

Cyber Robotics

Experimenting with Robot Simulation

If you could do all of the testing and training for a robot in a simulated environment, you would be able to cut down the design time, and therefore reduce the development time and cost. James introduces an open-source project that allows you to first work with a virtual robot.

ne of my fondest wishes is to one day (preferably) create or purchase a robot assistant that would follow me around and remind me of upcoming tasks while remembering requests that I have verbalized throughout the course of the day. Such a robot would be equipped with state-of-the-art sensor technology, and would require high-end computing resources to cope with both the outside world and its demanding creator. After a thorough web search for robotics companies equipped to provide a robot of this caliber, I found that prices range from \$1000 for a minimal sensor suite to upwards of \$20,000 for a multi-modal sensor suite maintained by an onboard Pentium III-class motherboard.

Even at the \$1000 price point, my budget would be strained, so I began to consider developing this robot in cyber space as a mere simulacrum running on my existing PC. This would give me the flexibility to experiment with a variety of robot designs, functioning in an environment similar to the layout of my apartment. It would also allow me to research the use of adaptive mobile robot software based on neural nets and genetic algorithms without the need for building and debugging on a remote target platform contained in the robot chassis. More importantly, I would be able to test the adaptive effectiveness of the software by running it in an accelerated time frame, thus simulating many days worth of robot learning in a matter of minutes.

After a brief search on the 'Net, I found that the commercially available robot simulation software was also beyond my budget, ranging from \$2000 to \$5000. Thankfully, the Rossum project was a notable exception and was actually available as open source. This article will provide you with an overview of the Rossum project's impressive capabilities along with a sample Robot simulation project to get you started building those imaginary, but nonetheless impressive, robot assistants.

The Rossum project is an open source project hosted by Source Forge at rossum.sourceforge.net and is the brainchild of G. W. Lucas. As described on its home page, Rossum is a project dedicated to the redistribution of ready-touse software components. To date, its primary focus has been the 2-D robot simulator called Rossum's Playhouse (RP1). By the time of publication of this article, RP1 should be available in V.0.50. For more information about the Rossum project, read the "Inspiration for Rossum's Playhouse" sidebar.

RP1 may be downloaded in source and non-source forms along with a PDF file containing user documentation that may be downloaded separately. Refer to the user documentation for additional information on many topics not provided in this article. The web site also provides download and installation instructions.



Photo 1—The RP1 server renders this sample floor plan of the "white room."
AN OVERVIEW

From the beginning, RP1 was intended to evolve into an opensource collaboration. So, the overall architecture needed to permit contributions by a disparate group of developers, sometimes working in different programming languages. To address these requirements, RP1 was written in Java using a client/server model. As depicted in Figure 1, the simulator (written in Java) acts as the server and communicates with separate clients using TCP/IP connections. The clients may be viewed as implementations of virtual robots or autonomous agents that, because they run as separate processes, may be written in any language.

The RP1 protocol deliberately avoids Java-specific features such as Java object serialization, and depends on data primitives that can be represented in most modern programming languages. In fact, I wrote an RP1 API in C to allow the RP1 to communicate with Win32 and Windows CE clients.

RP1 is capable of simulating the 2-D environment of the robot and its body. The environment is represented as a floor plan defined using a specialized syntax that describes the walls, targets, and obstacles of the 2-D environment. The robot body is represented as a basic geometric form with a predefined wheel system, and can be equipped with contact, range, target, and paint sensors. When the simulation is running and the client is in communication with the RP1 server, your robot is accurately portrayed as an on-screen graphic moving around in its 2-D environment.

RUNNING RP1 SAMPLE CLIENTS

In order to run the RP1 server, the Java Virtual Machine (JVM) must have already been installed. If this is the case, you should be able to run the batch files provided for the fire fighter and line tracker sample RP1 clients. The command in each batch file conforms to the syntax of the following:

java Server -p [name of RP1 properties file].ini server java Server -cp [name of the RP1 simulator jar file] -p [name of RP1 properties file].ini

The first command is used with the RP1 source distribution. The second command is used with the non-source distribution containing a Java jar file. The properties file referenced on the command line provides additional information on how to handle your client. Many options can be configured from the properties file, including the name of your client's main Java class file, how to communicate with your client (either internally or using network

Listing 1—Two types of statements, specifications and declarations, are used to define an RP1 floor plan.
/*
Specification syntax:
{specification name}: {parameter name}, [parameter name],
[parameter name], ...;
Example specification indicating that object dimensions are in meters.
*/
units: meters;
/*
Declaration syntax:
{object class name} {object name} {
 {specification name}: {parameter name}, [parameter name], ...;
Example declaration indicating that object dimensions are in meters.
*/
wall a { geometry: 0.0, 0.0, 3.0, 0.0, 0.05; }

I/O), and whether or not logging output may be combined with the logging output of the server.

A second properties file configures the operation of the RP1 server. The most interesting option available is the simulationSpeed entry, which allows you to control whether your RP1 client runs in real time or with an internal clock that runs faster or slower than real time. The value assigned to the simulation speed may range from 0.001 to 1000.0 and refers to the ratio of a simulated second to that of a real second (simulated seconds divided by real-time seconds). A simulation speed value of 2.0, for example, will allow the simulation sequence to run twice as fast. You will find this option most valuable when developing navigation algorithms that apply soft computing principles to produce adaptive behavior. With the simulation speed running fast, you can simulate many hours of learning in a matter of minutes.

Precisely how fast your simulation may run depends in part on the horsepower of the computer(s) running the client and server processes, along with the complexity of your virtual robot and simulation environment. In most cases, factors of 10 to 20 are achievable; higher values may be used to simply direct the simulator to run as fast as possible. When the accuracy of the timing in the simulation is critical, it is helpful to slow the simulation speed (using values of less than one) to reduce the noise introduced by system task management and communication overhead.

For a list of all of the defined properties file entries for the server and client, refer to the RP1 documentation.

FLOOR PLANS

No robot simulacrum can exist without somewhere to exercise its sensory and locomotive capabilities. This is where the RP1 floor plan comes in. A floor plan, as the name implies, is a complete definition of the objects and characteristics of the environment where your robot will navigate. RP1 requires that you provide this definition as a text file that contains predefined tokens positioned according to a simple grammar. Two types of statements, known as specifications and declarations, are defined for floor plan files. Specifications may appear as standalone or contained within the body of a declaration. In standalone form, specifications define global characteristics of the floor plan, such as the units used for the dimensions of the objects described elsewhere in the floor plan file. Listing 1 provides the syntax of a specification and

an example indicating that the dimensions of the objects defined throughout the floor plan file are in meters.

Declaration statements are used to describe floor plan objects. In this context, the term object is applied broadly to refer to any localized area of the floor plan that may be described by an RP1 declaration. It contains the object type, name of the object (used later when referring to the object in other declarations), and one or more specifications (described above) to define the characteristics appropriate for the object type named in the declaration. The declaration, which appears at the bottom of



Figure 1—The RP1 client/server architecture allows client processes to run on a remote or local host to communicate with the RP1 server over a TCP/IP connection.

Listing 1, defines an object of type "wall" named "a". The beginning coordinates are (0.0, 0.0) meters (meters is used assuming that the unit specification at the top of the code is in effect) and ending coordinates of (3.0, 0.0) meters. The thickness is 0.05 meters.

Six types of objects may be defined with a declaration, including a wall, target, floor paint object, placement, nodes, and links. A wall declaration defines a rectangular wall and contains a geometric specification defining the coordinates of the beginning and ending of the wall and its thickness. A target declaration describes a

INSPIRATION FOR ROSSUM'S PLAYHOUSE

The inspiration for the RP1 simulator came about during an attempt to develop algorithms for the Trinity College Fire-Fighting Home Robot Contest (developed by Jake Mendelssohn and David Ahlgren), one of the most challenging and elegant of the annual robot competitions at various universities. The idea of the contest is to exercise, on a small scale, the basic functions that would be required of a robot fire fighter stationed in a warehouse or home. In the contest, a small robot searches a mock house to find and extinguish a fire represented by a lighted candle. Scoring is based on speed, reliability, and consistency.

One of the contest founders, Jake Mendelssohn, stresses that the Trinity competitors are true robots, the key element being their capacity to perform autonomously. Trinity competitors are not remotely operated vehicles, but are driven by software implemented on PCs or onboard processors.

The elegance of the concept for this competition is in the broad set of problems—sensing, navigating, and interacting with its environment—for real-world robots while still permitting competition by robot builders at many different skill levels. Student-level teams have fielded successful entries, yet experienced engineers and academicians still find that the contest represents a challenging and interesting problem.

The 2001 contest featured more than 100 different robots. Contests in the Trinity format have sprung up across the U.S., Europe, and as far away as Buenos Aires, Tel Aviv, and Beijing. circular region of the floor plan that the robot may detect with a target sensor. It effectively provides a means of modeling a point source of visible or infrared light, although RP1 doesn't provide a target sensor for a specific wavelength of light. Considering the object-oriented design of the RP1 library, it would be simple to define such a sensor (I'll come back to this).

A placement object defines a location(s) in the floor plan that may act as a reference point in your RP1 client code for positioning and orienting the robot. To begin the simulation, you must position the robot at a placement object. A placement declaration is defined by four specifications as depicted in the sample white room floor plan file in Listing 2. The first three values of the geometry specification define the x and y coordinates of the center of the robot when positioned at this placement and the orientation of the robot (in degrees). The last value defines the radius of the home plate shape used to depict the location of the placement (see Listing 2 and Photo 1).

Floor paint is an object type that allows a region of the floor plan to be defined as a polygon. If the robot simulacrum is equipped with a specialized sensor, an event is generated when it enters this region, thus, allowing your navigation algorithms to alter its path based on whether or not a particular region has been entered. The geometric specification may contain an arbitrary number of points allowing irregularly shaped regions to be defined.

You may also associate the region with an integer value by using the optional region specification. This value may be used later to indicate that a paint sensor is able to detect floor paint only with the specified region value. By using multiple overlapping regions, you also may represent areas of curved lines or tracks. The line tracker floor plan (contained in the file FloorPlans\Line Track.txt) cleverly uses two regions of the same shape, one contained within the other, to create the appearance of a racetrack, which the robot dutifully tracks as it navigates the floor plan.

Node and link objects are provided to support floor plan notation. Effectively this allows you to draw on your floor plan with points that appear with designated labels and lines to interconnect them (lines are optional). This is particularly handy if you're creating a floor plan that is defined to scale and you need some means of visualizing the progression of the robot as it proceeds toward a particular check point at a precise number of meters from your starting point. The Trinity Fire-Fighting Robot Contest floor plan contained in the file FloorPlans\trinity2001. txt, is an excellent example of the use of floor plan notation to mark the ideal path around the walls and the locations at which fire detection should occur.

After you define the robot floor plan, you'll need to decide what sort of locomotion and sensor capability to give your robot simulacrum. As of this writing, RP1 provides support for only the classic two-wheel actuator with differential steering. This can be a serious limitation if your navigation algorithms depend on a particular feature of your wheel actuators (e.g., holonomic or hexapod drive systems). If you're careful, however, and develop the navigation algorithms for your physical robot to use some sort of software abstraction of your wheel actuators and its associated steering mechanism, this may not be a problem.

The robot body may be any simple (nonintersecting) polygon and can be equipped with contact, range, target, and paint sensors. The contact sensor provides an indication of physical contact with walls defined in the floor plan and may be used to model bump switches located on the outer perimeter of the robot platform.

The range sensor provides an indication of an obstacle within a maximum range as defined by your RP1 client software. It may be used to model infrared or optical sensors. This range is divided into a series of equally sized bins, each of which produce separate notification events, thereby controlling the resolution of the simulated range sensor and the number of events generated as an

Listir	ng 2—For the "	white room" sa	ample floor	plan, these are	the RPI dec	larations and	d specifica	tions.
/*	White Ro	om						
	Sample f This enc	loor-plan oding is	n file based o	for Rossum on the RP1	n's Play rev 0.4	house 1 floor	plan f	format.
*/								
uni cap	ts: meters tion:"Whit	s; te Room (RP1 rev	/. 0.4)";				
wal wal wal wal	1 a { geon 1 b { geon 1 c { geon 1 d { geon	metry: metry: metry: metry:	0.0, 3.0, 3.0, 0.0,	0.0, 0.0, 3.0, 3.0,	3.0, 3.0, 0.0, 0.0,	0.0, 3.0, 3.0, 0.0,	0.05; 0.05; 0.05; 0.05;	} } } }
wal	l e { geoi	metry:	1.0,	2.0,	2.0,	1.0,	0.025	;}
tar }	get goal label: geometry: color: lineWidth	{ "Goal" 0.4, red; : 3;	; // 0.4, 0.	F is for 25;	fire			
pla }	cement hor label: geometry: color: lineWidth	ne { "Home" 2.5, 2 green; : 3;	; .5, 225	, 0.275;				

ROBOT ENTHUSIASTS





Mobile Robots: Inspiration to Implementation, Second Edition Joseph L. Jones, Anita M. Flynn, Bruce A. Seiger paperback; 486 pp.; \$34.00, £25.00, €41.00

Using photographs, illustrations, and informative text, *Mobile Robots* guides the reader through the step-by-step process of constructing two different and inexpensive yet fully functional robots.



Receive \$50 Off the List Price if you mention this ad when ordering Rug Warrior ProTM



Rug Warrior Pro™ Entire Kit \$599,000, *Special Offer: \$549.00 Brains and Brawn available separately: The Brains: \$359.00 The Brawn: \$240.00

The Rug Warrior Pro kit contains the complete processing, memory, and sensor circuitry (Brains), as well as the motors, wheels, chassis, and custom body parts (Brawn) to build Rug Warrior Pro (as featured in the bestselling book *Mobile Robots: Inspiration to Implementation*).

"The kit is easy to build in a weekend and provides a powerful platform for learning and playing with robotics and artificial reasoning." –Alan McDonley, Moderator of www.egroups.com/groups/rugwarrior





A K Peters, Ltd. 63 South Ave. Natick, MA 01760-4626 Tel: (508) 655-9933 Fax: (508) 655-5847 service & akpeters.com obstacle is approached. Figure 2 illustrates the use of bins when defining the characteristics of a range sensor.

Target sensors are similar to range sensors, but differ in that they are used for detecting only target objects (defined using the target declaration in the floor plan file). Target objects model a point source of light; the target sensor is designed to detect this light within the area of a specified arc. Bins are used for target sensors as they are in range sensors, but in two distinct dimensions. The first dimension divides the arc and the second a range from the sensor to the target. Figure 2 depicts the existence of bins separately dividing the arc and range of the target sensor.

Paint sensors model a sensor specifically oriented to detect a region of color on the floor of the exercise area, as in an optical sensor pointed toward the floor to trace the length of a dark line. When the robot enters or exits a polygon, an event is generated. The line tracker sample RP1 client uses paint sensors to trace a curved line around the exercise area.

The paint sensors also allow you to tune the sensor to particular regions. In the floor plan, you may supply an optional region specification and give it an integer value. In the RsBody PaintSensor class, you can use the setRegionSensitivity command to tell the sensor that it can detect painted areas only with that region value.

RP1 SIMULATION CLIENTS

RP1 is designed using a client/server architecture in which the robot simulacrum runs as a client process of the separately running RP1 server process. The easiest way to write an RP1 client is to write it in Java using the Rossum package, contained in the file rossum.jar. (The entire RP1 project is also provided in a separate jar file named for the version of the software, such as sim049.jar.)

The Rossum package provides the class hierarchy used for the implementation of the RP1 simulator. In addition, it provides many base classes that assist with certain setup and housekeeping functions required of RP1 clients. Thankfully, the client/

Event/Request Object (Java)	Event/Request Code (C/C++)	Description
RsPaintSensor	EVT_PAINT_SENSOR	This event is generated when your robot's paint sensor detects a transition into or out of a defined region of floor paint, declared in your floor plan file
RsRangeSensorEvent	EVT_RANGE_SENSOR	This event is generated when a range sensor on the robot has undergone a state change caused by detection of an obstacle in the floor plan (e.g., wall). The frequency of the state change, effectively the resolution of the sensor, is controlled by the number of bins dividing up the maximum sensor range (discussed earlier in this article).
RsTargetSensorEvent	EVT_TARGET_SENSOR	This event is generated when a target sensor on the robot has undergone a state change caused by detection of a target object in the floor plan. The frequency of the state change, essentially the resolution of the sensor, is effected by bins in both the range and width of the sensor's active area.
RsContactSensor	EVT_CONTACT	This event is generated when the defined region of a contact sensor on the robot has under- gone a state change, caused by either contact (collision) with an obstacle or by removal of contact (release of pressure on the sensor).
RsMotionHaltedEvent	EVT_MOTION_HALTED	This event is generated when the motion of the robot has ended, either because a previous request for motion (RsMotionRequest/REQ_MOTION) has been completed, or because of a collision with an object in the floor plan.
RsMotionStartedEvent	EVT_MOTION_STARTED	This event is generated when the motion of the robot is started in response to an RsMotionReguest/REQ_MOTION reguest.
RsTargetSelectionEvent	EVT_TARGET_SELECTION	This event is generated whenever a request is received by the simulator to select a particular target (RsTargetSelectionRequest/REQ_TARGET_SELECTION), whether or not the target is already selected. This is commonly used to allow target selection through the GUI, allowing the robot (client) to react in someway when the event is received.
RsPlacementEvent	EVT_PLACEMENT	This event is generated after the RsPlacementRequest/REQ_PLACEMENT request has been processed by the simulator and the robot has been repositioned to the coordinates and orienta- tion of the placement object.
RsPositionEvent	EVT_POSITION	This event is generated in response to a RsPositionRequest/REQ_POSITION request and provides the absolute position of the robot in the floor plan. In the context of a robot navigation algorithm use of this request and event would of course be considering cheating.
RsHeartBeatEvent	EVT_HEARTBEAT	This event is produced periodically at the time interval specified in the RsHeatBeatReguest/REQ_HEARTBEAT reguest.
RsTimeoutEvent	EVT_TIMEOUT	This event is produced after a period of time has elapsed that is specified in the BsTimeBequest/BEQ_TIMEQUT request
RsPlanEvent	EVT_PLAN	This event is generated in response to an RsPlanRequest/REQ_PLAN request to obtain a copy of the fully parsed floor plan file. An abstract of the floor plan, represented as an object in Java or as a structure in C/C++, is provided in the event object/structure.
RsMouseClickEvent	EVT_MOUSE_CLICK	This event is generated when the end user clicks in an area of the floor plan. It is particularly useful when you would like the robot to respond in some way to this clicking action. The Demozero sample RP1 client (activated through FireFighter.bat) uses this event to cause the robot to reorient itself or move to a location in line with the position of mouse click, depending on which button was pressed.

Table 1-RP1 communicates with its clients through the events listed here. Certain events are generated in response to a specific request and others are sent unsolicited.

Working on your next masterpiece?



It's time for new tools.



multiSIM 2001 from \$399

SCHEMATIC CAPTURE, SIMULATION & PROGRAMMABLE LOGIC Advanced modeless schematic capture

- Library of 16,000 parts supplied, with 12 million online
 Analog, digital and mixed-mode simulation
 - Patented co-simulation of SPICE, VHDL and Verilog
 - Suite of "virtual instruments" including oscilloscope, wattmeter, spectrum analyzer & network analyzer
 - Design collaberation across the Internet

UltiBOARD 2001 from \$399

POWERFUL PCB LAYOUT

- Powerful & easy-to-use PCB layout & editing
 - Reroute while move (full rubberbanding)
 - 3D board visualization
 - Real-time design rule check
 - Component push & shove with springback
 - Extensive copper placement capabilities



ulti**route**

from **\$399**

- AUTOROUTING & AUTOPLACEMENT
- Benchmark test leader with superior routing resultsCombination of grid-based/gridless routing available
- Highly flexible router provides complete control
 - Optimal part placement improves routing performance
 Supports manual wire pre-placement
 - Pin and gate swap

COMMSIM 2001 from \$399

NETWORK & COMMUNICATIONS SIMULATION

- Powerful, yet easy-to-use fast, accurate simulation tool
- Model & simulate end-to-end communications systems
 - Analog, digital & mixed system design capability
 - Industry-leading block libraries channels,
 - encoders/decoders, modulators/demodulators

 View simulation results in various methods
 - time domain, frequency domain, xy plots, log scale, eye diagrams and power spectra



ics Workbench. Multisim and Ultiboard are registered trademarks of Interactive Image Technologies I td. All other trademarks are registered trademarks of



The World's Most Popular Electronic Design Tools

Electronics Workbench products are used by more designers than any other software of their kind. They will help you reduce development times and produce higher quality circuits — We guarantee it, or your money back!

We offer a portfolio of products to meet all your design needs: Schematic Capture, Component Database, SPICE/VHDL/Verilog HDL/RF Simulation, PCB Layout, Autoplacement, Autorouting and now Network & Communications Simulation.

Ease-of-use continues to be one of our biggest strengths, letting you produce designs in the time it takes to install and configure most other programs.

Our unique combination of power and unrivaled ease-of-use allows us to offer this guarantee. Power means you get capabilities simply not available elsewhere, such as patented co-simulation of SPICE/VHDL/Verilog/RF all together.

That's why we are the EDA supplier of choice to over 150,000 users worldwide.

sales: 800.263.5552 • http://www.electronicsworkbench.com

server architecture of RP1 makes it possible to write RP1 clients in any language that's capable of communicating over a specified IP port.

Also provided are RP1 client libraries for the Windows desktop and Windows CE. Development of a client library for the Palm OS is currently underway. Now, let's examine sample RP1 clients written in Java and C/C++ for the Windows and Windows CE platforms. Considering that RP1 was implemented in Java, it will of course run on other platforms with Java support, but I'll focus on clients for the Windows desktop and Windows CE devices.

EVENTS AND REQUESTS

The primary method for your client to communicate with the RP1 server is through predefined requests. Your client sends requests to the server to obtain certain information about the state of the robot or to cause actuator movement. As we will soon learn, certain requests induce certain events to be generated by the simulator and received by your RP1 client. At other times a request may not produce the expected event (e.g., when the robot is stuck in a corner and unable to comply with your client's request for motion). Unsolicited events are produced by the server and sent to your



Figure 2—The resolution of the range sensor is divided into width bins for the angle of view and range bins for distance from the obstacle.

client when something happens in the robot's virtual world that might require your client code to react. Table 1 contains a list of events described in terms of their associated requests.

The primary responsibilities of any RP1 client are initialization, event registration, and event handling (see Figure 3). After the client is initialized, its registered event handlers will be called when the associated state change occurs for the specified event. If the client is a simple one, it may be called only for the most basic start and stop events, as is the case for the sample Drunk client described next.

Let's begin our discussion of how to write an RP1 client by examining Drunk, a sample Java RP1 client (provided in the current RP1 download). You don't have to know Java because I'll describe the code in terms of the functional requirements common to all clients written in any language.

AN RP1 SAMPLE CLIENT

Drunk, as its name implies, performs a random walk about its environment using two events and one request. Motion is initiated relative to the robot's current position by sending an RsMotion Request command to the server. The server responds with an RsMotionStarted Event to inform Drunk that motion has actually begun, effectively confirming that the robot was not stuck. Later, when the robot makes contact with an obstacle or has successfully moved (or pivoted) the requested distance (randomly derived by Drunk), an RsMotion Halted event is sent by the server to Drunk.

Drunk's event handler then calculates the next randomly derived distance and orientation and sends another RsMotionRequest command to produce movement. Once again, an RsMotionStarted event is generated after the server completes its processing of the motion request and movement has begun. Drunk's handler for this event simply increments certain variables used to track the progress of the simulation. Figure 3 illustrates the messages exchanged between the Drunk client and RP1 server in a typical navigation sequence when Drunk collides with a wall, something it does often.

The DrunkMain class inherits from RsClient (principal client-communications class of RP1) for initialization and event handler registration functionality (see Listing 3). DrunkMain uses Java's primary entry point for an application, the main method, to call the initialization method in the RsClient parent class. This method will load the rossum. ini properties file in the Rossum package and will attempt to connect to the server. It is not uncommon for the initialize method to be overridden in the client's derived class to allow a local properties file to be loaded before calling the initialize method of the parent class. An example of this can be found in the Line Tracker sample client located in linetracker/LtMain.java.

After the initialize method runs successfully, the run method is called. The first task performed by this method is to construct the body of the Robot simulacrum by calling the build static method in the ClientZero object (located in clientzero/ClientZero.java). ClientZero provides utility functions commonly used in the implementation of other clients. In this case, you're borrowing the body design produced by ClientZero and encapsulated in the RsBody object value returned by the build method. (RP1 body construction code is discussed in more detail when the client code written in C is presented.) After the body is constructed, the sendBodySpecification method, implemented in the RsClient parent class, is called. This method causes the contents of the body object to be encoded and transmitted to the server.



Figure 3—In terms of the message sequence for an active simulation, the sample RPI client and server relate to each other as shown.

Mixed-Signal Controllers

MSP430 offers 50X more throughput.



MSP430—the choice in ultra-low-power Flash MCUs

Experience the ultimate SOC solution for battery-powered measurement. A flexible clock system switches from ultra-low-power standby to high-performance signal processing in less than 6 µs. Embedded emulation reduces design cycle time. Get your design started today with the easy-to-use MSP-FET430P120 Flash emulation tool.

Device	Flash	RAM	1/0	WDT	Timer_A	USART	ADC	Price
MSP430F1232	8 kB	256	22	~	~	1	10-bit	\$2.79
MSP430F1222	4 kB	256	22	~	~	1	10-bit	\$2.62
MSP430F1132	8 kB	256	14	~	~	-	10-bit	\$2.48
MSP430F1122	4 kB	256	14	~	~	-	10-bit	\$2.24

THE WORLD LEADER IN DSP AND ANALOG

MSP-FET430P120 development tool-\$99



MSP-FET430P120 development tool for \$99, product bulletin, F12x2 Datasheet and FREE samples

- 8-kB ISP Flash
- 0.8-µA standby mode, 250-µA active mode
- On-board brown-out detector
- USART can be used in UART/SPI™ mode



www.ti.com/sc/hpa7554u 1-800-477-8924 ext. 7554



New Communications Modules... Visit Our Website! Castatistificaning navious as a statistic to the second statistic terms in the second statistic to the

CHINESE PROCESSING STATES STATES AND STATES

Use Your Imagination.



Embedded controllers with an attitud

With a bit of imagination and a relentless pursuit of innovation, the divers that film Shark Week™ for the Discovery Channel figured out how to extend their dive times from only 30 minutes to eight hours by creating a re-breather that uses PicStic 4[™] technology.

With over 250,000 controllers in the marketplace, Micromint has been providing innovative, turn-key solutions to the OEM market for over twenty years- from design through production, as well as packaging and shipping the final product. Our broad line of embedded controllers and turn-key solutions can turn your imagination into reality.



Visit our website @ www.micromint.com to see our complete line of OEM Solutions.

As mentioned earlier, the RP1 server communicates with a client by sending events. After the body is established, DrunkMain registers event handler classes so that it may respond to those events. The methods, addMotion HaltedEventHandler and addMotion StartedEventHandler, register event handlers for the MotionHaltedEvent and MotionStartedEventevents. The Drunk client handles these events simply to detect when the robot simulacrum has started moving and stopped moving. Next, the addPlacement EventHandler method is called to register an event handler for the

```
Listing 3-The Java source code implements the main routine for the sample RP1 client, Drunk, and regis-
ters certain event handlers. The simulation is initiated with a call to the sendPlacementRequest method.
public class DrunkMain extends RsClient
//Implement a main to allow this class to serve as an application.
public static void main(String args[]) throws IOException
     DrunkMain c = new DrunkMain();
     c.initialize(); //throws IOException if unable to reach server
     c.run();
//code omitted for brevity
  public void run(){
     body = ClientZero.build();
      sendBodySpecification(body);
      addPlacementEventHandler(newDrunkPlacementEventHandler(this));
      addMotionHaltedEventHandler(newDrunkMotionHaltedEventHandler
      (this)):
      addMotionStartedEventHandler(newDrunkMotionStartedEvent
      Handler(this));
      try{
         int seed = rsProperties.extractInt("seed");
         log("DrunkMain establishing random series with seed
           seed"+seed);
         random = new Random(seed);
      }catch(RsPropertiesException rpe){
         random = new Random();
      sendPlacementRequest("home");
      super.run();
  }
  protected long nextMove(){
   RsMotionRequest
                      r;
      double
                         x. v:
     boolean
                         pivot;
      do{
        pivot = (random.nextDouble()>0.75);
        if(pivot){
        x=2*random.nextDouble()-1.0;
        y=2*random.nextDouble()-1.0;
        r = body.wheelSystem.getMotionRequestForPivot(false,x, y,
           0.2):
         }else{
        x=(2*random.nextDouble()-1.0)*0.25;
        y=(2*random.nextDouble()-1.0)*0.4;
        r = body.wheelSystem.getMotionReguest(false, x, y, 0.2);
      }while(r==null);
      sendMotionRequest(r);
      return r.durationMillis;
  }
  public RsBody
                               body;
  public Random
                               random:
}
```

PlacementEvent event. This handler responds to the placement of the robot at its starting position by generating the first randomly derived movement.

Recall that when creating an RP1 floor plan, a placement object may be declared in the floor plan text file. The Drunk client uses the location of this placement object as its starting position for the simulation sequence. When the sendPlacementRequest method is called at the end of the run method, the robot immediately moves to the placement object called home (declared in the floorplans/trinity2001.txt file), which enables the DrunkPlacement EventHandler event handler object.

The last line of the DrunkMain.run method contains super.run(), which calls the run method of the parent class. This method never returns and continually decodes data received from the server. In some cases, this data is integrated into event objects that are then passed to the event handlers.

To understand how event handler code is written in Java, let's examine the source for the DrunkPlacment EventHandler object (see Listing 4). Event handler objects inherit from a class named for the associated event, RsPlacementEventHandler in this case. The RsTransactionHandler interface is included in the inheritance hierarchy and requires that all derived event handler objects implement a processTransaction method. This method casts the RsTransaction parent class to the derived RsPlacement Event class (defined by this event handler) and calls the process method.

At this point, the absolute coordinates of the placement object are obtained from the RsPlacementEvent object (referenced in the event parameter) and the setMotion method is called to instantly position the robot at these coordinates. The RsMotionNull object is passed to cause all previous coordinates used for relative positioning to be ignored. Then, the nextMove method is called (see Listing 3).

Movement of your Robot simulacrum is accomplished by sending a request to the server. Movement occurs relative to the current position of the robot for a more realistic model of how real movement would be commanded. The use of a request is appropriate because it's possible that an obstacle may interrupt your robot's movement. If so, the MotionHalted Event event is generated and your code's registered Motion HaltedEvent Handler is called. The nextMove command in the drunk\ DrunkMain.java module calculates the relative coordinates of its next movement using the random method. First, it determines if the robot should pivot. If so, an RsMotionRequest object is initiated using the getMotion RequestForPivot command, which is implemented in the RsWheelSystem class. Other-wise the system calls the getMotionRequestObject command.

The x and y coordinates passed to the getMotionRequestForPivot method represent the location that you want the robot to face and will cause the robot to pivot at the rate specified in the last parameter as meters per second. If, for example, the coordinates of (0.0, 1.0) are specified, you are requesting that the robot face the location that is 1 meter directly to the left. Figure 4 illustrates the coordinate system relative to the robot's position and orientation. A robot is like a human being in that it perceives the world from its own frame of reference. Thus, for RP1, movement is always specified using relative rather than absolute (relative to the floor plan) coordinates.

The coordinates for the getMotion Request method are processed differently. They represent the desired offset from the current coordinates of the robot. So, if a request was sent for the robot to move to position (-1.0, 0.0), you would be requesting that the robot move 1 meter backward from its current position. If, however, you requested movement to position of (1.0, 1.0), a path would be calculated by the simulator to get the robot to a position 1.4 meters from its current position, 45° to the front and left.

With your actual robot, you would have to figure out the relative rotational velocity of each wheel to produce the right degree of curvature that leads to the specified coordinates. In the virtual world of Drunk, the RP1 simulator does this work for you when you call the getMotionRequest utility method in the RsWheelSystem class. If you use the getMotionRequest UsingWheelSpeed method, however, you can provide separate speeds for the left and right wheels, along with the amount of time both wheels should continue rotating.

If the Drunk robot simulacrum were equipped with any sensors, the simulator would generate additional events for each sensor. Additionally, you would require separate event handlers in your client. The Paint SensorEvent handler, for example, in the line tracker sample client uses its handler to update public variables in the main client class (LtMain) with the state of the paint sensor described in the RsPaintSensorEvent object. Later, the HeartBeatEvent handler uses this information to formulate the next request for motion.

CLIENTS WRITTEN IN C

The RP1 distribution includes an RP1 library written in C (RCAPI). It supports the creation of RP1 Win32 clients written in C/C++ for both a Windows desktop and Windows CE device. Creating an RP1 client for a CE device presents additional advantages because the same navigation algorithms can be tested on the actual robot by attaching your CE device to the robot platform. Carnegie Mellon University used a similar design to implement the Palm Pilot Robot Kit (PPRK) (www.cs.cmu.edu/~pprk). I adapted the PPRK platform to carry a Compaq Aero 1530 Palm-sized PC.

The primary difference between the Drunk RP1 client written in Java and the one written in C (under the RCAPI\ {CE|Win32}\RCAPITester directories) is the lack of an object-oriented construct from which to create derived classes that extend the functionality of existing parent classes. When the Java Drunk client was introduced, it was clear that a good deal of functionality was borrowed from the RsClient parent class. Although this is clearly not a requirement for developing an RP1 client, it simplifies creation of the code.

The existence of a C API for RP1 client creation is an important step, however, in achieving portability to low-end platforms in which the navigation algorithms are being developed for the target platform in C and need to be tested under simulation in the same language.

The RCAPI library (located in the RCAPI folder of the RP1 distribution) contains Windows desktop-specific code, located in the RCAPI\Win32 sub-directory, and the Windows CE-

```
Listing 4—The sample RP1 client Drunk uses this handler for the placement event. This event is generated
in response to the request sent by the call to the sendPlacementRequest method (see Listing 3).
public class DrunkPlacementEventHandler implements
RsPlacementEventHandler {
   public DrunkPlacementEventHandler(DrunkMain drunk){
       this.drunk
                         = _drunk;
   public void processTransaction(RsTransaction t){
       process((RsPlacementEvent)t);
   public void process(RsPlacementEvent event) {
       drunk.log("Received placement at "+event.name);
       drunk.body.setPlacement(true);
       drunk.body.setMotion(
           new RsMotionNull(
               event.simTime.
               event.orientation,
               event.x,
               event.y));
       drunk.nextMove();
   }
   protected DrunkMain drunk;
}
```

specific code, located in the RCAPI\CE sub-directory. The Microsoft Visual C++ and eMbedded Visual C++ make files are provided in the RCAPI directory, named RCAPIWin32.* and RCAPIWinCE.*, respectively. You also get an example client implementing the same robot simulacrum as the Java Drunk client for both the desktop and Windows CE in the RCAPI\{CE | Win32}\RCAPITester directories.

I tested the Windows CE client using the Win32 (WCE x86em) debug target and the Palm-size PC 2.11 emulation platform options. You shouldn't have any problems with other targets and platforms because this code provides only a minimal test harness user interface. When the build has successfully completed, the emulator for the selected platform automatically loads. An application should appear with various menu options that were automatically generated by the WCE MFC wizard tool. Consider that this effectively represents two levels of simulation, in that the CE platform emulation is a representation



Figure 4—Robots move similarly to people. RP1 coordinates are in the relative frame of reference of the robot, not the absolute coordinates of the floor plan.

of the robot's virtual target hardware, which in turn uses RP1 to simulate the robot's virtual world.

The one menu option of interest is the test menu option. When selected, the runRCAPITest function performs the same operations as the run method in the DrunkMain class. A connection is established with the server, the client properties file is loaded, the ClientZero body is sent to the server, and the RsClientMainLoop function is called to allow processing of incoming events.

The allocateClientZero function in the ClientZero.c module constructs the robot body and is called from TestMain.cpp (see Listing 5). Each piece of the robot body is created by calling utility functions with the prefix "build". These utility functions extend their simplified parameters with additional information needed for calling the functions with the prefix "RsBodyCreate" in the RsBody.c module. The sendClientZero function is called after allocateClientZero and is responsible for calling the functions with the prefix "RsSend" (in the RsBodySend.c module) for each of the body parts allocated in allocateClientZero and saved in a static pointer variable. The functions called next in TestMain.cpp should appear familiar because they have analogs in the Java RP1 client API.

When developing your own RP1 clients in C, most of your modifica-



> For more information and the contest rules, contact Jake Mendelssohn at: jmendel141@aol.com

tions will be focused on the ClientZero.c and TestMain.cpp modules. The latter contains the event handlers that will need to be augmented as you add more sensors and enhance the navigation algorithms of your robot simulacrum. The former contains a series of utility functions that would create a robot body identical to the one used in the Drunk client. You could modify the functions for any changes to the capabilities or appearance of your robot.

One other thing to be aware of with RP1 C clients is that your client is launched separately from the RP1 server, which runs in a JVM. This means that the . INI file (properties file) entries reserved for the dynamic loading of Java clients (with prefix "dlc") need not be assigned a value. See the RP1 user documentation for a complete description of all of the available properties file entries.

FUTURE OF ROSSUM

The Rossum project is an opensource project, which means that its success is largely determined by the contributions and hard work of both users and developers. The Rossum project has given us a framework from which to build many other capabilities, and there is no shortage of ideas on what to do next. Let me briefly list the most ambitious ideas under consideration now.

One plan is to enhance the RP1 base GUI to control more simulation options in real time, particularly those specified in the RP1 properties files. Another idea is to create a WYSIWYG floor plan authoring tool to allow the RP1 floor plan to be defined interactively, and then saved in the RP1 floor plan file syntax. People also are working on additional object types for the RP1 floor plan, such as walls with reflectivity, light absorption, scattering properties, and furniture that can be specified as moveable. Additional wheel systems beyond the two-wheel, differential system are in the works, as well.

Rossum's Playhouse (RP1) is the first software released by the Rossum project and represents a major step in providing an open source alternative **Listing 5**—When using the RCAPI, the robot body is constructed through a series of function calls that are preceeded with the prefix "build".

۷O	id	allocateClientZero(char *strNewBodyName)
l	/* dou	Radius of the circle of the robot body (0.30 metersacross). */ uble circleRadius = 0.15;
	/*	Build the container structure which describes the overall robot body. */
	rsE	<pre>3odyContainer = buildBodyContainer(strNewBodyName, 10);</pre>
	/* rs(rsl	<pre>Progressively build the robot body and retain the pointers to each structure encapsulating the details of each body component.These pointers will be used later in the transmis sion of the whole body definition to the RP1 server. */ Circle = buildPrimaryBody(circleRadius); WheelSystem = buildWheelSystem();</pre>
	/* rs(rs(ci rs(ci	Build each of the three contact sensors located along the circular perimeter of the robot's body, 120 degress apart. */ ContactSensor1 = buildContactSensor("Contact Sensor 1", ircleRadius, 0, 120); ContactSensor2 = buildContactSensor("Contact Sensor 2", ircleRadius, 120, 240); ContactSensor3 = buildContactSensor("Contact Sensor 3", ircleRadius, 240, 360);
	/* rsF cf rsF cf rsF cf	Build each of the four range sensors. One sensor is located every 90 degrees around the perimeter of the robot. */ RangeSensor1 = buildRangeSensor("Range Sensor 1", ircleRadius, RADIANS(45)); RangeSensor2 = buildRangeSensor("Range Sensor 2", ircleRadius, RADIANS(135)); RangeSensor3 = buildRangeSensor("Range Sensor 3", ircleRadius, RADIANS(225)); RangeSensor4 = buildRangeSensor("Range Sensor 4", ircleRadius, RADIANS(315));
}	/* rs1	Build the omnidirectional target sensor, designed to detect a point object. */ TargetSensor = buildTargetSensor("Target Sensor");

to expensive commercial robot simulators. RP1 provides a basic set of options for configuring a two-wheel, differentially driven robot simulacrum equipped with an array of tactile and optical sensors. The robot's virtual world is defined using a floor plan file that is displayed graphically in a 2-D rendering. RP1 clients are written in Java or C/C++ and may be used to test navigation algorithms prior to being run on a physical robot. Visit the Rossum project web site to learn more about all the features of RP1 and to download the latest RP1 release. **■**

Author's note: Many thanks to G. W. Lucas for making this article possible, both as the creator of the Rossum project and for his dedicated review of its content. James Wilson is a software engineer for Loronix Information Systems in San Diego, California. He coauthored Building Powerful Platforms with Windows CE. In his spare time, he also acts as a Microsoft eMVP for Windows CE. You may reach him at jywilson@ieee.org.

SOURCES

Aero 1530 Palm-sized PC Compaq Computer Corp. (800) 282-6672 (281) 370-0670 Fax: (281) 514-1740 www.compaq.com

The Rossum project

G. W. Lucas rossum.sourceforge.net



Thousands of peripheral combinations possible in a reprogrammable 8-bit microcontroller!

PSoC Microcontroller Family

Part	Flash Program Memory	SRAM (Bytes)	Pins	Single- Cell V _{CC}	Packages
CY8C25122	4KB	128		No	PDIP
	8KB				PDIP, SOIC, SSOP
CY8C26443	16KB	256	28	Yes	PDIP, SOIC, SSOP
CY8C26643	16KB				PDIP, TQFP,

PSoC User Modules

Digital User Modules	Analog User Modules
Timers	A/D: SAR, DS, Incremental
Counters	DAC
Pulse Width Modulators	Filters
PWMs w/Deadbands	Comparator
Serial Receivers/Transmitters	Gain stage
UARTs	U U
CRC Generators	
SPI	
Pseudo-Random Sequence Generators	

An 8-bit microcontroller with exactly the peripherals you need.

Cypress's innovative Programmable System-on-Chip[™] (PSoC[™]) MCUs are dynamically reconfigurable, with the ability to change peripheral definitions at run-time. You can customize your microcontroller in just ten minutes by simply choosing the pin count and memory sizes you need. Each PSoC MCU has digital and analog building blocks that can be configured as traditional MCU peripherals. They can even replace many external components, creating a true System-on-Chip (SoC) solution. A complete development kit, including software and a full-function ICE, is available for under \$200, and the C compiler is available for \$150. To learn more about our innovative PSoC solutions or purchase a development kit, visit our web site at **www.cypress.com/ad/psoc.**



FEATURE ARTICLE

Scott Vu & James Conrad

Racebot

A Two-Degree-of-Freedom Stiquito Robot

Some of the best designs are inspired by old favorites. Building on the classic Stiquito model, Scott and James modified the robot to walk with steps rather than push itself along. Read on to learn how they used the BASIC Stamp II to control the nitinol muscles. n the summer of 2001, James gave his small engineering class at North Carolina

State University an assignment to design and implement a functional robot for a robot race. This race had specific design rules and restrictions. The rules for the race and the requirements for the robot were that the robot must use nitinol (Flexinol) for locomotion, use legs in its propulsion (more than two legs required), and walk four times its length in the fastest amount of time on smooth Formica. Other demands were that it have an on-board microprocessor, stop walking via sensor input, and measure no greater than $12'' \times 12''$. We wanted to perform multiple trials and take the best time for competition comparison. The ability to add either an external (tethered) or attached power supply rounded out the list of requirements.

The summer course, titled "Simple Robots and Microprocessors," concentrated on the Stiquito robot (see Photo 1) and the BASIC Stamp II microcontroller. Naturally, most students used Stiquito as the basis of their designs. Stiquito is an inexpensive hexapod robot that uses nitinol for movement. [1] Nitinol is an alloy actuator wire made of nickel and titanium. This wire retains or remembers its original shape after bending. Jonathan Mills of Indiana University invented Stiquito, which means "little sticky."

The engine of Stiquito consists of nitinol and music wire. Nitinol contracts when current runs through it, and then returns to its original position when the current is removed and the music wire pulls the nitinol back. This action causes a forward and back pulling movement, hence the primary movements of Stiquito.

A Stiquito with two degrees of freedom uses a regular Stiquito design, with an addition of six more nitinol actuator wires that lift the legs off the ground. Two-degree movement means movement on both a horizontal and vertical planes (i.e., both back and forth and up and down). Using the two tripods of legs, tripod A is lifted off the ground while the leg is in the forward position. Then tripod B is moved backward while on the ground. Next, tripod B is moved up into the air and forward, and tripod A is moved down onto the ground and backward.

The previous steps are repeated beginning with tripod A. Moving forward means the leg relaxes and returns to the standard position. Moving backward means the nitinol tightens. Twodegree robots have a lot less friction than robots with one degree of freedom (movements on the horizontal plane only), and thus move faster.

The students decided that a microprocessor-controlled Stiquito with two degrees of freedom (using screws) was the most time-efficient and effective racebot. Designing the Stiquito with two degrees of freedom or any robot requires three parts.

The first and major part of this project is the mechanics, which consists of the whole frame of the robot. This



Photo 1—Stiquito is an inexpensive hexapod robot that uses nitinol for propulsion.



Photo 2—The BASIC Stamp II microcontroller contains a complete microprocessor system, including RAM, EEPROM, an operating system, serial port, and 16 input/output ports.

includes how the robot will look and function. The mechanics are important, because they're the basis for the other two design parts. If the mechanics do not work, the robot does not work. The second part is the electrical hardware, which consists of the microprocessors, chips, and circuitry needed to control the robot.

The third and last part of the design is the software. The software is not required if the robot is built with analog circuitry. However, software is needed for robots that use a microprocessor. The software obviously drives the microprocessor, which in turn drives the robot. Making the software efficient makes the robot efficient.

MECHANICAL DESIGN

The frame for the two-degrees-offreedom Stiquito racebot was built with the original one-degree-of-freedom Stiquito frame. The original robot had six legs and used a tripod gait, but also used two degrees of freedom. A tripod gait is the movement of three legs at the same time, the front and back legs on one side and the middle leg on the other side. Read *Stiquito for Beginners: An Introduction to Robotics* for the original one-degree-offreedom Stiquito design. [1]

For a Stiquito with two degrees of freedom, an extra 60-mm nitinol wire was attached to each leg. The extra wire was screwed to the upper surface of the Stiquito body for vertical movement. Two extra 28-AWG wires were attached to the bottom of the body to control the vertical tripod gait movements. A small piece of rubber was attached to the end of each foot to allow it to grip onto the surface. Two holes were drilled at each end of the body to attach the printed circuit board on top of the Stiquito.

ELECTRICAL DESIGN

The printed circuit board is a regular epoxy board (perfboard) that has several 1-mm holes used to construct circuitry from scratch. A microcontroller, ULN2803A chip, bump/switch sensor, as well as sockets were on the PCB. All of these were wire wrapped and soldered together.

The BASIC Stamp II microcontroller, sold by Parallax, controlled the legs of Stiquito. The dimensions of the BASIC Stamp II are $1.1875'' \times 0.625'' \times$ 0.375", and it weighs only 0.02 lb (see Photo 2). The BASIC Stamp II contains a PIC16C57 (interpreter chip) microcontroller, runs at 20 MHz, uses 2 KB of EEPROM, and includes 32 bytes of RAM. [2] The microcontroller interprets BASIC language programs for instructions. The BASIC Stamp II has 16 input/output pins and a total of 24 pins. It needs 7 mA of current to run, and 50 µA of current to sleep. It can also provide source current of 20 mA and sink 25 mA of current.

The ULN2803A, an eight-transistor Darlington array chip, was used as a source of current from a battery to each tripod because the current from the BASIC Stamp is not enough. [3] The primary function of the chip is to act as an inverter. If a digital one goes into the input pin, a digital zero goes



Figure 1—There are only two components of the Stiquito controller circuit, the BASIC Stamp II and a ULN2903A Darlington transistor array.



Photo 3—The completed Stiquito with two degrees of freedom and the microcontroller attached was able to walk 30 cm in 1 min. compared to about 10 cm per minute for a Stiquito with one degree of freedom. (The power tether is not shown in this photo.)

out the output pin, and vice versa. The ULN2803A chip has a total of 18 pins. Pins 9 and 10 are used for ground and power, respectively. Pins 1 through 8 are used for inputs, and pins 9 through 18 are used for outputs. Each input pin corresponds to the output pin directly across from each input pin. Refer to Figure 1 for the pin numbers. The chip requires at least 5 V to run.

The rules of the race required that each robot have a mounted sensor to stop the robot from walking. The sensor used for the two-degrees-of-freedom Stiquito robot was a bump/ switch sensor. The sensor was connected in parallel to a starting switch. The robot starts when the start push button is pressed, and it stops when the sensor switch is pressed.

Four sockets were soldered onto the printed circuit board. The purpose of the sockets was to easily remove the BASIC Stamp II and ULN2803A

> chip for testing and debugging. One of the sockets had the same dimensions and pins of the BASIC Stamp II, and the other socket had the same dimensions and pins of the ULN2803A chip. The other two sockets (two pins) were power sockets to attach the battery to the board. The power was provided by two 9-V batteries. One of the batteries drove the BASIC Stamp II and ULN2803A chip, and the other battery drove the legs. The batteries were attached to the PCB by

Want to migrate your Z180/HD64180 design to a solid Ethernet solution?

See why the Rabbit 2000™ is the new favorite of design engineers!

Rabbit 2000 Features

- Glueless architecture
- IMB code/data space
- 4 serial ports
- 40 I/O pins
- 7 timers
- Battery-backable real-time clock
- Watchdog
- Remote cold boot
- Slave interface

Cost Benefits

- Royalty-free TCP/IP stack with source code
- Feature-rich Dynamic C[®] SE software in Dev Kits
- No in-circuit emulators required
- Free tech support

Choose from our multiple low-cost Development Kits

All Kits Include: Rabbit 2000 development board Dynamic C® SE software (not a trial version) Peripheral development tools



Designers familiar with the Z180 or HD64180 will be completely at ease with the Rabbit's Z180style architecture, improved for higher performance. The Rabbit runs 3X faster and is loaded with on-chip peripherals. Powerful software support is provided in industry-proven Dynamic C[®], an interactive editor, compiler, and debugger.

VKITS

ORDER ONLINE @ www.rabbitsemiconductor.com

2932 Spafford Street, Davis CA 95616 • T: 530.757.8400 • F: 530.757.8402



Figure 2—To save battery power and protect the nitinol wire, we pulse-frequency modulate the current going to the nitinol wire. Note the variable delay.

way of four 34-AWG magnet wires. A schematic of the electrical design is shown in Figure 1.

SOFTWARE DESIGN

Scott wrote the program in the Parallax BASIC programming language. We used the software to control the movements of the legs, control frequency of current transmission to the legs, and detect sensor inputs.

Nitinol requires a lot of current; running nitinol robots can drain a 9-V battery in 15 min. It also needs cool ambient air to return to its normal position quickly. To compensate for these, the software uses pulse width modulation. By using pulse width modulation, the software lets just enough current to go through per time unit to contract the nitinol.

Sending a pulse to the nitinol for a number of milliseconds, pausing for the same amount of time, and then repeating the pulse again perform pulse width modulation. In order to jump-start the nitinol, a pulse of 20 ms on and 20 ms off is required. This is done five times. To sustain the nitinol in the current position, a pulse of 20 ms on and 80 ms off is required. This is done eight times. In order to move a tripod, a jump-start pulse and a sustaining pulse is executed. See Figures 2 and 3 for details about how we implemented PWM for this robot.

Many embedded systems can be characterized by using a state machine to transition between states. We used a state diagram to show the transition between these states and identified the control of the legs in each state. Because there are four movements per tripod, there are four states per tripod. Table 1 describes the states.

A one means the nitinol is activated. The state machine requires one initial state (all nitinol relaxed) and eight operational states. The different states can be represented visually as demonstrated in Figure 4. Follow the state diagram for programming. You may download the source code from the *Circuit Cellar* web site.

The software loops in the beginning of the code waiting for the Start button (pin 7) to be

pressed. After the pin registers a digital one, the leg algorithm starts executing. In order to detect the bump sensor more frequently, the software polls pin 7 every pulse. After a digital one has been detected by way of the bump sensor, the program stops the leg algorithm and loops for a digital one in pin 7 to start the legs again.

IMPLEMENTATION

Building the electronics took many of steps. You may download the parts list from the *Circuit Cellar* web site. To start, first solder pins 8 through 15 of the BASIC Stamp II socket to pins 8 through 1 of the ULN2803A socket, respectively. Solder every two pins together on the ULN2803A socket starting with pin 1. This connects two transistors for more current to the legs. The result from the ULN2803A is four outputs for the two tripods with two-degree movements.

Both pin 10 of the ULN2803A socket and pin V_{IN} of the BASIC Stamp II socket are soldered to the power of the battery number one socket. Pin 9 of the ULN2803A socket, the V_{ss} pin of the BASIC Stamp II socket, the ground of battery number two socket, and pull-down resistor are soldered to the ground of battery number one socket.

Solder the 34-AWG wire and one pin of the DIP switch (two pins) to the V_{DD} of the BASIC Stamp II socket. Next, connect the 20-AWG, the other pin of the DIP switch, and the power



Figure 3—The nitinol actuator responds to the heat generated by the current pulses and the loss of heat caused by convection of the wire.







of the pull-down resistor to pin 7 of the BASIC Stamp II. The pull-down resistor pulls down the voltage of pin 7 to its default voltage, because the pin's voltage tends to

Direction			Description
Down and forward	DF	00	Relaxed state for all legs
Down and back	DB	01	Legs are on the ground and pushing the robot forward
Up and back	UB	11	Lift the legs off the ground so that they do not pull the robot back when they go forward
Up and forward	UF	10	With the legs out of the way, return them forward to prepare to place them on the ground

Table 1—Each tripod can have four different states. The three legs of each tripod can be up or down and forward or back. Each leg has two pieces of nitinol to control this up/down and forward/back position. A zero means no current flows through the nitinol; a one means that current flows through and the wire contracts.

vary in an unknown voltage.

After you complete all of the soldering, you'll use pin 7 of the BASIC Stamp II as a detection pin. Every two pins starting with pin 8 through 15 will control one movement and one tripod.

It took us roughly 6 h to build the two-degrees-of-freedom Stiquito mechanics (specific steps can be found in *Stiquito for Beginners*). [1] The board took another 6 h to produce. Testing and tweaking the Stiquito took yet another 6 h. We completed the software in 15 min. The total estimated time is 18 h. Photo 3 shows the completed robot. For more information regarding the building process from two students' perspective, read the "Experiences with a Simple Robot" sidebar.

While building this robot we learned several tips that you should remember when building your own. The key to making the nitinol contract well (pulling 7 to 8 mm of music wire backwards) is sanding the nitinol well. When sanded well, the nitinol will appear white and make a better connection.

Our second tip is to test your code on a breadboard before you put it on your Stiquito robot. The breadboard should consist of the BASIC Stamp II, ULN2803A chip, and LEDs acting as legs. Neglecting to perform this test may damage the Stiquito's nitinol. In addition, do not tighten the vertical nitinol actuators too tight. The horizontal nitinol actuators break easily if this is done.

EXPERIENCES WITH A SIMPLE ROBOT

By Sarah Goodman and Priti Patel

At the end of the semester, professor James Conrad announced he would be holding a summer class on simple robotics. He made robotics sound so fun that we signed up for it. We first started by examining the Stiquito. Although it walks at a blazing 3 to 10 cm per minute, it is an amazing learning tool and a terrific way to introduce anybody to the field of robotics. A Stiquito robot is relatively easy to build, which makes it a great in a lab course or project.

During the early weeks of the class, we each constructed one of these small robots and made it walk. This gave all of us the experience of working with small parts and some basic knowledge of how to go about programming a robot to walk. Our final goal was to build a robot using nitinol as propulsion and an on-board microcontroller to compete in a race at the end of the summer (and maybe even win).

The robot for the race was not required to be a Stiquito, so some people opted to construct their own, larger versions of this cute little bug. However, the majority, including us, decided to keep the original Stiquito base design and modify it a bit. By adding a second degree of motion (upwards), the Stiquito no longer has to drag its other feet but can instead lift them into the air while moving forward to greatly reduce friction. Besides the addition of a second degree of motion and the use of screws in place of crimps, the body of the Stiquito remained essentially the same.

Because our Stiquito had a tripod gait and two degrees of freedom, it was necessary to design a state diagram to create a stable and efficient walking pattern. The chosen microcontroller brain of our Stiquito was the BASIC Stamp II. Using the Parallax "Board of Education" during labs, we became fairly proficient at programming this chip to light up several LEDs and make them blink in attractive patterns.

After you program it, you could pop off the BASIC Stamp II from the board and it would remain programmed to do whatever you wanted. In our case, the Stamp would control the gait of a Stiquito and detect the output of a bump sensor. After programming the BASIC Stamp II with similar code, we removed it from the Parallax "Board of Education" and placed it in our circuit board attached to the Stiquito. This was repeated many times as we began to make the final adjustments to give our Stiquito optimal performance for the race.

On that great day of glory at the race, our valiant Stiquito managed to stand at the starting line and pull off a convincing impersonation of a paperweight. When the batteries were connected, it failed to move at all. After examining our robot, we discovered that one of the wires in the circuit was soldered poorly. For this reason the Stiquito was receiving little to no power. The free pizza provided was our only consolation. The next day, after many repairs, our Stiquito managed to hobble in a circle on a lab table.

One of the most enjoyable classes of our college careers is now over. We still have that first Stiquito we built now sitting in a box as a memento. Two of the legs are not functioning, but he is still a nice reminder of all the fun (and pain) we had as we were introduced into the world of robotics.



Figure 4—Embedded applications are often designed using a state diagram. Each state represents the possible positions of the four groups of nitinol actuators. The top letters are one tripod and the middle letters are the other tripod. The four binary digits represent which of the four groups of nitinol wires are activated.

Next, check your board with a voltmeter. Be careful not to short your circuit connections on the board. Do not invert power and ground inputs. We advise you to test nitinol actuators with 3 V. Nine volts will burn a single nitinol actuator.

Another tip we learned is that the 28-AWG wire and 34-AWG magnet wires will snap easily, so be careful when twisting or turning them at a joint. Lastly, note that nitinol can get hot when current is running through it, so take care not to burn yourself.

RESULTS AND FUTURE PLANS

The Stiquito won second place in the race; it was able to walk 30 cm in 61 s. The first place robot won with an elapsed time of 41 s (and is the subject of an article we're working on). The third place robot took 61.6 s to complete the 30 cm. The reason for the second place finish was because of the way we implemented the state diagram in software. The nitinol contracted too long for the sustaining pulse. The pulse should have been done only five times rather than eight times. Also, we didn't allot enough time for the nitinol to cool in certain states. To make matters worse, the temperature in the racing room was high and the nitinol could not relax fast enough. If only someone would develop a solution for cooling nitinol faster.

Our future plans are to improve the software for the robot, especially a better implementation of the state diagram. We also noticed a direct correlation between the ambient temperature and nitinol relaxation time, so adding a simple thermal sensor to distinguish between 20°C and 30°C could further optimize the gait.

Scott Vu is an Electrical and Computer Engineering student at North Carolina State University. You may reach him at skvu@eos.ncsu.edu.

James M. Conrad received his BS in Computer Science from University of Illinois, Urbana, and his Master's and Doctorate degrees in Computer Engineering from North Carolina State University. He is currently a project manager at Sonv Ericsson Mobile Communications and an adjunct professor at North Carolina State University. He is the author of numerous book chapters, journal articles, and conference papers in the areas of robotics, parallel processing, *artificial intelligence, and engineering* education. You may reach him at jconrad@stiquito.com.

SOFTWARE

To download the code, go to ftp. circuitcellar.com/pub/Circuit_ Cellar/2002/141/.

REFERENCES

- J. M. Conrad and J. W. Mills, Stiquito for Beginners: An Introduction to Robotics, IEEE Computer Society Press, Los Alamitos, CA, 1997.
- [2] Parallax, Inc., BASIC Stamp Manual, rev. 1.9, Rocklin, CA, 1998.
- [3] SGS-Thomson Microelectronics, "ULN2801A, ULN2802A, ULN2803A, ULN2804A, ULN2805A—Eight Darlington Arrays," September 1997.

RESOURCE

Official web site for Stiquito www.stiquito.com

SOURCE

BASIC Stamp II Parallax, Inc. (888) 512-1024 (916) 624-8333 Fax: (916) 624-8003 www.parallaxinc.com



· PIC12 · PIC16 · PIC17 · PIC18

APPLIED PCs

Fred Eady

Replacing Relays with Ladder Logic

Part 2: The T100MD-1616+

Last month Fred introduced us to the T100MD-888+ and discussed some of the basic functions and potential of the PLC. This time around, he steps up to the 16 I/O port model and delves into a communications application that talks over an RS-485 LAN. ome of the greatest songs in musical history are performed as duets. Being a Tennessee boy, my musical mind recalls melodies from performers like George Jones and Tammy Wynette and the contemporary country sounds of Tim McGraw and Faith Hill. Of course, rock and roll had (and still has) its share of duet performers as well. Jan and Dean, the Everly Brothers, and Elton John with Kiki Dee come to mind.

Because there's math involved, I find that things that work well for music usually work well for electronic things. Although the previous statement is not one of the sacred laws of embedded computing, it should be. With that thought, let's put a pair of T100MD's together and make some control music with a PLC duet.

THE T100MD-1616+

In Part 1 of this series, I talked about the T100MD-888+, a programmable, Internet-capable PLC with eight physical inputs and outputs. As you might have grasped from its moniker, the T100MD-1616+ has 16 physical I/O ports. The basic operations I described last time also can be applied to the more robust T100MD-1616+ PLC you see in Photo 1.

The major differences between the T100MD-888+ and T100MD-1616+ can be found in the analog area. Unlike the T100MD-888+, the T100MD-1616+ demands a separate power supply for the analog components. Instead of the maximum of eight 10-bit A/D channels, the number of '1616+ A/D channels has been reduced to four. The first two 1-V full-scale analog channels are buffered with LM324 op-amps set for a gain of five. The remaining two channels are not buffered and accept 0- to 5-V inputs. All of the analog inputs are protected with a Zener diode just in case you pull a "Fred."

The two T100MD-888+ DACs are unbuffered outputs that you must condition. The first analog output on the T100MD-1616+ is configured at the factory to provide a 20-mA current loop signal. Current loop output allows a longer distance between nodes, as wire resistance does not inhibit its signal. The current loop output is converted to a voltage using a resistor across the output terminals. Ohm's Law prevails here and to achieve a 0- to 5-V output, a 250- Ω resistor would be used. The current loop DAC on the T100MD-1616+ is calibrated using an onboard potentiometer and a single line of TBASIC code, SETDAC 1,2048.

The second analog output of the T100MD-1616+ is an unbuffered, unprotected, high-impedance output that is directly connected to the T100MD-1616+ CPU. This analog output is included to allow you to build a custom D/A interface for applications that requires more than one DAC or cannot be adapted to 20-mA current loop operation.

My T100MD-1616+ came with the 4 \times 20 backlit LCD and a batterybacked 62256 SRAM module. The advantages of the LCD are obvious. Having the battery-backed SRAM enables the T100MD-1616+ I/Os, timers, internal variables, and counters to retain their values at PLC power down. Only PWM data isn't retained in the battery-backed SRAM. Another feature of the battery-backed SRAM is the inclusion of a PLC realtime clock that's nonvolatile.

FORMING THE DUET

The T100MD-888+ and T100MD-1616+ super PLCs both contain RS-232 and RS-485 communications capability. Each PLC depends on a host PC to get their control signals out onto a LAN, WAN, or the Internet. PLC COMM1 is configured as a DCE RS-232 interface. This allows PLC COMM1 to attach directly to the DTE RS-232 interface of the host PC without special crossover cables or null modems. The '1616+ RS-232 interface is simple and uses only three lines, including receive, transmit, and ground. PLC COMM1 also can be attached to a standard modem by crossing the transmit and receive lines.

You can use PLC COMM1 for other RS-232 communications tasks when PLC COMM3 is pressed into service. PLC COMM3 is a half-duplex RS-485 interface. Like PLC COMM1, the PLC COMM3 can be used to send and receive programs and commands to either the host PC or another PLC on the RS-485 network. For instance, PLC COMM1 could service a modem while PLC COMM3 is connected to a num ber of other local PLCs on a RS-485 multi-drop LAN. I didn't use a modem for my Florida-room PLC performance, instead I chose to explore the RS-485 and RS-232 connectivity of the T100MD-888+ and T100MD-1616+ with a persistent Internet connection.

RS-485 will be used to connect the T100MD-888+ and T100MD-1616+ no matter how I configure the PLC LAN, therefore the first order of PLC business entails assigning a unique RS-485 ID to each PLC on the RS-485 multidrop network. Both of my PLCs came out of the box with their IDs set to 01. Using the factory-provided RS-485 driver IC, the RS-485 LAN can consist of a maximum of 32 TM100MD+ nodes (0x00–0x1F) including the PC node if it is used. Other more efficient pincompatible RS-485 drivers can be used to raise the LAN node count to 256. I won't experiment with the RS-485 driver IC upgrade, because I will have only a three-node network at the maximum.

Issuing a host link command from the serial port setup area of the TLServer window reads or sets the RS-485 device ID of the PLC. Host link commands allow the PLC programmer to read and write the internal variables of the PLC. The PLC variables I'm referring to include the physical inputs and outputs, logical relays, timers, counters, memory locations, PWM, ADC, and DAC channels, time of day registers, timer preset values, integer storage, and string storage.

Most of the variables are 16 bits wide. For instance, Output[1] is a 16-bit variable representing physical outputs 1 through 16. Output[2] represents physical outputs 17 through 32, and so forth. The maximum number of inputs is 256 with the same number for outputs. There can be 512 internal relays arranged in blocks of 16 just like the inputs and outputs (Relay[1], Relay[2], etc.) and 64 each of timers and counters again arranged in blocks of 16.

There are also 26 32-bit integers (A through Z) defined in each PLC and the 26 predefined string areas (A\$ through Z\$) can hold up to 70 characters each

including the trailing null character. Counters and timers come in at 128 each. Add all of that to 4000 16bit user memory locations. TBASIC can use the results of host link commands executed against these variables, making the TBASIC/ host link command duet a powerful PLC programming tool. In fact, the TBASIC instruction set includes a TBASIC command, called NETCMD\$, to handle issuing host link commands. In addition to



Photo 1—What you see here is a fully loaded T100MD-1616+. The analog area (unlabeled blue screw terminals) is directly left of the red-tagged CPU module. The SRAM battery/clock module is sandwiched between the SRAM module and IC socket.

the syntax you would expect in a BASIC language, TBASIC contains specialized commands to operate against the PWM, ADC, and DAC modules.

The easiest way to perform the ID assignment task is to use one of the PC's standard serial ports and the PLC's COMM1 port and manually issue the host link command to change the ID. I must admit I spent an hour or so trying to do this the first time. The host link command to read an ID is IR* plus the Enter key or Send Command button. The correct response from the PLC would be IR01*, with the *01* being the currently assigned ID. Well, the IR* command worked perfectly.

The ID assignment command is IWXX*, where XX is the ID number you wish to impose on the PLC. I couldn't get that simple command to work to save my life. After consulting the T100MD-1616+ documentation, I discovered that by placing "@01" before the command, I could direct the command directly to ID 01. That worked for the IR* command but I still couldn't change the PLC ID to 00 with IW00* or @01IW00*. More reading put me on the trail of adding "00" between the command and the delimiting "*". In went @01IW0000* and bingo! It seems the final 00 tells the command interpreter to ignore the frame check sequence (FCS) and do the operation no matter what. I moved on and set the T100MD-888+ ID to 00 and the T100MD-1616+ ID to 01. The PLC ID assignment windows for both PLCs are shown in Photo 2.

The PLC RS-485 LAN will reside on the Internet at 216.53.172.209:9080. The PC server at this IP address has no



Photo 2—In both cases, the starting ID for each PLC is 01. The host link command, *IWXX*, is executed with trailing zeros to force the operation. The COMM1 port is automatically opened when the Send Command button is clicked.



Photo 3—This arrangement has lots of possibilities. By networking the two PLCs, I now have twice as many system resources to work with. I can either set up a master/slave relationship with the PLCs and share the slave resources or make each PLC an independent entity.

internal RS-485 adapter card. So, I'll use one of the server PC's serial ports to connect the PC server to PLC COMM1 of the PLC, which is connected through PLC COMM3 to the RS-485 LAN. The next order of business is to assemble the RS-485 and PLC power cables.

In the first part of this duet of PLC articles, I wondered why the RS-485 connector was soldered in instead of removable like the rest of the interfaces on the T100MD-888+ and T100MD-1616+ PLCs. According to Leon at Triangle Research, the answer is simple. In the heat of PLC battle, it is too easy to confuse the power terminal with the RS-485 terminal when both use the removable terminal housings. So, to keep from messing up the RS-485 driver IC when the terminals got swapped, the engineers at Triangle Research decided to not allow the user to remove the RS-485 terminal block. This little inconvenience would per-



Photo 5—In the upper window, you can see the 26 integer system variables (A through Z), high-speed counters (HSC1-3), analog and PWM channels, and what is displayed on the LCD if it is attached. While debugging, you can actually edit the variables from this window. The bottom window is animated and allows you to control the PLC it is logically connected to.

haps prevent any accidental swapping of the RS-485 and power cables. As my reputation sometimes precedes me, I asked Leon if he did that only on the PLCs I have in my possession. Thankfully, the answer was, "No."

Moving on, I used standard Cat 5 cable to assemble my half-duplex, three-wire RS-485 cable. The PLC power cable was constructed with 0.156 header pins and 24 AWG hookup wire. I also used the 0.156 header pins on the RS-485 cable because that makes it easy to insert and extract the cables without damaging the ends of the cables. I added a $120-\Omega$ terminating resistor across the T100MD-1616+ RS-485 connector. Photo 3 shows the complete PLC RS-485 LAN.

After knocking a hole in my router to allow the Internet TRiLOGI server's 9080 TCP/IP port to be seen on the big wire, I swung over to another PC tied to the persistent Internet connection I call Road Runner and opened a TRiLOGI window. Using the default administrator login, I connected to the TLServer at 216.53.172.209:9080 as expected. If I did everything right in the PLC setup, clicking on the Detect ID button should have hooked me up to the T100MD-888+ PLC whose ID is 00. So I went ahead with the click, and....

00 appeared in the box beside the Detect ID button when I was in the T100MD-888+ via an Internet connection. I wrote a simple ladder program

> in Part 1 that sequenced through the PLC outputs 1 through 4 blinking the LEDs as it went. I decided that was as good a test as any and proceeded to upload the program to the now Internet-connected T100MD-888+. Less than a minute later, I was flashing LEDs on ID 00. You can see the sequence of events to get to this point in Photo 4.

> The next logical step is to contact the T100MD-1616+, which is identified as 01 on the RS-485 LAN. I entered "01" in the window next to the Detect ID button in the "Select PLC with ID#" window. Click....

But there was no response. Click again.... Still there was no response. After some thought, it occurred to me that the TLServer could not directly reach the T100MD-1616+ PLC at ID 01 because the TLServer was not directly participating in the RS-485 LAN. A few minutes later, the light got even brighter. My simple blinker ladder program does not incorporate any programming to force communications over the RS-485 link between the T100MD-888+ and T100MD-1616+. So, anything destined for the PLCs that are downstream of the PLC directly attached to the PC must go through the directly attached PLC initially. Also, in this case, a master/slave rela-



Photo 4—The sequence of events that leads to getting a ladder program across the big pond to the PLC begins with connecting to the remote server and selecting the target PLC by its ID. The ladder program is then compiled and sent down the line.

tionship must exist on the RS-485 LAN, because there is no master RS-485 adapter in the PC.

Here's where NETCMD\$ goes to work. The NETCMD\$ command in TBASIC directs a host link command out of a selected PLC COMM port. In addition, the NETCMD\$ instruction automatically calculates the FCS and appends any necessary command termination characters. So, by using TBASIC and NETCMD\$, you can access all of the parameters of all of the PLCs attached to an RS-485 multi-drop LAN. Although that's a good thing and I can manipulate other PLC resources programmatically, the downside to the configuration I have now is that I can only use TRiLOGI to download ladder programs to the directly attached PLC. I would then have to write a ladder program to access the resources of any other PLCs on the

LAN. Even worse, if I wanted to run different programs on each PLC, I would not be able to load or reload the programs remotely without fancy coding.

It was time to change my tune. I pulled out another Windows-based PC I keep in reserve and stripped it down to the minimal configuration. I added an Ethernet adapter card and an RS-485 adapter card from B&B Electronics. The motherboard of my PC provided the necessary serial ports as well as a video interface. I decided to use the PC's standard COM1 port as the PLC serial interface, and configured the RS-485 adapter card to provide RS-485 services on COM3 using IRQ5. Next, I loaded a full complement of Internet TRiLOGI V.5.1 on the new PLC server PC. I also reset the PLC IDs to 01 for the T100MD-888+ and 02 for the T100MD-1616+ using the COM1-to-PLC COMM1 RS-232 serial connection.

The RS-485 adapter card interfaces to the RS-485 PLC LAN via a standard RJ-45 connector. The PLC RS-485 setup is configured as a half-duplex, two-wire LAN. The RS-485 adapter card assumes a four-wire, fullduplex RS-485 LAN. Those aren't problems; I used another piece of Cat 5 twisted-pair cable and tied the A and B TX/RX pairs together (TXA to RXA, TXB to RXB) to form a two-wire interface at the other end of the adapter interface cable.

Earlier, I called the half-duplex implementation a three-wire interface. The third wire in the half-duplex, two-wire interface is the common ground connection between the PLCs and the RS-485 adapter card, which comes from pin 8 of the RJ-45 interface. I carefully matched the RS-485 A lines to the PLC's RS-485 "-" lines and the RS-485 B lines of the adapter to the PLC's RS-485 interface "+" lines, and made the common ground connection at the T100MD-888+ power interface.

At this point, I purposely left the T100MD-1616+ out of the LAN just in case I made a mistake. I adjusted the new IP settings of the TLServer to match the new IP address of the PC and



Photo 6—The selected location is S2, the Florida room. In the upper right window, one read action (A1) is defined for PLC 00. The lower right window is a detailed view of action A1. The result is the Excel window, which shows LED4 (0x08 in cell B1) was active at the day and time shown in cell A1.

activate the RS-485 COM port, plugged the card into the PLC RS-485 LAN, and applied power to the T100MD-888+.

The result was both positive (no smoke) and negative (no data). After hours of troubleshooting, I managed to correct some minor errors, like not having a switch thrown on the PLC to tell it that PLC COMM 3 was in charge. I accidentally discovered that the RJ-45 connector on the card had a problem,



SERIAL MINI-TERMINAL



RS232 terminal for Stamp, PC, z80, AVR etc. -super low current, powers from serial line -LED backlit LCD, visible in all conditions -9600-115.2k-667k, DB9, simple commands -specify 20 customizable or 16 tactile keys eval(1) \$75, oem(1k) \$21.30, w/basic cpu \$27

eval kit SINGLE CHIP COMPUTER \$ oem (1k) \$1.99 ZERO external components **NEW! 8K SUPER CHIP**

-Built-in BASIC / Assembly RS232 program download -1K flash,64ee,3irq,2timers -15 I/O bits, A/D comparator -20mips,faster than pic/8051 -20 pin DIP part #MV1200



40x the BASIC program space! -32 i/o, 12 irq, 3 timers, Bus -8K flash, 512ee, 512nvram -Watchdog with internal osc. -40 pin DIP part #MV8515 oem(1k) \$7.10 eval(1) \$25.00

reset version oem S21.30, eval kit S75

Reboots PC on hardware/software hang

3 versions: RESET, TIMER, PHONE

PC WATCHDOG CARD



LO COST MINI-PC

Includes DOS,NV mem,ADC,RAM,clock,ISA bus Ports for serial, parallel, LCD, keyboard Program in Turbo C, BASIC, MASM, etc. Complete, no costly development kits required Lowest power lowest cost PC compatible available XT: oem S27 eval S95 AT: oem S55 eval S195

640x480 VGA LCD \$27

Controller for most single/dual scan LCDs Works with lo-res (160x120, 320x240,etc.) Use with PC or SBC, standard VGA BIOS Source code demo shows VGA initialization Adaptable for other CPUs (i.e. Z80, HC11) oem(1k) S27 evalkit(1) S95 w/10"LCD S195





PC SOLID STATE DISK

Replace mechanical drives with faster, more relaible, more secure solid-state. Use FLASH, NVRAM, UV EPROM. Both DIP and PCMCIA versions from 32Kbyte (PCM1) to 1Gigabyt (PCD2) starting at \$14,200em(1k) \$50,00eval(1)

STAMP DR Read / Write PC compatible hard

disk, PCMCIA, & Compact Flash on stamp, 8051, pic, avr, z80, x86 -up to 4 gigabyte capacity -low power, simple commands -RS232 @ 9600, 115.2k, 667k -\$95eval \$27oem, IDE version\$14

NEW!! CREDIT CARD COMPUTER 2

- 4meg FLASH - 12 bit ADC - MV8515 CPU



MVS PLUG-N-GOtm no cables/power supply to buy! Low-power RISC cpu 10x faster than PIC,z80,8051 4m flash, ser, par, RTCC, 4ch 12bit adc, ISA/104 bus Built-in BASIC/assembly, other compilers available Friendly instructions, RS232 download (DOS.WIN) Eval kit (1) \$75, oem (1k) \$21, CC computer I \$50

WWW.STAR.NET/PEOPLE/~MVS

MVS Box 803 Nash.,NH 03060 (508) 792 9507



5yr Limited Warranty Free Shipping Mon-Fri 10-6 EST

SERVING THE EMBEDDED **COMMUNITY SINCE 1979!**

but I still could not get a consistently good data transfer or TLServer connection to the new RS-485 LAN. Things would work for a moment and then go totally berserk for no apparent reason.

Before I went with a total RS-485 LAN, I experimented with using the NETCMD\$ command to manipulate ports and memory on the slave T100MD-1616+ with a master program running on the T100MD-888+. I decided that this might be my problem because the card is now the master on the LAN and not the T100MD-888+.

I was doing some basic stuff like reading and writing Input[1] and Output[1] of the T100MD-1616+ from code in the T100MD-888+. Here's a typical line of code:

Z\$ = NETCMD\$(3, "@01WVS02015555")

The ASCII result of the NETCMD\$ operation is returned in the variable Z. The 3 points to PLC COMM3 as the port to use. The Write System Variable host link command in my example begins with a pointer to the ID of the target PLC, @01. The 02 is a type parameter that correlates with the Output[n] variable. 01 is the system variable index denoted by the n in the brackets and translates to Output[1]. The 16-bit pattern "5555" is written to Output[1] with the NETCMD\$ command adding the FCS and any other necessary characters to the command stream.

To read the system variable Input[1] on the slave PLC, I issued the following host link command:

Z\$ = NETCMD\$(3,"@01RVS0101")

The significant differences in this line of code are that the first 01 points to the Input[n] system variable and the second 01 is the index that substitutes a one for n. Thus, the system variable to be read is Input[1]. In both cases, input and output, an ASCII string similar to the host link command is returned. You must parse and convert the returned ASCII data.

I noticed that from time to time, I would get the 01010101 LED sequence I generated on the slave outputs on the T100MD-888+. I also produced windows asking for upload passwords that I had not enabled. The masteracting-like-the-slave LED display and the sudden password protected files clued me that maybe the firmware of the T100MD-888+ was toasted. So, I figured I would totally erase the T100MD-888+ and see if that would fix anything. After an unsuccessful attempt to find a nuke command in TRiLOGI, I opened a TRiLOGI window with nothing in it and sent the "nothing in it" ladder program to the confused T100MD-888+ using the directly attached PC serial port and the PLC COMM1. That method worked. I could now see the T100MD-888+ across the Internet as ID 01 and the funny LED pattern had left the building. I loaded my sequencer program into the T100MD-888+ remotely via an Internet connection and the LEDs went to work as designed.

Will I be able to see the T100MD-1616+ on the RS-485 LAN as ID 02? Learning from my previous mistakes, I made sure the DIP switch setting on the T100MD-1616+ card was as I used it as a slave in my NETCMD\$ experiments. I was certain it would not speak on its own when it should be keeping its interface shut. After putting the T100MD-1616+ physically on the new RS-485 LAN, I once again remotely connected to the new TLServer machine and selected 02 as my desired target ID. I was in there without a problem and I proceeded to upload the same sequencer program into the T100MD-1616+ at ID 02, which was now running on the T100MD-888+ at ID 01.

I am now able to independently control, program, and monitor either PLC over the Internet. By turning on the online monitoring feature of TRiLOGI, I can pause, stop, and start a PLC application and interrogate its internal variables without writing more code (see Photo 5).

RECORDING THE MUSIC

Now that everyone is in tune, harmonies from the PLC duet are wafting across the Internet. The T100MD-888+ and T100MD-1616+ not only control things, but are also capable of monitoring devices. There are times when someone wants to know just how things are with a PLC in the form of a report or spreadsheet. Normally, that means more work for the poor PLC programmer, because you must resort to writing scripts or programs to capture the desired data. However, that's not so here.

When all of your PLCs are humming, TRi-ExcelLink allows you to monitor and collect data from up to eight PLC sites into Excel spreadsheets (see Photo 6). Basically, Tri-ExcelLink uses read and write actions to gather data from the multitude of PLC variables I mentioned earlier. For instance, you could read Output[1], which is the full complement of 16 outputs on the T100MD-1616+, and then write the 16-bit value representing Output[1] to memory location 2000 (DM[2000]). Then, you could set an action in Tri-ExcelLink to put the contents of memory location 2000 into a time stamped cell of an Excel spreadsheet.

PLC OR SBC

Although the T100MD-888+ and T100MD-1616+ are called PLCs, in reality they're more like SBCs. The capability of computing as well as controlling outside your everyday view sets T100MD+ devices into the SBC realm. The strong Java-based TRiLOGI software made putting the PLCs on the Internet a snap. PLC or SBC, the '888+ and '1616+ aren't complicated, they're embedded.

Fred Eady 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. Fred may be reached at fred@edtp.com.

SOURCES

3PXCC4A RS-485 Adapter card B&B Electronics Manufacturing Co. (815) 433-5100 www.bb-elec.com

T100MD-1616+/888+

Triangle Research International 877-689-3245 www.tri-plc.com

ABOVE THE GROUND PLANE

Ed Nisley

Switches and Glitches

Although most of the switch-related problems you run into aren't of epic proportions, there are a lot of little things that can go wrong. This month, Ed sorts through a variety of problems related to switches and provides plenty of insight for your next electronics project. ssentially every electronic device includes at least one switch, if only to select an option or start an action. Mechanical switches seem to be a solved problem, just pick one from a catalog and get on with the more complicated decisions. Sometimes, however, it's the simple things that get you. Mechanical switches not only control current flow, but also generate electromagnetic noise while producing timing irregularities. Let's take a closer look at some switches and their glitches.

A SIMPLE SWITCH

Suppose you put a switch and a 1-k Ω resistor in series with a 6-V supply, then connect an oscilloscope across the switch. What happens when you press the switch? Well, if you're like me, nothing much, because you haven't properly set up the oscilloscope triggering. After you leap that hurdle, you'll probably see something like the top screen in Figure 1.

Instead of the clean edge you might expect, the switch I used requires about 60 ms to settle down. The transitions range from a few microseconds to tens of milliseconds, with no particular pace or rhythm. For high-current, low-speed applications like switching a lamp, this poses no problem. But if you're sampling the switch with a microcontroller, you could easily come up with the wrong answer, repeatedly in fact.

If you trawl the 'Net, you'll see schematics with capacitors across switches to suppress transients. That doesn't work at all, as shown by the lower screen in Figure 1, where I bridged the switch with a 100-nF bypass capacitor.

Two things should stand out clearly. First, switch bounce varies dramatically between actuations, with this press being complete in about 35 ms. Second, the capacitor has no effect on the larger bounces and not much on the smaller ones; it does not round off the transitions at all. That's because the switch shorts the cap when it closes, which, by definition, produces an abrupt transition regardless of the size of the capacitor. When the switch opens, the cap does slow the voltage rise time, but the time constant of a small capacitor and a small resistor is generally smaller than the width of the bounce, so you get a full voltage transition anyway.

Notice, too, that most of the low-tohigh transitions now have large, positive-going spikes reaching well above the supply voltage. How can adding a capacitor cause spikes? It turns out



Figure 1—Adding a capacitor to a switch doesn't reduce bounce. Instead, stray inductance combines with capacitance to form a tank circuit that produces sharp spikes after each interruption.



Figure 2—An optical interrupter provides a smooth transition, but increasing its spatial resolution dramatically reduces the available current. Note the different horizontal and vertical scales.

that the capacitor forms a series-resonant circuit with the stray inductance in the wires connecting the power supply, switch, and resistor. When the switch is closed, about 6 mA flows through the wires and resistors. When the switch opens, the energy stored in those inductances forces the current through the only available path: the oscilloscope's 1-M Ω input impedance.

In theory, 6 mA through 1 M Ω should produce a 6-kV spike. In practice, it's much less than that, for the simple reason that there isn't much energy stored in the stray inductors and the current drops rapidly. Nevertheless, pay attention if your circuit puts a switch at an unprotected FET gate; those spikes can punch through gate oxide like a hot knife through butter.

You'll also notice a few low-going glitches on the trace in both captures. Those are from the same effect in reverse: when the switch closes, it shorts the oscilloscope's input capacitance to ground and a single data sample might catch the small spike. It's a tiny effect, but still visible. Remember that the oscilloscope sees the switch through a meter of cable, so there's room for different voltages at either end, at least for one or two nanoseconds.



Figure 4—A relay without transient suppression is an invitation to disaster, but adding those components has other effects on the circuit.

You've probably seen designs that add a large capacitor, tens or hundreds of microfarads, across a switch. That's a terrible idea, because the switch contact must carry the transient current resulting from a dead short across a live capacitor. If you assume the switch has 10 m Ω of contact resistance, the initial current from a 6-V capacitor is 600 A and the total energy in a 100-µF capacitor is 180 mJ. That can be enough to either fuse the contacts of a small switch closed or dramatically reduce its life expectancy.

You can, however, exploit the energy in a moderately large capacitor to burn through the film that inevitably collects on switch contacts. By keeping the energy stored in the capacitor well within the switch contact ratings, you can ensure the switch makes a solid, albeit bouncy, connection every time. Smaller capacitors may be needed for other reasons. For example, in previous columns I've used small capacitors across switches to prevent RF from entering circuits. At those frequencies, small caps can be beautiful.

THE OPTICAL ALTERNATIVE

An optical interrupter can be a nice alternative to a mechanical switch, because it operates without switching transients and with electrical isolation from the switch actuator. Basically, an optical interrupter is a U-shaped housing with a phototransistor peering across the U at an IR emitter. The transistor is on when it can see the emitter and off when an object blocks its view.

Photo 1 shows the breadboard I used to collect the data for Figure 2. A micrometer positioning slide moves a razor blade through the gap of an optical interrupter. I drove the IR LED at 17 mA and recorded the transistor current from a 6-V supply using a digital VOM.

The left graph in Figure 2 shows that the interrupter switches from fully off to fully on as the blade moves about 2.5 mm, with most of the transition occurring over the central 1 mm. Although the curve is much nicer than Figure 1, you can see that this isn't a digital device either.

Placing a pinhole aperture in front of the phototransistor improves the spatial resolution and reduces the total signal. The graph on the right in Figure 2 shows how a 0.5-mm pinhole affects the response. In most cases, you'd order an optical interrupter with a specific aperture rather than install one yourself.



Figure 3—The phototransistor can pick up external illumination, such as this signal from a fluorescent lamp appearing across a 4.7- $k\Omega$ resistor. Fluorescent lamps have a 120-Hz flicker that appears as an asymmetric 60-Hz waveform.

A typical application would use an analog comparator to produce a digital logic output. A simple comparator with a single threshold won't work correctly, however, because the phototransistor responds to illumination from both the IR LED and ambient light, as shown in Figure 3. It's easy to fall into this trap. The phototransistor produces an output current, not an output voltage, so you must convert from current to voltage. If you have 3 mA available, a 4.7-k Ω resistor ensures that the output saturates at 6 V with a third of that current.

The 4.7-k Ω resistor, connected from the phototransistor to ground, generated a 250-mV_{PP} signal when I positioned a fluorescent lamp about 6" from the optical interrupter. That 50- μ A modulation was visible for any blade location that didn't saturate the transistor. A single-threshold comparator will produce 120-Hz chatter as the blade moves into the interrupter. The solution requires enough hysteresis to ensure that the output switches once and remains switched until the input changes by more than the possible variation caused by ambient light. You determine those values for each installation based on the phototransistor sensitivity, ambient light intensity, and mechanical speed of the blade.

Pop Quiz: Will a light shield suffice to eliminate ambient light? Hint: Low-pressure sodium lamps put out a tremendous amount of energy at 589 nm (the sodium D-line), well within a typical phototransistor's response curve, with modulation at nearly all harmonics of 60 Hz. Extra Credit: Will a pinhole eliminate enough extraneous light to reduce the problem? Hint: Consider your circuit's saturation levels. So, you may need not only hysteresis, but a major low-pass or comb filter as well. Suddenly, that simple optical switch looks a lot more complex, doesn't it?



Figure 5—The lower trace in each screen shot shows the logic level driving the transistor switch. The upper trace shows the 12-V supply. Without suppression, the relay causes 40-V supply transients. A capacitor or Schottky diode dramatically reduces the problem. Note the different scales.

RELAY MADNESS

I suspect relays have caused more digital designer headaches than all other circuit components combined. Everybody has a horror story about how a relay messed up their logic. Discovering why requires a trip through the analog domain.

Figure 4 shows a simple circuit using a 12-V DIP DPDT relay with a 700- Ω coil that draws 17 mA at 12 V. Any small NPN transistor will suffice for the driver (I used a 2N4401), and a pulse generator set for a 2-Hz square wave provides repetitive switching.

The top screen in Figure 5 shows the magnitude of the problem with no suppression components. The top trace reveals 40-V_{PP} transients on the 12-V power supply that also couple into the base drive circuit appearing in the lower trace. Further probing revealed a 1.4-MHz, 80-V_{PP} oscillation at the transistor collector. Fairly obviously, transients of this magnitude would disturb nearly any circuitry using the same

Parts List Software

for Engineers and Designers

- Easily create and manage multi-level parts lists for products in development...and after.
- Track sources for items with multiple price breaks.
- · Calculate product costs at any quantity.
- · Launch CAD, viewer or browser from any Item.
- Automatically generate RFQ's or PO's.

Version 4.0

- New Data Security.
- New Simple Rev Control.
 New Dymo/Costar® and
- Seiko® label support. • New Winfax® support
- for RFQs and POs.



Visit www.trilogydesign.com and download our FREE DEMO.



n For Windows 95/9 MO. 3 Editions, starti \$99



Or, Call 800-280-5176 530-273-1985 Fax 530-477-9106 P.O. Box 2270, Grass Valley, CA 95945



power supply. Imagine the results if you drove a 5-V version of this relay directly from the logic supply.

Adding a 100-nF capacitor across the relay coil produced the middle screen in Figure 5. Note the change in both time and voltage scales. Compared with the previous example, this is a small and well-controlled transient caused by the current circulating through the capacitor and coil. The initial slope of the transient shows the capacitor charging from a constant-current source. Plugging those measurements into the familiar capacitor equation indicates that the initial current is about 20 mA, which is close enough to the actual 17 mA to inspire confidence:

 $\frac{\Delta v}{\Delta t} = \frac{1}{C}i$

The energy driving the current comes from the magnetic field stored in the armature and core of the relay. As that field collapses, the voltage across the inductor remains relatively constant:



Photo 1—*A* razor blade mounted on a micrometer slide cuts off the light path between the IR LED and photo-transistor of the optical interrupter. The micrometer barrel shows the slide position in 0.02-mm increments.

$$\frac{\Delta i}{\Delta t} = \frac{1}{L}v$$

However, as the transistor shuts off, the current forces an increasing voltage across its collector-emitter junction.

A small resistor in series with the capacitor increases the power dissipation and reduces the duration of the transient. That RC circuit, called a snubber, should be tucked close to the relay coil terminals to reduce stray inductance that would cancel out the effect of the capacitance. Replacing the capacitor with a Schottky diode produces the transient in the lower screen of Figure 5. The voltage remains low until the core emerges from saturation, then jumps with the coil current spike.

Homework: Build a similar circuit, measure the voltage and current, and then compare the energy stored in the coil when power is off with the amount dissipated in the various resistances. Although relays aren't generally used in circuits that demand precise timing, adding transient suppression to the coil can dramatically increase the relay's release time. Figure 6 shows what happens to the circuit in Figure 4.

In the top trace, the lack of suppression should be obvious in the burst of noise just after the input goes low. About 660 µs later, the common contact comes off the normally closed contact and the channel two voltage begins dropping. After 500 µs, the flying common contact hits the normally closed contact and begins bouncing for 500 µs. Roughly 1.5 ms after the input goes low, the relay finally stabilizes.



CIRCUIT CELLAR®



Figure 6—Suppressing the coil transient lengthens the relay's release time. Trace 2 shows the normally closed contact bouncing as the relay armature comes to rest.

Recall that a Schottky diode suppresses the coil transient by diverting current from the transistor that is turned off back through the coil. Because the coil can't tell the difference between current through the transistor and recirculating current (electrons being identical, after all), the relay remains active until the current drops below the release point. The lower screen in Figure 6 shows this effect, as the NO contact now opens at 1.3 ms and the NC contact first closes at 4.4 ms. Removing the transient doubles the release time and increases the "no contacts closed" duration. Homework: The snubber resistor I mentioned earlier dissipates energy quicker. Measure the effect of various RC combinations on release time.

The graceful exponential voltage decay appearing in channels two and three comes from the oscilloscope probe's input capacitance (15 pF) discharging through its input resistance (10 M Ω). The time constant in the figure is 200 µs, which indicates an additional 5 pF of stray capacitance in the wiring. Close enough?

CONTACT RELEASE

For switch debouncers, check out the MAX6816-18 from Maxim. If you insist on building a switch debouncer yourself, the series of pages starting at www.play-hookey.com/digital/rs_ nand_latch.html should get you started. Note that debouncing a switch with firmware is a nontrivial proposition. It's easy to get it almost right, but you must pay attention to limiting cases. Watch out for infinite loops and razor-fine glitches!

You should invest a few hours breadboarding some circuits and pondering your oscilloscope if you've never seen these effects up close. Believe me, you'll have a much better understanding of why switches and relays can scramble your firmware.

Speaking of relays, the 9th Annual Trinity College Home Fire-Fighting Robot Contest takes place in April. You may check the schedule and details at www.trincoll.edu/events/ robot/. I'll see you there!

Ed Nisley, PE, is an electrical engineer and a ham radio geek (call sign KE4ZNU). You may contact him at ed.nisley@ieee.org.

RESOURCE

Maxim Integrated Products, "Switch Bounce and Other Dirty Little Secrets," A153, September 2000.



Network Enable Any Serial Device 10/100 Ethernet for \$99*



www.netburner.com



THE FASTEST WAY TO ENABLE YOUR NETWORK DEVICE

*Quantity 100

FROM THE BENCH

Jeff Bachiochi

You Too Can Design with SoC

A Design Challenge 2002 Primer



With cash prizes on the line in the PSoC Design

Challenge, Jeff details working in the world of SoC. Anyone new to designing with SoC parts will walk away with enough insight to implement an SoC in their next project. educing the number of components in a circuit has many benefits. The most obvious yield is inventory and assembly cost savings. Circuitry requires varying amounts of both analog and digital components. For the most part, analog circuitry requires the majority of components. So, improvements in analog devices will have a major effect on the overall component count.

You've seen the micro grow to include internal RAM and reprogrammable ROM. It was a natural progression for these micros to begin including digital peripherals as well, like UARTs, counters, timers, and PWMs. The core processor now has a ton of models with various permutations. You've no doubt used one of those white blobs to make a prototype. You stick all of your components into an array of small contact buses and then interconnect them using small jumper wires. Imagine if manufacturers gave you the ability to use this technique within a single chip!

Application-specific integrated circuit (ASIC) manufacturers have been filling a niche for a long time. Mixedsignal ASICs reduce your parts count by offering a way to include all of your circuitry on the same chip. However, the outcome is specific to your particular design and no changes can be made without producing a new ASIC.

Cypress Microsystems changed the rules when it introduced a family of programmable System-on-a-Chip (SoC) microcontrollers. The CY8C25/26xxx combines a fast core, RAM data memory, reprogrammable code memory, and digital and analog blocks onto the single chip. These blocks can act like different peripherals each time you reconfigure them. Configuration of all of the blocks is mapped into the register space of the core (see Figure 1).

PSoC

The Harvard architecture of this 8-bit core provides faster overall throughput because it has separate address and data buses. Although the processor can be clocked up to 24 MHz, slower clocks have the advantage of consuming less current. The instruction set has more than 130 instructions based on variants of less than 30, including bit manipulations. All devices have access to both analog and digital blocks from the smallest 8-pin device to the largest 48-pin device.

Each block can be used alone for simple functions or in combination with other blocks to produce higher level functions. An 8 × 8 hardware multiply and 32-bit accumulate module (MAC) presents results in a single instruction cycle. Every I/O pin can serve as an interrupt source and has highly configurable output drive specifications.

The main oscillator achieves ±2.5% accuracy without the use of a crystal. If higher accuracy is necessary, a 32-kHz clock crystal provides both low-speed oscillator accuracy and a PLL reference for the high-speed oscillator. The low-speed oscillator, which can run without a crystal, provides additional clocking for the watchdog/sleep timer and the PSoC blocks.

The flash memory has an endurance of more than 100,000 erase/write cycles and uses a flexible protection scheme to prevent not only unwanted writes but also unauthorized reads. Let's take a brief look at some important points of the basic architecture before delving into the analog and digital blocks.

CORE

The primary operation of the M8C core is controlled through six primary registers, which are not directly accessible. Although these registers are not mapped into either of the two register banks, they affect or are effected by various instructions either directly or indirectly.

Figure 2 shows the bit breakdown of the six CPU registers (CPU_F, CPU_A, CPU_X, CPU_SP, and CPU_PCH/PCL). Note: Although it isn't shown in Figure 2, the upper three reserved bits will be used for RAM bank switching on devices that support more than 256 bytes of RAM.

The instruction set is broken down into five areas, program flow, nondestructive tests, arithmetic, movement, and logical manipulations. There are 10 possible addressing modes for instructions. Four modes affect the source register: Immediate, Direct, Indexed, and Indirect Post Increment. The source is the constant value during Immediate mode. RAM or the register location's value is the source during Direct mode. For Indexed mode, an offset is added to RAM or a register's location and the new location's value is the source. For Indirect Post Increment mode, a pointer stored at a

Address	Interrupt priority number	Description
0x0004	1	Supply monitor interrupt vector
0x0008	2	DBA 00 PSoC block interrupt vector
0x000C	3	DBA 01 PSoC block interrupt vector
0x0010	4	DBA 02 PSoC block interrupt vector
0x0014	5	DBA 03 PSoC block interrupt vector
0x0018	6	DBA 04 PSoC block interrupt vector
0x001C	7	DBA 05 PSoC block interrupt vector
0x0020	8	DBA 06 PSoC block interrupt vector
0x0024	9	DBA 07 PSoC block interrupt vector
0x0028	10	Acolumn 0 interrupt vector
0x002C	11	Acolumn 1 interrupt vector
0x0030	12	Acolumn 2 interrupt vector
0x0034	13	Acolumn 3 interrupt vector
0x0038	14	GPIO interrupt vector
0x003C	15	Sleep timer interrupt vector
0x0040		On-chip program memory starts here

Table 1—Interrupt vectors occupy the bottom of the flash memory space.

RAM location points to the source value; the pointer is incremented after the instruction is executed.

Three modes, Direct, Indexed, Indirect Post Increment, affect the destination register. Three additional modes affect both the source and destination and are combinations of the previous modes: Destination Direct Source Immediate, Destination Indexed Source Immediate, and Destination Direct Source Direct.

RAM, flash memory, and register banks serve as three separate memory spaces for the CY8C2xxx. The internal flash memory is a linear array, beginning with interrupt vectors at location



Figure 1—The CY8C2xxx microcontroller goes way beyond the basic core with on-chip RAM and flash memory. The general-purpose I/O interfaces internally to programmable interconnections for both analog and digital blocks.

0x0000. User memory begins at 0x0040 and ends with the available memory up to 16 KB.

The internal RAM memory is 128 or 256 KB. The hardware stack builds up from low memory at 0x00 to a user-defined top of stack (TOS). General-purpose RAM extends from the TOS to the end of available RAM.

The register memory is divided into two register banks—zero and one. The bank is selected using XIO, bit 4 of the CPU_F non-addressable register. All peripheral registers, including those associated with the programmable analog and digital blocks, have a location set aside

in at least one of the two register banks.

I/0

Refer to Figure 3 for the I/O port register configurations. The state of an output bit is set when output data is written to the data register. The logic state of an input pin is read through the same data register.

The interrupt enable register allows every bit to be enabled as an interrupt source. The global select register allows the I/O pin to be routed to one of the global buses for use with the analog and digital blocks. The next pair of drive mode registers designates the pin configuration.

The state (logic 0 or 1) of each pin can be configured as High-Z, harddriven, or resistive pull-up/down. Finally, the last pair of interrupt control registers configures any enabled interrupt pin for rising, falling, or change-of-state interrupt operation.

CLOCKING

The main internal oscillator is sensitive to voltage because a crystal doesn't control it (unless you add an external crystal). The main oscillator is factory-trimmed (8 bits) for 5 VDC. This value is stored in the trim register and can be modified if a $V_{\rm CC}$ other than 5 VDC is used. An additional low-speed oscillator has a similar factory-set, 6-bit trim register allows the low-speed trim register allows the low-speed oscillator to be turned off. If a 32-kHz crystal is connected as an

external source for the low-speed oscillator, the crystal also can serve as a phase locked loop (PLL) reference for the main oscillator, thus improving the $\pm 2.5\%$ accuracy of the internally trimmed oscillator.

As a base of operations, all available clocks are derived from one of these two sources (see Figure 4). So, it makes sense that these two clocks are available from the start. The SLP clock is derived from the 32-kHz clock and runs from 512 Hz down to 1 Hz. The 24V1 clock has a 4-bit divider from the 24M main oscillator, which allows it to run from 24 MHz down to 1.5 MHz. The 24V2 clock has a 4-bit divider from the 24V1 divider, which allows it to run from 24 MHz down to 93.7 kHz (based on both 24V1 and 24V2 values).

The CPU clock has an 8-bit divider derived from the 24-MHz main oscillator and can divide the source down to 93.7 kHz. The digital blocks can also produce ACLK0 and ACLK1 clock signals to be used by the analog blocks. These two clocks along with the 24V1 and 24V2 clocks are available to each of the four Acolumnx analog block multiplexers. These select one of the clocks and allow it to be further divided by a 4-bit divider for clocking any of the analog blocks (more on this a little later).

CPU FEATURES

CY8C2xxx devices have several hardware features. The two's compliment MAC provides immediate results. Writing 8-bit signed values to both MUL_X and MUL_Y will cause a multiply to occur and the result available as a 16-bit signed word in MUL_DH and MUL_DY. A multiply and accumulate begins when either the MAC_X or MAC_Y register is written to. The 32-bit signed double word result is available in registers ACC_DR0 through ACC_DR3.

Although the analog blocks include high-speed comparators, without a data decimator these comparators would be little more than 1-bit A/D converters. The decimator allows a high-speed 1-bit datastream to be converted to lower speed multiple bit data. When used in conjunction with the other PSoC blocks, this function can provide an *n*-bit ADC.

There are two types of reset for CY8C2xxx devices. Power-on reset (POR) takes place when V_{CC} rises

Bit	7	6	5	4	3	2	1	0	
POR 0 0 0 0 0 0								0	
Read/write – – – RW R RW RW									
Bit name Reserved Reserved Reserved XIO Super Carry Zero Global IE									
Bit 0—Globa instructions	I IE determine (0 = disable	es whether al d; 1 = enabled	l interrupts a 1)	are enabled	or disabled	I. Must be	set or rese	t with logi	
Bit 1—Zero is arithmetic o	s set by the C peration (0 =	PU to indicat	e whether o zero; 1 = equ	or not there l ual to zero)	nas been a	zero result	in the prev	vious logio	
Bit 2—Carry arithmetic o	is set by the peration (0 =	CPU to indica no carry; 1 =	te whether carry)	or not there	has been a	a carry in th	e previous	logical/	
Bit 3—Super 1 = supervi	indicates wh sor code)	ether the CPI	J is exexutir	ng user code	e or superv	isor code (() = user co	de;	
Bit 4—XIO is set by you to select between the register banks. Must be set or reset with logical instruction. "Set" can also mean to put a logic 1 in this bit (0 = Bank 0: 1 = Bank 1).									
"Set" can al	so mean to p	ut a logic 1 ir	this bit (0 =	= Bank 0; 1	= Bank 1).	t of reset w	iti iogicai	Instruction	
"Set" can al Bit 5—Reser	so mean to p ved	ut a logic 1 ir	this bit (0 =	= Bank 0; 1	= Bank 1).	t of react w	ni i logical	instruction	
"Set" can al Bit 5—Reser Bit 6—Reser	set by you to so mean to p ved ved	ut a logic 1 ir	this bit (0 =	= Bank 0; 1 :	= Bank 1).	t of reset w	in logical	Instruction	
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser	set by you to so mean to p ved ved	out a logic 1 ir	this bit (0 =	= Bank 0; 1	= Bank 1).	i of readit w	in logical		
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser Accumulator re high register	ved ved ved egister (CPU_ (CPH_PCH)	_A), Index req	gister (CPU_ counter lov	_X), Stack p	ointer regis	ster (CPU_S	SP), Progra	am counte	
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser Accumulator re high register Bit	ved ved egister (CPU_ (CPH_PCH)	_A), Index reg	gister (CPU_ o counter lov	_X), Stack p v register (C	ointer regis PU_PCL)	ster (CPU_S	SP), Progra		
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser Accumulator rr high register Bit POR	ved ved ved egister (CPU (CPH_PCH) 7 0	_A), Index reg , and Progarr 6 0	gister (CPU_ counter lov 5 0	X), Stack p v register (C 4 0	ointer regis PU_PCL) 3 0	ster (CPU_9 2 0	SP), Progra	am counte	
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser Accumulator m high register Bit POR Read/write	ved ved ved (CPH_PCH) 7 0 System	A), Index reg , and Progam 6 0 System	pister (CPU_ counter lov 5 0 System	_X), Stack p v register (C 4 0 System	ointer regis PU_PCL) 3 0 System	ster (CPU_s	SP), Progra	am counte	
"Set" can al Bit 5—Reser Bit 6—Reser Bit 7—Reser Accumulator ra high register Bit POR Read/write Bit name	so mean to p ved ved ved (CPH_PCH) 7 0 System Data [7]	_A), Index reg , and Progam 6 0 System Data [6]	gister (CPU_ counter lov 5 0 System Data [5]	_X), Stack p v register (C 4 0 System Data [4]	ointer regis PU_PCL) 3 0 System Data [3]	ster (CPU_s 2 0 System Data [2]	SP), Progra 1 0 System Data [1]	am counte 0 0 System Data [0	

Figure 2—You cannot directly access these system registers.

above the 2.3-VDC threshold. An external reset input (CY8C26xxx) will also force a POR. A POR provides a minimum of 864 μ s for the V_{CC} to complete its rise prior to executing any code at the reset vector. A watchdog reset (WDR, based on SLP) will also force a POR. Register CPU_SCR (0xFF in both register banks) holds status bits showing which function caused the reset.

Sleep mode reduces power consumption in one of two ways. When the stop bit is set in the CPU-SCR register, the main oscillator is halted. All functions associated with this clock or those derived from it cease.

To further reduce current, you can disable the analog block power by clearing the PWR bits in register ARF_CR (0x63 in bank 0). When the CPU is halted, only an interrupt, WDR, or POR will restart the processor. Although the stop bit is reset, allowing the CPU to operate, you must turn on the analog power. The low-speed oscillator continues to run unless you intentionally disable it.

Another hardware feature is an onboard Switch mode pump that creates a temporary working voltage higher than the rising $V_{\rm CC}$ to allow the supply voltage monitor (SVM) circuitry to operate properly. One of eight programmable low-voltage trip levels can then initiate a POR if the $V_{\rm CC}$ is lost or reduced for some reason. An internal band-gap reference source is used for the SVM and as an analog reference. Because the band-gap reference is sensitive to voltage, a trim register is provided to adjust compensations when a $V_{\rm CC}$ other than 5 VDC is used.

The CPU has an on-board supervisor ROM to manage flash memory programming, erasure, and protection issues. The ROM has additional capabilities like reading product IDs and calculating flash memory block checksums. You can access these ROM routines with the system supervisor call instruction SSC. Various functions can be called based on the value in the accumulator (ACC). Certain functions require parameters to be preset into the upper eight RAM locations and any values returned are put in those same locations.

Port data regi	ster 7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	RW	RW	RW	RW	RW	RW	RW	RW		
Bit name	Data [7]	Data [6]	Data [5]	Data [4]	Data [3]	Data [2]	Data [1]	Data [0]		
When writte	n, bits 0–7 a	re for output	on the port	pins. When	read, they	are the stat	e of the port	t pins.		
Port 0 data re Port 1 data re Port 2 data re Port 3 data re Port 4 data re Port 5 data re	egister PRT0L egister PRT1[egister PRT2[egister PRT3[egister PRT4[egister PRT5[DR is at addr DR is at addr	ess bank 0, 00 ess bank 0, 04 ess bank 0, 04 ess bank 0, 04 ess bank 0, 14 ess bank 0, 14	Dh 4h Bh Ch Dh 4h						
Port interrupt	enable regis	ter								
Bit	7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	W	W	W	W Int En [4]	W Int En 10	W	W Int En [1]	W Int En 10		
Port 0 intern Port 1 intern Port 2 intern Port 3 intern Port 4 intern Port 5 intern	upt enable reg upt enable reg upt enable reg upt enable reg upt enable reg upt enable reg	ister PRTUE jister PRT1E jister PRT2E jister PRT3E jister PRT4E	is at address is at address is at address is at address is at address is at address is at address	bank 0, 01n bank 0, 05h bank 0, 09h bank 0, 0Dh bank 0, 11h bank 0, 15h	1					
Bit	7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	W	W	W	W	W	W	W	W		
Bit name	GlobSel [7]	GlobSel [6]	GlobSel [5] G	ilobSel [4]	ilobSel [3]	GlobSel [2]	GlobSel [1]	GlobSel [0		
Port 1 global Port 2 global Port 3 global Port 4 global Port 5 global	select registe select registe select registe select registe select registe	er PRT1GS is er PRT2GS is er PRT3GS is er PRT3GS is er PRT4GS is er PRT5GS is	s at address b s at address b	ank 0, 021 ank 0, 06h ank 0, 0Ah ank 0, 0Eh ank 0, 12h ank 0, 16h						
Port drive mo	de 0 register									
Bit	7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	W	W	W	W	W	W	W	W		
Bit name	DM0 [7]	DM0 [6]	DM0 [5]	DM0 [4]	DM0 [3]	DM0 [2]	DM0 [1]	DM0 [0]		
Port 2, Drive Port 3, Drive Port 4, Drive Port 5, Drive	mode 0, regis mode 0, regis mode 0, regis mode 0, regis	ter PRT3DM ter PRT3DM ter PRT4DM ter PRT5DM	0 is at addres 0 is at addres 0 is at addres 0 is at addres	s bank 0, 09 s bank 0, 0D s bank 0, 11 s bank 0, 15	ו h ו					
Port Drive mo	de 1 register									
Bit	7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	W	W DM4 (0)	W DNM (F)	W	W DIAL (0)	W DNA (0)	W	W DNM (0)		
Bit name	DM1[/]	DM1 [6]	DN1 [5]	DM1 [4]	DM1 [3]	DM1 [2]	DM1[1]	DM1 [0]		
Port 0, Drive r Port 1, Drive r Port 2, Drive r Port 3, Drive r Port 4, Drive r Port 5, Drive r	node 1, regist node 1, regist node 1, regist node 1, regist node 1, regist node 1, regist	ter PRT0DM ter PRT1DM ter PRT2DM ter PRT2DM ter PRT3DM ter PRT4DM	I is at address is at address is at address is at address is at address is at address is at address	s bank 1, 01 s bank 1, 05 s bank 1, 05 s bank 1, 09 s bank 1, 00 s bank 1, 11 s bank 1, 15		1				
-ort interrupt	control 0 reg	ister	-		-					
Bit	/	6	5	4	3	2	1	0		
Read/write	Ŵ	Ŵ	Ŵ	Ŵ	Ŵ	Ŵ	Ŵ	Ŵ		
Bit name	IC0 [7]	IC0 [6]	IC0 [5]	IC0 [4]	IC0 [3]	IC0 [2]	IC0 [1]	IC0 [0]		
The two Inter IC1 [x], IC0 [x IC1 [x], IC0 [x Port 0, Interru Port 0, Interru Port 2, Interru	rupt Control I $[] = 0 \ 0 = Disc] = 1 \ 0 = Risc pt Control 0, r pt Control 0, r$	bits that cont abled (defau ing edge (po register PRT register PRT	rol a particula lt) sitive) DIC0 is at add IIC0 is at add	ar port pin ar IC1 [x], IC0 IC1 [x], IC0 ress bank 1, ress bank 1,	e treated a 0 [x] = 0 1 = 0 [x] = 1 1 = 02h 06h 04b	s a pair and Falling edge Change fror	are decoded (negative) n last direct	as follows read		
Port 2, Interru Port 3, Interru Port 4, Interru Port 5, Interru Port interrupt	pt Control 0, r pt Control 0, r pt Control 0, r pt Control 0, r control 1 reg	egister PRT2 register PRT2 register PRT4 register PRT5	BICO is at add BICO is at add BICO is at add BICO is at add	ress bank 1, ress bank 1, ress bank 1, ress bank 1,	0Eh 12h 16h					
Bit	7	6	5	4	3	2	1	0		
POR	0	0	0	0	0	0	0	0		
Read/write	W	W	W	W	W	W	W	W		
Bit name	IC1[7]	IC1 [6]	IC1 [5]	IC1 [4]	IC1 [3]	IC1 [2]	IC1 [1]	IC1 [0]		
The two Inter IC1 [x], IC0 [x IC1 [x], IC0 [x]	rupt Control I $\langle] = 0 \ 0 = Dis$ $\langle] = 1 \ 0 = Ris$ pt Control 1	bits that contabled (defauting edge (potential)	trol a particula lt) sitive)	ar port pin a IC1 [x], IC0 IC1 [x], IC0 ress back 1	re treated a D[x] = 0 1 = 0 D[x] = 1 1 = 0 D[x] = 1 1 = 0 D[x] = 0	s a pair and Falling edge Change fro	are decodec e (negative) m last direct	l as follows read		
Port 1, Interru Port 2, Interru Port 3, Interru Port 4, Interru Port 5, Interru	pt Control 1, pt Control 1, pt Control 1, pt Control 1, pt Control 1,	register PRT register PRT register PRT register PRT register PRT	1IC1 is at add 2IC1 is at add 3IC1 is at add 4IC1 is at add 5IC1 is at add	ress bank 1, ress bank 1, ress bank 1, ress bank 1, ress bank 1,	07h 0Bh 0Fh 13h 17h					

Figure 3—Up to 44 bits of I/O are supported on CY8C2xxx devices. The flexibility of each I/O bit is visible though the seven registers used to configure each port. If these registers are implemented, port 5 is 4 bits wide.

IALL 🔰	
ELECT	RÖNICS
CORPO	RATION
SHOP OUR WWW.allele	ONLINE STORE ectronics.com
Great Pr Switching	rice! 60 Watt Power Supply
Autec # UPS61-100- Input: 115/230 Vac Outputs: 3.3V @ 4A, 5 V @ 2A, 12V @ 3A, 32V @ 5 New, open-frame sw supply. 5" x 3.46" x "Panasonic-type" two for AC input, can be Fused input. 10 com output. Ferrite bead output wires. Large o CAT # PS-70	4-T 6 mA. vitching 1.68." o conductor receptacle hardwired if desired. hductor molex-type plug for EMI suppression on quantity available.
\$ 4 50 each	case of 48 pieces for \$3.50 each
Piez	zo Alarm
Audible piezo alarm high-pitched tone of loudness. Operates of 0.54" diameter x 0.3. 0.3" centers. CAT#	emits a solid, low to medium on 3-15 Vdc. " PC leads on # SBZ-412 100 for 85¢ each
♥ ∎each	500 for 75¢ each
Cheap (j LED	yet effective) 9 Flasher
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$100 each	yet effective) Flasher bit flasher ever. etting asnap A cell battery holder. Will a set of batteries. Flash 120 flashes per minute. d. CAT# FSH-10 100 for 70¢ each 500 for 60¢ each
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$100 each	yet effective) Flasher bit flasher ever. the flasher ever. the flasher ever. shap A cell battery holder. Will a set of batteries. Flash 120 flashes per minute. d. CAT# FSH-10 100 for 70¢ each 500 for 60¢ each 1000 for 50¢ each
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$100 each	yet effective) Flasher bring the flasher ever. the flasher ever. t
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$1000 \$1000 CHARGE ORDER American Ex TERMS: NO MINIMUM OR 48 continental U.S.A. \$6.00 HI, PR or Canada must pay in CALIFORNIA must included	yet effective) Flasher Discrete Status Shap A cell battery holder. Will a set of batteries. Flash 20 flashes per minute. d. CAT# FSH-10 100 for 70¢ each 500 for 60¢ each 1000 for 50¢ each
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$100 each CHARGE ORDER American Ex TERMS: NO MINIMUM OR 48 continental U.S.A. \$6.00 HI, PR or Canada must pay in CALIFORNIA must included CALL, WRITE FAX or E-MAIL for our FREE 96 Page CATAL OG	yet effective) Flasher inc flasher ever. and flasher ever. and cell battery holder. Will a set of batteries. Flash 20 flashes per minute. d. CAT# FSH-10 100 for 70¢ each 500 for 60¢ each 1000 for 50¢ each
Cheap () LED The simplest electror Great for attention ge displays, party decorations, night time warning signals. A red 5mm LED with flasher chip is attached to a battery that snaps onto a 2 A flash for months on a rate, approximately 1 Batteries not included \$100 each CHARGE ORDER American Ex TERMS: NO MINIMUM OR 48 continental U.S.A. \$6.00 HI, PR or Canada must pay in CALIFORNIA must includ CALL, WRITE FAX or E-MAIL for our FREE 96 Page CATALOG Outside the U.S.A. send \$3.00 postage.	yet effective) Flasher inc flasher ever. thing a cell battery holder. Will a set of batteries. Flash 120 flashes per minute. d. CAT# FSH-10 100 for 70¢ each 500 for 60¢ each 1000 for 50¢ each 1

Function	DR0	R/W	DR1	R/W	DR2	R/W					
Timer	Count	R*	Period value	W	Capture value	RW					
Counter	Count	R*	Period value	W	Compare value	RW					
CRC	Current value/CRC residue	R*	Polynomial mask value	W	Seed value	RW					
PRS	Current value	R*	Polynomial mask value	W	Seed value	RW					
Dead-band	Count	R*	Period value	W	Not used	RW					
UART	Shifter	N/A	TX data register	W	RX data register	R					
SPI	Shifter	N/A	TX data register	W	RX data register	R					
R*—each tin	R*—each time the register is read, its value is written to the DR2 register										

Table 2—For each function selected, the digital block data registers hold function-specific data.

INTERRUPTS

Of the 15 interrupts (excluding reset) in the vector table of a CY8C2xxx device, 12 are associated with the analog and digital blocks (see Table 1). The other three interrupts come from the SMV, sleep timer, and general-purpose I/O pins. Of the 12 PSoC interrupts, four come from the Acolumn x analog block column outputs. Eight interrupts come from the digital PSoC blocks.

Two interrupt mask registers (INT_MSK0/1) allow each of these sources to be enabled and disabled separately. A separate register, INT_VC, holds the highest priority pending interrupt. (Warning: Writing to this register clears all pending interrupts!) The GPIO interrupt vector is an OR of all enabled port bit activated interrupts. This could get confusing when you're figuring out where the interrupt(s) came from.

PSoC BLOCKS

Programmable System-ona-Chip blocks are user-configurable system resources. The PSoC from Cypress includes eight digital and 12 analog PSoC blocks. These are configured using PSoC designer software. Not only will the designer software help in the configuration process, but will also prepare a device datasheet unique to your configuration.

Many of the digital block functions may be included in today's microcontrollers, but the ability to configure the function of each block allows you to include multiples of the same function for those special applications. Including analog functions in a microcontroller is not a new idea, but the CY8C2xxx analog blocks include devices not previously available on a microcontroller. Let's take a closer look at both of these types of PSoC blocks.

DIGITAL BLOCKS

The eight PSoC digital blocks are divided into two types—four are basic and four are communications-type digital blocks. The communications blocks are identical to the basic



Figure 4—The system clocks are derived from the internal main oscillator and the internal/external low-speed oscillator.

blocks except for additional communications features. Each of the digital blocks (00 through 07) has a function, input, and output register.

The function register (DBAxxFN) allows the block to be defined as a timer, counter, cyclical redundancy checker, psuedo-random sequencer, dead-band generator, UART, or SPI module. Some of these functions are capable of being chained for larger than 8-bit operations. The input register (DBAxxIN) defines the path of any data or clock source used by the block. These can come from the system clocks, analog comparator bus, global input and output bus (port pins), or output from one of the other digital blocks. The output register (DBAxxOU) defines the path for any data out of the block. Here the choices are the global input and output buses or the interrupt controller. Digital block outputs also can be used as a clock input source to the analog PSoC blocks.

In addition, each of the digital function blocks has three data registers

and a control/status register. The three data registers (DBAxxDR0-2) have different uses depending on the function chosen in the function register. Table 2 lists the register definitions. The control/ status register (DBAxxCR0) uses only 1 bit for enabling and disabling unless it is configured for one of the two communications functions, UART, or SPI. These functions have additional control/status bits specific to their functions.

The timer function of the digital PSoC block can capture incoming edges and interrupts on an 8-bit compare or be chained into longer counts. The timer can also create periodic interrupts. As a counter function, the block can count tics between edge inputs or, with the addition of a down counter, produce duty-cycleadjustable PWM output.

The dead-band generator produces complementary outputs equal to the input fre-
quency of the input, but with a programmable dead-band time (time when both outputs are off) meant to drive MOSFETs for motor control. Programmable linear feedback shift registers can be used for psuedo-random generator and cyclical redundancy check functions.

UART transmit and receive functions are treated separately. This allows you to conserve resources when bidirectional communications is not necessary. As an SPI master, clock and bidirectional data lines are implemented. If necessary, SS must be implemented using direct port pin control. In SPI slave mode, SCLK, SDAT, and SS are all implemented. Communication interrupts are generated for Transmitter Empty and Receiver Full UART functions and an SPI Transmitter Empty function.

ANALOG

Getting any kind of analog support in a microcontroller is a relatively new idea. Cypress incorporates 12 analog PSoC blocks in CY8C2xxx devices. These analog blocks are continuous-time and switch-capacitive. The four continuous-time blocks contain programmable gain/attenuation circuitry for op-amp and comparator configuration. Three registers are used to configure the continuous-time blocks (ACAxxCR0–2).

The first register sets the programmable resistor divider and how it is internally connected to provide either gain or attenuation around the opamp. The second register routes various signal inputs to the inverting and noninverting inputs of the op-amp and enables or disables output to the analog output bus or comparator bus. The third and final register determines if the control latch is transparent and on which phase the latch occurs. The compensation for the opamp also can be enabled or disabled, which will improve the amplifier (comparator) switching times.

The eight switch-capacitive blocks are divided into four type A and four type B blocks. Type A blocks have three (two switched and one nonswitched) input arrays of binaryweighted capacitors (plus a direct capacitor input normally used in conjunction with a type B block for biquad filter configurations) with a nonswitched feedback capacitor. Type B blocks have two switched input arrays of binary-weighted capacitors, a nonswitched feedback capacitor, and an output array of binary-weighted capacitors for bi-quad configurations.

The three control registers associated with the two switch capacitor blocks are similar and for the sake of brevity I will treat them as identical.



Photo 1—In the selection screen of the device editor, you get to choose preconfigured module types. Each preconfigured module comes complete with its own datasheet.



B+K Precision's new family of Device Programmers offer extensive libraries that include all of the commonly used <u>IC's</u>, as well as many of the new <u>low voltage / low</u> <u>power chips</u>. Updated libraries and complete device list are available on our web site:

www.bkprecision.com

Dependable, fast and reliable programming is now as easy as child's play.

For more information call us toll-free at (800) 462-9832 Ask for your free new full line catalog.



EK PRECISION

1031 Segovia Circle, Placentia, CA 92870 (714) 237-9220 Fax: (714) 237-9214 www.bkprecision.com



Photo 2—Placement of a selected module is the next task. The design editor shows legal placement locations.

The first register (ASA/BxxCR0) defines the first switched input capacitor value and feedback capacitor value. This register also controls the phasing of the capacitor switches. The second register (ASA/BxxCR1) defines the second switched input capacitor value and the input multiplexer selection.

The third control register (ASA/BxxCR2) defines the third nonswitched capacitor value. This serves as the third array input for type A blocks. For type B blocks, this is an array on the output. ASA/BxxCR2 also controls connection to the analog and comparator output buses and the gated switches, which can be used to short input capacitors to ground.

There are a few odd registers that are necessary for control and status of the analog blocks. The analog comparator control register (CMP_CR) allows the state of the comparator bus for each analog column x (0–3). It also defines whether the Acolumnx interrupt inputs are directly from the bus or qualified with the falling edge of phase two for synchronization purposes.

The analog synchronization control register (ASY_CR) supports a way to perform synchronization with the switch capacitor blocks. The SYN-CEN bit will prevent any write to the switch capacitor registers from being operated on until the rising edge of phase one. Also, when the block is used as a successive approximation ADC, this register can limit the number of bits converted to speed up conversion times.

To route analog signals, there are two registers, an input multiplexer (called AMX_IN) and an output buffer (called ABF_CR) register. The input multiplexer determines which pins of port A are used as inputs to the analog PSoC blocks. The output buffer register enable/disables analog column x to its port A output pin. There is also control over the strength of the output drive.

ARF_CR, the analog bias and reference register, controls the amount of bias and reference used for the analog array. It also controls power to all the analog blocks and can be used to reduce power consumption by shutting down all of the analog circuitry. The final register is the modulation control register (AMD_CR). AMD_CR can select a modulation source for analog columns zero and two by applying the modulation source the switched capacitor block.

As far as the possible interconnects go, it would be a waste of space to allow entire flexibility, in other words to allow every possible connection between blocks. The choices are simplified by predetermining which make the most sense. Figures 5 and 6 illustrate the block interconnections.

PSOC DESIGNER

To help get a feel for the capabilities of CY8C2xxx devices, you need to take a look at the design tools available for the CY8C2xxx family of PSoC microcontrollers. The PSoC Designer is a PC-based integrated development environment (IDE) configured to take you from project design through appli-



Photo 3—The device editor's final step is the physical connection of I/O signals to the pins of the device.

cation development including device configuration, compiling, assembling, debugging, and device programming. All of this is handled in three steps. Here's a look at each.

Things start off at a gentle pace with the creation of a project. First, you're asked for a project name and directory. Then, you must make a selection from the parts catalog of the device that you will use for your project. At this point, you must also indicate whether you will be designing the code in C (available for less than \$150) or assembly. The choice of which language to use will determine how the main application code will be generated as you add modules.

Next, you get into the nitty-gritty. On the Config pull-down menu, you'll see the three parts of the device editor. As you can see in Photo 1, the lefthand frame of the selection screen has a list of module types. A click on any of these will display all of the pre-configured modules available using the digital or analog PSoC blocks.

Photos 1 and 2 display various amplifiers. I clicked on the instrumentation amplifier (INSAMP), as you can see in the top frame of Photo 1. The middle frame shows a block diagram of the module chosen and the amount of resources used both by this module and in the total design thus far. The lower frame displays a data sheet of the module. The datasheet also shows the code, which will be added to your project when and if the module is added to the project.

You can choose all of the modules for your project before proceeding or go through the placement of each as they are chosen. Photo 2 shows the INSAMP being placed after choosing the placement screen of the device editor. The hatched area in the large frame highlights the blocks into which the module can be placed.

Alternate placements are cycled through using the Next Allowed Placement command. Remember to pay attention to the possible interconnection because they differ depending on the placement of the module. After you've chosen the position of the module, stick the module down using the Place command.



Figure 5—Analog and digital blocks can be connected via various buses within the PSoC device. Interconnection possibilities have been included for only the most logical paths to decrease complexity.

You will see the global resources in the upper left frame. These can be set from the drop-down menu that appears when the entry at the right of the resource is clicked on. For instance, the 24V1 and 24V2 can be set to values that will be needed by various modules you select (i.e., data rate generator). The configurable parameters of the chosen module are listed in the user module parameter frame. The parameters can be set the same way as the global resources or by clicking on the appropriate module labels in the large placement frame. You will notice that some of the module parameters include the I/O interconnections.

Now, it's time to configure the I/O pins. Choosing Pinout from the Config drop-down menu will display a screen similar to Photo 3. Even though you've previously configured modules with specific I/O pins, you must again configure the interconnection at the I/O pins. Note that both ends of all user connections must be handled separately. Configuring the I/O pins can be done either on the pinout diagram in the center frame or in the pin list in the lower left frame.

One of the modules I placed in this project was a low-pass filter (LPF). Now, design of a switched capacitor filter is not something you can whip off in an afternoon. There are endless decisions to make, such as filter type, clock frequency, capacitor values, and so on. Cypress included filter spreadsheets for LPF and BPF, which simplify the design of a filter and spit out appropriate values to transfer to the configuration parameters.

Finally, the PSoC designer can take your configured design and generate application files for each of the modules you chose. This takes place when you enter

the Generate Application command. But be aware that like most Windows applications, PSoC designer commands can be initiated in a number of ways—shortcut keys, drop-down menu, or icon on a tool bar.

Let's go on to the third step in the PSoC design process and look at the files created by the generator. At this point, you've indicated which PSoC microcontroller you wish to use as well as what modules are necessary to make up your design. You have interconnected the modules and the I/O pins of the device. The Generate



Photo 4—Here's a list of the files created by the Generate Application command for this project.

Ports 0 to 7	
P0.7 P0.6 P0.5 P0.4 P0.3 P0.2 P0.1 P0.2 P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.2 P2.7 P2.6 P2.5 P2.4 P2.3 P2.2 P2.1 P2.2 P3.7 P3.6 P3.5 P3.4 P3.3 P3.2 P3.1 P3.2 P4.7 P4.6 P4.5 P4.4 P4.3 P4.2 P4.1 P4.2 P5.7 P5.6 P5.5 P5.4 P5.3 P5.2 P5.1 P5.2	
Global output 7	
Global output 0 Global input 7	
Global input 0	

Figure 6—The I/O pin connections have many possible paths to the various analog and digital blocks

Application command has created a number of files based on your previous selections. Photo 4 shows a list of the source, library source, and library header files my project generated.

There are two files, boot.asm and main.asm, in the source folder. The former contains the interrupt table (filled with jumps to the interrupt routines needed by your module selections), sleep timer interrupt service routine supported through interrupt 15, initialization of the oscillator register, C support code if needed, and a jump to the main.asm file. This boot.asm file is regenerated from a template file anytime a Generate Application command is executed to assure it reflects any changes to modules. main.asm, on the other hand, contains nothing at this time. This is where your actual application code will go when you get around to writing it. I'll come back to this later on.

Every module placed in your project will produce a module source code file and, if necessary, a module interrupt source code file in the library source folder. The module source code file contains the routines to initialize, enable, disable, and otherwise make use of the module. The module interrupt source code file contains the interrupt service routine necessary for that module, which requires interrupt action.

In support of the source code files, the library header folder contains include files for each module placed in your project and header files used exclusively for C. The include file holds register declarations for use with the code of the associated module.

If you were to look into the boot.asm file, you would see an LCALL to an external routine, known as LoadConfigInit. This routine is in one of the remaining files in the library source folder, PSoCConfig.asm. The PSoCConfig.asm library folder uses the PSoCCConfigTBL.asm

file to configure the PSoC modules based on their configuration for your particular project.

The final two files in the library header folder are ftb_test_one_ GlobalParams.inc and m8c.inc (except the ftb_test_oneAPI.h file that's used exclusively with C). As the name implies, the ftp_test_one_ GlobalParams.inc file holds those global parameters set while you configure the device. The m8c.inc file contains all of the specific CY8C2xxx family system declarations and a set of predefined macros.

That's a whole lot of code and we haven't even written a byte of application yet. Essentially, all of the necessary code to use the placed modules is now ready for use. All you need to do to make use of the modules is place the appropriate calls to initialize the modules and use their functions in your application. Use the main.asm file for this. Because I configured the global and module parameters in the previous step, the boot.asm file automatically initializes those registers and the only task I have to do is initialize the modules. Now that I've covered the groundwork, it's time to move onto the application.

PROJECT APPLICATION

Oops! This application is so simple I forgot an important part. A small signal input of the sensor is amplified and filtered prior to the 8-bit A/D con-

verter. I need a conversion to take place at periodic intervals and the conversion data sent to the UART transmitter. I forgot to place a counter to divide one of the clocks down to the appropriate interval. I will need a 16-bit counter for this and it will create a periodic interrupt upon which the rest of the application code will be executed. Changing, adding, or deleting modules or configurations is not a problem with the PSoC Designer, because all of the files are easily regenerated to keep the project current.

With the project application code finished, use the Build command to assemble the pieces into a single ROM file. Before programming a chip, you may want to debug to find out how accurately you've defined the project's logic.

DEBUG

The development kit from Cypress includes an in-circuit emulator (ICE). Debugging your code takes place in your hardware environment and not just simulated by your PC. A short CAT-5 patch cable connects the ICE to one of many possible pods. Each device size and package type has a pod that makes the physical connection to your PCB. Debugging begins with a solid parallel port connection to the ICE and downloading the project.rom file. The standard debug commands include read and write to the program and data memory and I/O, CPU, and RAM registers. The other commands supported are run, halt, step, set, clear breakpoints, trace, watch, and dynamic event points.

Let's take a closer look at some of these. Trace has three modes of logging. The PC-only mode logs the instructions executed. The PC/registers mode adds all of the CPU registers and the external 8-bit input of the ICE. The PC/time stamp adds a time stamp to each log entry. You can monitor other TTL signals from your PCB with the external inputs of the ICE.

A watch variable can be input as an address or label within RAM or flash memory and be formatted as decimal or hex in the watch window. Dynamic event points (DEP) differ from breakpoints (BP) in two ways. Breakpoints halt execution upon a program counter (PC) match whereas dynamic event points use a match of any or all of the PC, data bus, address bus, instruction type, external logic input signal, x register, accumulator, SP, and F register to trigger an event. A DEP can halt, turn on or off trace monitoring, or set an external trigger. With this flexibility, complex series and parallel monitoring can help find the causes of stack overflow, memory trashing, and out-of-bounds operation.

YOUR TURN

It is amazing the amount of stuff that can be packed into an 8-pin chip these days. Except for the memory size and number of I/Os (depending on package size), all of the CY8C2xxx micros are comparable to eight digital and 12 analog PSoC blocks. The 8-pin versions are less than \$3 in 100-unit quantities and the big guys are less than \$6, making these extremely cost effective in eliminating external circuitry. The availability of helpful low-cost tools that you get with this product, is of high importance to most designers.

One of the advantages of flash memory devices is, of course, reprogramming of the device. Think about this, with a CY8C2xxx microcontroller you have to ability to write code to reprogram your device on the fly. Your application could change from one module arrangement configured for data input into a new module configuration for data output. That's really fitting 10 lb into a 5-lb bag.

Jeff Bachiochi (pronounced BAH-key-AH-key) is an electrical engineer on Circuit Cellar's engineering staff. His background includes product design and manufacturing. He may be reached at jeff.bachiochi@circuitcellar.com.

SOURCES

CY8C25/26xxx Microcontroller Cypress MicroSystems (408) 943-2600 Fax: (408) 943-6841 www.cypressmicro.com

PSoC Design Challenge 2002 site www.circuitcellar.com/psoc2002



text onto an incoming video source or display colored text on a self-generated colored background screen.

OSD-232 \$99.00



JUMP START YOUR USB DESIGNS

New Edition Includes USB 2.0



"This is the best book on the topic. I recommend it highly." Jon Titus, Test & Measurement World

USB Complete

Everything You Need to Develop Custom USB Peripherals Second Edition, by Jan Axelson

Lakeview Research 523 pages ISBN 0-9650819-5-8 \$49.95

www.Lvr.com/usb.htm



Tom Cantrell

Home Is Where the Plug Is



Although working with X10 can be trouble-

some, Tom is still turned on to the idea of an intelligent home control system. He says the HomePlug Standard looks like a promising solution to home control. n principle, I'm a big fan of the "smart" home concept. It's just that practice, not principle, is the problem. As I sit here poking at my PC on a chilly winter morning, I consider, wouldn't it be nice if I could bump up the thermostat a tad without having to trudge up the stairs?

When the hot summer days return, I'll be thinking similar thoughts about the sprinklers. If I want to give the yard an extra drink, it's out to the garage, squeeze past the cars, and try to remember just what buttons to push on the sprinkler controller. Or how many times have you driven off on vacation and as soon as you're on the freeway you're wondering if the coffee pot was left on or if the front door was locked? Yes, I can hear the dryer buzzer when the clothes are dry. Indeed, it's so annoyingly loud that everybody in the house can. Too bad it's the middle of the night and we're trying to get some sleep.

The list goes on and on. Frankly, in this era when everything from greeting cards to kitty boxes has a micro in it, life at home remains surprisingly low tech. The problem isn't so much with the individual components. The furnace, sprinkler controller, and even the coffee maker and dryer already have a microcontroller or two. The problem is connecting them.

HEX-10

Like many of you, I've dabbled with X10 over the years, even going so far as to connect SBCs and PCs via a power line TW523 two-way X10 interface. However, long ago I decided there was no way X10 would ever be able to serve as the true backbone for a smart home, at least not mine. I still catch flap from the wife for the time I upgraded our outdoor lights with X10 switches.

The setup was flaky from the start in just about every respect. Some outlets could talk to certain outlets but not others. Lights might not respond to commands issued or might respond to commands not issued. Ultimately, even the manual switching ability fizzled out and my wife gave me the "If technology is so grand" lecture, so X10 went bye-bye.

Yeah, maybe if you buy better stuff, condition your AC line, bridge your phases, and pray a lot, X10 might be marginally OK for convenience items, but in my experience it's just too



Figure 1—Thanks to proper planning and proven technology, the HomePlug Alliance has met an aggressive schedule.

unstable for much else. My rule is that I will not connect anything to X10 that I wouldn't mind being on when it's supposed to be off or vice versa. That list is short, and definitely doesn't include things like furnaces, sprinklers, or coffee makers.

NETWORKING OPTIONS

Obviously, it would be wonderful to run Ethernet cable to every room in the house.

Unfortunately, that is not, and never will be, an option for most folks. As much as fancy structured wiring schemes have been talked about (remember CEBUS?), as far as I can tell even builders of brand new homes generally haven't bitten the bullet.

Wireless Ethernet (a.k.a., 802.11) is an attractive option. But, without first-hand experience, I'd have to say compatibility, interference, and range are areas of possible concern. When it comes to the 2.4-GHz spectrum, I'm reminded of Yogi Berra's famous saying about a popular restaurant, "Nobody goes there anymore because it's too crowded."

The phone line is a wired alternative that should not be overlooked. Modern DSL-like networking technologies such as Home PNA allow simultaneous use of the line for both voice and data. And, modern homes have many more phone jacks than their predecessors.

But, look around your own place of residence and I think you'll agree that phone jacks still aren't nearly as pervasive as AC power outlets. Without doing a census, I'd say latter outnumber the former by 10 to 1 in my own abode. There are AC outlets in practically every room in the house, including bathrooms, hallways, the garage, and outdoors. This brings us full circle, except this time we're not stuck in the infinite X10 loop. Say hello to HomePlug (see Photo 1).

STEALTH STANDARD

I was surfing along the web one day when I stumbled across HomePlug. Unlike many of the myriad of standards efforts that crowd the headlines,



Figure 2—Orthogonal Frequency Division Multiplexing (OFDM) divides available bandwidth into multiple low-rate subcarriers to minimize the effects of multi-path (echo) interference. With the HomePlug, 83 subcarriers are spaced 195.3125 kHz apart in the spectrum from 4.5 to 20.7 MHz.

the HomePlug Powerline Alliance (HPA) has been working diligently to get out the spec rather than just issuing press releases.

As a result, in less than two years the HomePlug Powerline Alliance managed to go from a blank sheet of paper to products on the shelf (see Figure 1). The technology was created in early 2000, followed by the first products to the market in late 2001. Then, the certification program was established in 2002. That's impressive by any measure and particularly so in comparison to the slower and more tortured journey of bigger ticket standards like 802.11, 1394, and Bluetooth.

HPA was wise to start at the top with a comprehensive marketing requirements document (MRD) that focused on what needed to be done, rather than launching into and getting hamstrung by the details of how to do it. Nimble response is no doubt a by-product of the fact that the 90-member HPA is of a more manageable size than other groups, which have hundreds or even thousands of members to placate. In any case, the alliance was able to agree quickly and quietly on a specification based on technology from Intellon. Notably absent, at least publicly, were the competitive corporate posturing, positioning, and poli-

tics that seem to be par for the standards course these days.

HPA was able to take advantage of the fact that Intellon already had a lot of experience and proven know-how in the powerline networking field. Thus, the HPA not only quickly agreed on the spec, but also even found time to validate it in field trials covering nearly 500 homes (10,000 wiring paths) in the U.S. and Canada.

OBSESSED FANGIRLS OF DUO MAXWELL?

No, that wasn't exactly what I was looking for when I punched "OFDM" into a search engine. Fortunately, I did find a number of informative links about Orthogonal Frequency Division Multiplexing. You may have heard of OFDM before. It's been around since the '60s, but has really taken off in recent years. [1] It's used in a variety



Figure 3—The Intellon INT5130 is one of the first HomePlug-capable ICs to emerge and is an easy retrofit for existing Ethernet-based designs.

of communication schemes, everything from digital TV to DSL to wireless LANs (e.g., 802.11a).

How does OFDM work? In essence, it's like going from a fast serial to a slow parallel

cable. What does that mean? To understand the concept, consider the example of going from a single-wire at 1 Mbps to 10 wires each at 100 Kbps.



Figure 4—Thanks to the built-in MII interface, the INT5130 is a straightforward drop-in for any system that incorporates an 802.3 MAC.

The total bandwidth is the same, so why in the world would you do that? Well, you'd do it if the problems associated with getting the higher speed

On Time RTOS-32 On Time's royalty-free embedded RTOS for 32-bit x86 implements a Windows NT subset kernel in only 16k.

On Time RTOS-32 has a scalable component architecture. Only the components actually needed by the application are loaded on the target.

Application								
RTKernel-32 RTFiles-32 RTIP-32 RTPEG-32								
RTTarget-32								
Target Hardware								

On Time RTOS-32 Features:

- Source and binary compatibility with Windows 95/98/NT/2000/XP
- Supports DLLs
- Unmatched real-time performance
- ► Context switch 0.73µs on P120
- Supports (but does not require) PC compatible hardware
- Boots from disk, BIOS extension, ROM, or DOS
- Windows 95 look-and-feel GUI
- ► Free tech support
- ► Full source code available
- Supports Microsoft, Borland, Delphi Compilers
- Fully integrates with MS Visual Studio 6/.NET
- Free Evaluation Kit available at http://www.on-time.com
- No run-time royalties

North America: On Time + 15 Spur Lane + Centereach, NY 11720 + USA Phone (BBB) 667-8200 • Fax (631) 471-8850 • email info@on-time.com International: On Time + Holwey 49 + 22085 Hamburg • Germany Phone + 49-40-2279405 • Fax + 49-40-2279263 • email info@on-time.de

On Time RTOS-32 Components:

RTTarget-32 Development System Boot Code, Locator, Cross Debugging, Win32 Emulation Library Price: \$1500

RTKernel-32

Real-Time Multitasking Scheduler Hard real-time, unlimited number of tasks, 64 priorities, 1µs time resolution Price: \$1500

RTFiles-32 Embedded FAT File System FAT-12, FAT-16, FAT-32 For floppy disks, IDE hard and flash disks, M-Systems DiskOnChip, linear flash Price: \$1500

RTIP-32 Embedded TCP/IP Stack BOOTP, RARP, ARP, ICMP, UDP, TCP, DNS Add-Ons: PPP, FTP, TFTP, Web Server, SMTP/POP3, TELNET, SNMP, DHCP Price: \$3000

RTPEG-32 Embedded GUI Windows 95 look-and-feel Graphical User Interface Price: \$2500



link working were far worse than implementing more, but slower, channels.

And, it turns out that's just the situation for wireless and, as you might imagine, AC power-line communication. As the data rate rises, symbols

become smaller and move closer together. Throw in some multi-path echo and you get inter-symbol interference, turning your signal to mush and making it harder and harder to discriminate where the ones and zeros start and end. By contrast, each of the slower links has plenty of space between symbols, allowing echo to fade and the signal to solidify between more leisurely samples.

There's one minor problem with the concept though. Unfortunately, there's only one universe for RF to propagate in, and only one piece of Romex going to an AC socket. That means, in this example, you have to mix the 10 pieces of bandwidth onto one signal, and then take them apart at the other end.

The orthogonal part of the equation comes from spacing each equal sized chunk of subcarrier at a fixed offset from the next. The secret is that proper spacing centers the peak power of each subcarrier with the minimum power, or null, of the next (see Figure 2).

Conceptually simple, at least simpler than blowing through the multi-path interference stop sign, OFDM isn't easy to implement. Heavy duty number crunching (FFT and IFFT) is called in to substitute for the impractical alternative of generating each subcarrier individually, but that's just the start.

The fact is it isn't likely that each subcarrier will deliver the same level of service under all circumstances. The amplitude and phase response varies for each frequency and the particular topology of the connections being made (i.e., which socket is connected to which socket). And we're still talking about a laboratory grade experiment. In the real world, there's noise on the powerlines generated by motors, switching power supplies, fluorescent lamps, dimmer switches, and even RF interference from the Ham radio operator next-door.

To make the best of the situation, HomePlug dynamically adapts to channel quality in three different ways. A particular subcarrier is temporarily dropped if it's completely blitzed. Less severe degradation is accommodated by adjusting the amount of data per symbol (for example, between 1 and 2 bits using DBPSK and DQPSK coding, respectively) and varying the amount of bandwidth devoted to forward error correction. As a result, HomePlug delivers up to 14 Mbps across the powerline, although as for most adaptive schemes, that's no doubt a "downhill with a tailwind" (i.e., bestcase) specification.

POWER MICROPROCESSOR

Nobody ever said it would be easy, but the march of silicon means it is possible. There's no better place to start than the Intellon chipset that was, unsurprisingly, one of the first chipsets to hit the street. As in a traditional modem, the two chips comprise the INT5130 data pump (see Figure 3) and INT1000 analog front end that jointly bridge most of the gap between ones and zeros (3.3- or 5-V tolerant) and 120 VAC.

From the designer's perspective, the '5130 offers two possible interfaces. The first interface mimics a typical Ethernet PHY by virtue of its Media Independent Interface (MII), which is defined as part of the IEEE 802.3u standard. For more information, check out the MII in Photo 2. The MII starts with separate 4-bit transmit and receive data buses. The PHY, in this case the '5130 subsystem, sources both the transmit and receive clocks at 25 MHz. These are derived from the 100-MHz primary clock of the '5130, which requires 25-ppm accuracy. Note this requires the use of an overtone crystal and associated components to guarantee reliable startup at the proper (fifth overtone) frequency.

The MII transmit and receive data and clocks are supplemented with various control signals that indicate errors (RX_ERR, TX_ER), valid data (RXDV), carrier sense (CRS), collisions (COL), etc. MII is good news for designs that easily (or already) incor-



Photo 1—*Promises are one thing, but real products* such as this Intellon-based router from Linksys prove HomePlug is for real. porate a standard 802.3 MAC. That explains why commercial deployment of HomePlug by net box suppliers like Linksys and NetGear is quick and painless. Adding HomePlug to their product lines is practically as simple as dropping the INT5130 plus the INT1000 into all of their existing Ethernet-based designs (see Figure 4).

However, it would be a shame to limit HomePlug to hosts with the horsepower to handle a full-blown, high-speed MII bus. Note that while



the INT5100 does contain internal buffering for it's own use, it acts like a non-buffered device as far as the host is concerned. After a packet is started, it's 25 MHz or bust.

Fortunately, Intellon added an alternative general-purpose serial interface (GPSI) that cuts the pin count in half (from 16 to eight) and the speed by even more (25 to 10 MHz). That's within the range of reason for the high-performance clocked serial interfaces found on newer MCUs.

Whether the MII or GPSI is Interface used for the data path, 802.3u also defines a separate Management Data Interface (MDI) that provides access to INT5130 internal status and control registers. Again, Intellon wisely offers a more streamlined SPI option, likely the more appropriate choice for MCU-based designs. There's also the option of loading the registers at powerup automatically from an external EEPROM via yet another SPI interface.

	crobat Reader-[Embedded File] Ele Edit Document Tools View W	indow <u>H</u> elp			_ 0 ×
0	83 . H . H	4 ▶ N ♦ ♥ ♥ Q • E • S			
Thurthraits Y Bookmarks 0 🗗 🕎	Converse Tox Mere Converse Tox Meree Converse Tox Meree	Prove Hee ↓ → H ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓			PHY
	AUI = ATTACHMI MDI = MEDIUM I MII = MEDIA INC GMII = GIGABIT MAU = MEDIUM	ENT UNIT INTERFACE DEPENDENT INTERFACE DEPENDENT INTERFACE MEDIA INDEPENDENT INTERFACE ATTACHMENT UNIT	PLS = PHYSICAL LAYER SIGNAL PCS = PHYSICAL CODING SUBL PMA = PHYSICAL MEDIUM ATE PHY = PHYSICAL LAYER DEVICE PMD = PHYSICAL MEDIUM DEPE	ING AYER CHMENT E ENDENT	
		in 🛛 🕂 🗰 📂 💆		100	>

Photo 2—The IEEE 802.3u specification defines the Media Independent Interface (MII) between the MAC (media access) and PHY (physical) layers.

On the analog side, the interface to the INT1000 is a 10-bit bus passing the transmit (DAC) and receive (ADC) data at 50 MHz. The INT5130 also outputs 4 or (optionally) 8 bits of automatic gain control (AGC) information to the receive front-end amp. This allows the INT5130 to dynamically adjust the gain from off (i.e., mute the receiver during transmit) to 48 dB in order to maintain the optimum level at the INT1000 analog input.

Finally, the INT5130 includes the link, collision, and activity outputs defined in the MII spec. Typically these are used to drive LEDs that let you know what's going on, a general design practice I strongly encourage.

MAC ACK

As I said, the beauty of the Intellon scheme is that their chipset looks just like a standard PHY to an Ethernet MAC. But inside, the INT5130 has

it's own MAC. It's a veritable MAC attack raising the question of whose MAC is on first?

Getting to this point wasn't easy though. Intellon had to overcome a dilemma. On the one hand, the existing Ethernet MAC leaves a lot to be desired in terms of features (for example, predictable timing, security) and isn't necessarily well-suited to the



Start-of-frame deli (uses all tones	rame delimiter Frame Frame body s all tones) header and pad 25 bits 17 bytes Valuable byte count		y	Check End-of-frame delimiter sequence (uses all tones) 2 bytes 25 bits			Response delimiter (uses all tones)		Contention resolution window		
Preamble Frai	ne Frame rol header	Frame body	PAD	FCS	Prea	mble	Frame control	RIFS	Preamble	Frame control	•••
4 OFDM Variable symbol 20–160 OFDM symbols			count ymbol:	s			4 OFDM symbols			4 OFDM symbols	PRS0 PRS1 CRS0 CRS0 CRS1 CRS1 CRS2
Start of frame indicates:	Pa	yload		End of fram indicates	ne :		Respons indicates	e :	PRS0 a	and PRS1	
Start of frame Contention control Length of frame Tone map index	Start of frame Up to 13.75 Mbps Contention control PHY Rate Length of frame Adapted modulation and tones Tone map index Decoded based on tone map Extensible to higher rates Extensible to higher rates		End Con Cha	End of frame Contention control Channel access priority F		ACK—Good packet NACK—Errors detected FAIL—Receiver busy		11—Highest priority (3) 10—Priority 2 01—Priority 1 00—Lowest priority (0)		(3) 0)	

Figure 5—Under the hood, the INT5130 handles things much differently than a traditional Ethernet, with a completely different packet structure incorporating specific features related to quality of service, security, and the unique characteristics of the powerline as a network medium.

transmission characteristics of the powerline. Obviously there's no way for Intellon to change the existing MAC standards, nor is it expeditious to wait for the new MAC standards efforts underway to come to fruition and migrate into the market.

Instead, they came up with a MAC in the INT5130 that's a better fit and made it impersonate the PHY, a standard Ethernet MAC expects. For instance, in a addition to data, a standard Ethernet MAC transfers frames across the MII that include a lengthy preamble comprised of 56 bits of alternating ones and zeros. But Intellon would rather do it their own way, so they simply strip the header coming from an 802.3 MAC and add it back when sending to an 802.3 MAC.

Meanwhile, the INT5130 MAC does things it's own way under the hood (see Figure 5). For instance, it uses a virtual version of the Ethernet's well-known

CSMA/CA protocol. In Ethernet, a collision is detected immediately by virtue of the transmitter monitoring it's own data transfer. However, the particular way HomePlug works doesn't allow that option because a transmitter will always hear it's own transmission, even if there's a collision that prevents a receiver from hearing it. Instead, HomePlug uses an ACK/NAK handshake protocol to mimic collision detection; a NAK signifies a virtu-

al collision followed by the familiar Ethernet back off and retry. Similarly, the scheme supplements true carrier sense with a virtual version. From the control field, a receiver can identify how long a transmission is expected to last. Should it lose track of the actual signal, it can still deduce a sensed carrier from the elapsed time.

The contention mechanism of Ethernet is fronted with a priority resolution window. In addition, automat-



Measure, Analyze, and Control

Use the RPC-330 for instrumentation and control. Program in C or floating 2 Quadrature/encoder Position Control 8 Channel, 12-bit A/D

2 Channel, 12-bit D/A Fast, simple, and easy programming. Setup and run in 5 minutes!

www.rp3.com Remote Processing Corporation 800.642.9971 Fax 303.690.1875 Denver, Colorado, USA 303.690.1588

CIRCUIT CELLAR®

Lowest Cost PCI Data Acquisition



ADACs new PCI line has uncompromising design features and high quality components at lowest prices anywhere! Plus drivers for Windows NT, 95/98/2000, LabVIEWtm and more. Just check the specs:

Lowest Cost \$199 PCI-5500MF

8 channels 12-bit A/D, 100kHz Sampling, DMA, FIFO, 16 digital I/O, Counter/Timer



High Performance PCI-5501MF

16 channels 12-bit A/D, 100kHz Sampling, DMA, FIFO, Gain of 1.2,4,8 or 1,10,100,1000, 16 digital I/O, Counter/Timer, Triggering, 12-bit Clocked D/A Option

\$499

High Resolution PCI-5503HR

16 channels 16-bit A/D, 200kHz Sampling, DMA, FIFO, Gain of 1,2,4,8 or 1,10,100, 16 digital I/O, Counter/Timer, Triggering, 16-bit Clocked D/A Option

High Performance Data Acquisition

Since 191	-4
voice	800-648-6589
fax	781-721-9888
web	www.adac.com
email	info@adac.com

ADAC

American Data Acquisition Corporation 24 River Street, Winchester, MA 01890 USA ic segmentation is imposed on larger frames to minimize the possible latency a higher priority request might face. The 802.3 MAC doesn't see it, but, as mentioned earlier, behind the scenes, the INT5130 MAC dynamically adapts the channel (useable carriers, coding, and error correction) periodically or in response to changing conditions. Note that channel adaptation occurs independently in both directions, meaning a receiver tells the transmitter what it would prefer. That allows each node to do the best it can rather than having to reduce the entire network to the least common denominator performance of the weakest link.

Oh yeah, don't forget your AC socket is likely connected to more than a few of your neighbors. Despite twisting the House Code dial, for all I know one of my neighbors may have been responsible for some of the ghosts haunting my X10 experience. There's little comfort knowing that I may have introduced a few ghosts in their home as well. I like my neighbors, but sharing my network with them is going a little too far.

HomePlug confines the activity to my logical network defined by a common encryption key feeding a 56-bit DES algorithm using cipher block chaining. Whether or not that's adequate to satisfy privacy advocates is a judgment I'll leave to experts, but it should keep casual or inadvertent eavesdropping at bay.

A NET FOR ALL SEASONS

It's tempting to view the myriad of wired and wireless network standards in a winner-takes-all light. However, of late I'm coming to the conclusion that we're just going to have to juggle all of them for the foreseeable future. Instead of wringing your hands trying to pick a winner, the proper course of action is to identify which networks work best for which applications and put your effort into teaching them to talk to each other.

HomePlug occupies a unique niche as a wired network that doesn't require any new wires. But as such, it's obviously not suitable for wireless applications typically characterized by portability and battery operation. On the other hand, that means it's a good candidate for purely AC-powered devices that never move. The difference between a laptop and desktop PC comes to mind.

The networked smart home of tomorrow will likely include half a dozen or more interconnected network technologies—cable, DSL, 802.3, 802.11, Bluetooth, HomePlug, and yes, even X10. Get over it, and get on with it.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@ circuitcellar.com.

REFERENCE

 R. W. Chang, "Synthesis of bandlimited orthogonal signals for multichannel data transmission," Bell System Technical Journal, vol. 45, December 1966.

RESOURCES

OFDM Resource center Palowireless Pty, Ltd. www.palowireless.com/ofdm

Interactive OFDM Tutorial Andrew C. McCormick University of Edinburgh Department of Electronics and Engineering Edinburgh, Scotland www.ee.ed.ac.uk/~acmc/OFDMTut. html

SOURCES

HomePlug Powerline Alliance (925) 275-6630 Fax: (925) 275-6691 www.homeplug.org

INT5130 IC

Intellon, Inc. (352) 237-7416 Fax: (352) 237-7616 www.intellon.com

Powerline router

Linksys Group Inc. (800) 546-5797 (949) 261-1288 Fax: (949) 261-8868 www.linksys.com





Your 68HC12 Source



Adapt912DT60 Adapt932D160 features Motorola's 112-pin 68HC912D60 running in single-chip mode, with 86 1/0, duil 8-channel 10-bit wertors, 60K on-chip Flash, 2K SRM, 1K EEPROM, R5232, R5463, 12C, 4-chan-nel PWM, CAK, and lots more- all on a lots more- all on a 3,25 in. x 2,25 in. board! Piggy-back RAM board available for rapid development.

Adapt912DT69CDP \$999 Adapt912DG128 same as 'DT60, with dual CAN subsystem and more on-chip memory: 128K Flash, BK RAM, 2K EEPROM. Starter Package: AD912DG1285P \$189	Starter Package includes serial cable, prototyp solderiess breadboard adapter, assembler, Flashloac example code, and documentation. (Requires BDM + AD912DT66SP C Developer Package has Starter Package + 64K I + BDM pod + 1CC12 C Compiler (Windows)	ing card, dutilities, pod) \$149 RAM card
Adapt912DG128 same as 'DT60, with dual CAN subsystem and more on-chip memory: 128K Flash, 8K RAM, 2K EEPROM. Starter Package:	• AD912D160CDP	
• AD912DG1285P	Adapt912DG128 same as 'DT60, with dual CAN su and more on-chip memory: 128K Flash, 8K RAM, 2K Starter Package:	bsystem EEPROM.
	• AD912DG1285P	\$189
C Developer Package:	C Developer Package:	
 AD912DG128CDP	 AD912DG128CDP 	\$539

TECHNOLOGICAL ARTS oll-free 1-877-963-8996 • fax 416-963-9179 www.technologicalarts.com



1/4" CMOS Camera Module. black & white, with digital out-put. Size: only 1.38"x1.14". Lens: f7.4mm, F2.1 FOV 32 deg +S&H \$4.96" Code: CM4881

Lens: 17.4mm, F2.1 FOV 32 deg + S&H \$4.9 x 24 deg, Pixels: 384x288. Scanning: progressive. Video output: digital 8bit. Programmable features. Shutter: 1/50 to 1/5000 sec. Power Supply, SVDC. Standby Current <10mA. Needs regu-lated power supply. <u>*S&H to Canada is</u> \$7.90

Toll Free: 1-888-549-3749 (USA & Canada) Tel: (330) 549-3726. Request a FREE catalog or visi us at: www.electronics123.com for more. Amazon Electronics, Box 21 Columbiana OH 44408





















VISA/MC









\$59

\$69

ADVERTISER'S INDEX

The Advertiser's Index with links to their web sites is located at www.circuitceller.com under the current issue.

Page

ge		Page		Page		Page	
85	Abacom Technologies	83	Decade Engineering	89	Micro Digital Inc	81	SeaFire Micro, Inc.
92	Abia Technology	89	Delcom Engineering	92	microEngineering Labs, Inc.	88	Sealevel Systems Inc.
17	Acroname Inc.	90	Digital Products	42	Micromint Inc.	84,88	Seattle Robotics
85	ActiveWire, Inc.	88	Dreamtech Computers	93	MicroSystems Development, Inc.	90	Senix Corp.
82	ADAC	1	Earth Computer Technologies	87	MJS Consulting	85	Sensory, Inc.
83	Advanced Circuit Designs Inc.	26	ECD (Electronic Controls Design)	58	MVS	83	Signum Systems
21,53	Advanced Transdata Corp.	84	EE Tools (Electronic Engineering Tools)	86	Mylydia Inc.	87	SmartHome.com
11	Advanced Vehicle Technologies, Inc.	62	EMAC, Inc.	90	Nav Masters	51	Softools
37	A K Peters	26	Engineering Express	65	NetBurner	18,63	Solutions Cubed
69	All Electronics Corp.	90	EVBplus.com	95	Netmedia, Inc.	92	Spectrum Engineering
85	Amazon Electronics	84	FDI-Future Designs, Inc.	88	OKW Electronics Inc.	83	Square 1 Electronics
25	American Raisonance Corp.	45	GoHubs, Inc.	78	On Time	57	SUMBOX
9	Amulet Technologies	84	Hagstrom Electronics	93	Ontrak Control Systems	31	Systronix
92	AP Circuits	80	HI-TECH Software,LLC	C2	Parallax, Inc.	C3	Tech Tools
90	Appspec Computer Tech. Corp.	90	HVW Technologies Inc.	83	Phytec America LLC	85	Techniprise Inc.
84	Athena Microsystem Solutions LLC	86	ICE Technology	93	Phyton, Inc.	16,32	Technologic Systems
90	Atlantic Quality Design, Inc.	87	IMAGEcraft	93	Picofab Inc.	85	Technological Arts
71	B+K Precision	86,92	Intec Automation, Inc.	91	Prairie Digital Inc.	87	Tern Inc.
85	Bagotronix, inc.	39	Interactive Image Technologies Ltd.	15	PSoC 2002 Design Challenge	41	Texas Instruments
10,84	Basic Micro	86	Intronics, Inc.	89	Pulsar Inc.	83	Triangle Research Int'l Inc.
79	CadSoft Computer, Inc.	75	Intuitive Circuits, LLC	86	R2 Controls	62	Trilogy Design
91	CCS-Custom Computer Services	24	JED Microprocessors Pty Ltd.	33	R4 Systems Inc.	45	Trinity College Robot Contest
86	Cermetek Microelectronics Inc.	64	JK microsystems	50	Rabbit Semiconductor	89	Vantec
92	Conitec	51	JR Kerr Automation & Engineering	85	R.E. Smith	86	Vetra Systems Corp.
11	Connecticut mircoComputer Inc.	75	LabJack Corp.	81	Remote Processing	84	Virtual Valley Artists
91	Copeland Electronics Inc.	80	Laipac Technology, Inc.	89	RLC Enterprises, Inc.	93	Weeder Technologies
91	Cyberpak Co.	75	Lakeview Research	91	RPA Electronics Design, LLC	87,89,90	Wittig Technologies
47	Cypress MicroSystems	87,93	Lemos International	91	Rutex	91	Xilor Inc.
C4	Dataman Programmers, Inc.	2	Link Instruments	27	Saelig Company	87	Z-World
86	Dataprobe Inc.	91	Lynxmotion, Inc.	88	Scidyne	51	Zagros Robotics
86	DataRescue	7	MaxStream	3	Scott Edwards Electronics Inc.	89	Zanthic Technologies Inc.

Hand Soldering Fine Pitch QFP Devices

Tiny Planet: A Single Bit Wireless I/O Port

RoCK Specifications—Part 2: Circuitry

Working with EMC

Animation System

Using a Plinius Power Line Modem for Home Control

From the Bench: You Are Not Alone

- Silicon Update: I-Way the Hard Way
- Applied PCs: Taking a Swim with Atmel's STK500

Signal Processing

Theme: Embedded Programming

Attention Advertisers:

June Issue Deadlines

Space Close: April 10 Material Due Date: April 17

94 Issue 141 April 2002

CIRCUIT CELLAR®

www.circuitcellar.com



PRIORITY INTERRUPT

Those Insidious Pop-Ups



here are a lot of people who wouldn't care to admit it but the best military-inspired trickle-down technology of the last century was the Internet. The concept was simple and straightforward—a packet management system for speeding communication among engineers and educators. Fortunately or unfortunately, today the Internet has become a central part of our lives. I can't speak for everyone but living without it would be difficult.

Of course, successful inventions tend to evolve in diverse directions. As I sit here writing this editorial, I'm listening to WQXR-FM radio in New York City. I'm receiving it as streaming audio via the Internet. We're only about 100 miles from NYC, which might not seem like much, but I can't receive it directly, no matter how big the antenna. More importantly, with a click of a few keys I can just as easily listen to WPR in Dallas, Texas, or Classic FM in London, England.

Did I also mention that it is 4 a.m., that I have an Asian markets stock ticker in another window, and a live weather map in the upper right corner next to the media player? Well, good or bad, our computerized world has become 24/7. Because there are no national or geographical boundaries on the Internet, it's easy for a compulsive personality to become a workaholic when they deal with 23 other time zones all at once. I'm not like that, I assure you. Aside from all of the simultaneous media watching, the most compulsive I get is checking e-mail immediately after the icon flashes and the tone sounds. I really hate it when it's another Viagra message!

Being an old-timer, I remember the early days of the Internet. I'd get one or two e-mails a day and they would always be business related. When I surfed the Internet, the pages that matched my keywords were G-rated and useful. Now when I enter a key word I get 243,506 matches, and have to think about every low-life interpretation of my keywords or the first 100,000 matches are X-rated.

My single biggest complaint with the Internet these days is all of the subversive advertising activity just to catch eyeball attention. I've come to live with sending 90% of the daily e-mail to the recycling bin, but the one thing that has got me really angry these days is browser pop-ups. These are the little extra browser windows (sometimes there are a whole series of them) containing advertisements that suddenly appear on your screen when you land on certain web sites (or try to exit one). I'm sure they were a brilliant idea from some advertising guy who said, "I can get their attention!" Of course, it's always the guys who don't use the Internet who want to fix it for us.

Surfing the Internet today is like navigating an electronic minefield, especially when you are blindsided at otherwise innocuous URLs. I'm told there are software packages to combat it like a virus, but pop-ups are much more insidious than viruses. We shouldn't have to experience pop-ups, redirected homepages, and cute new URLs added to the Favorites list as the expected penalty for surfing the Internet.

Unfortunately, I don't have an immediate solution either. The Internet is an international medium that seems totally governed by commerce instincts, not law. My aggravation of pop-ups aside, the more undisciplined the site, the more insidious the electronic disasters they perpetrate. I would guess that there isn't much we can do about the sizable portion of the Internet that is dedicated to illegal, immoral, and depraved activities. And, as much as I see the need for some supervision, I don't think that Big Brother watching over our Internet activities or deciding which web pages pass the morality test (whose?) before posting is anything short of outrageous. In fact, when I hear that they actually do that in some countries, I cringe.

If unrestrained commerce and economic adventurism have created the present situation, then the only way to reduce the problem is to demonstrate that it costs more than the benefits it provides. Complaining to commercial sites saturated with X10 pop-ups reminds them about customer relations. Letting host sites know what you think about their advertisers is important.

Unfortunately, I don't see any easy way to rid the Internet of the rest of the crap, especially from the dark fringes. I wish there was a magic bullet, however, I'm afraid it will take something more like carpet-bombing to get substantive results. I know it's sacrilegious to mention service fees in connection with e-mail and search engines, but economic logic may be the only incentive for unfettered surfing and a return to spam-free e-mail. I paid a premium to get a high-speed connection to the Internet. Maybe now I have to pay a fee for the insurance that keeps me from being trashed while I'm on it.

steve.ciarcia@circuitcellar.com



TM

Get all the details and preview the software at: www.tech-tools.com

Copyright @ 2002, TechTools ClearView, CVASM, QuickWriter, the "Wizard" symbol and TechTools are trademarks of TechTools, P.O. Box 462101 Garland, Texas 75046. PICmicro is a trademark of Microchip Technology, Inc. All other trademarks are trademarks or registered trademarks of their respective company.

STILL THE WORLD'S MOST

POWERFUL PORTABLE

PROGRAMMERS?

NEW MODEL

Dataman

......

SURELY SOMEONE SOMEWHERE

SURELY NOT.

MONEY.

INTELLIGENT UNIVERSAL PROGRAMMER

DATAMAN-48LV

CE

\$795 INC 4MB RAM

- Plugs straight into parallel port of PC or laptop
- Programs and verifies at 2, 2.7, 3.3 & 5V
- True no-adaptor programming up to 48 pin DIL devices
- Free universal 44 pin PLCC adaptor
- Built-in world standard PSU for goanywhere programming
- Package adaptors available for TSOP, PSOP, QFP, SOIC and PLCC
- Optional EPROM emulator

DATAMAN S4

- Programs 8 and 16 bit EPROMs, EEPROMs, PEROMs, 5 and 12V FLASH, Boot-Block FLASH, PICs, 8751 microcontrollers and more
- EPROM emulation as standard
- Rechargeable battery power for total portability
- All-in-one price includes emulation leads, AC charger, PC software, spare library ROM, user-friendly manual
- Supplied fully charged and ready to use

Actually, no. But don't take our word for it. Use the feature summary below to see how other manufacturers' products compare.

HAS DEVELOPED A PORTABLE PROGRAMMER THAT HAS EVEN MORE FEATURES, EVEN GREATER FLEXIBILITY AND IS EVEN BETTER VALUE FOR

......

S4 GAL MODULE

- Programs wide range of 20 and 24 pin logic devices from the major GAL vendors
- Supports JEDEC files from all popular compilers

SUPPORT

- 3 year parts and labor warranty
- Windows/DOS software included
- Free technical support for life
- Next day delivery always in stock

Still as unbeatable as ever. Beware of cheap imitations. Beware of false promises. Beware of hidden extras. If you want the best, there's still only one choice - Dataman.

Order via credit card hotline - phone today, use tomorrow.

Alternatively, request more detailed information on these and other market-leading programming solutions.

MONEY-BACK 30 DAY TRIAL

\$1295

If you do not agree that these truly are the most powerful portable programmers you can buy, simply return your Dataman product within 30 days for a full refund



Orders received by 4pm will normally be despatched same day. Order today, get it tomorrow!

VISA



Dataman Programmers Ltd 215 East Michigan Avenue Orange City, FL 32763 Telephone (904) 774-7785 Fax (904) 774-7796 Home page: http://www.dataman.com Email: sales@dataman.com