

# CIRCUIT CELLAR<sup>®</sup> INK<sup>®</sup>

THE COMPUTER APPLICATIONS JOURNAL

#65 DECEMBER 1995

## PROGRAMMABLE DEVICES

In-Circuit-Programmable GALs

Low-Cost FPGA Programmer

Design Contest Winners

In EPC: Novell's NEST



\$3.95 U.S.  
\$4.95 Canada

# EDITOR'S INK

## Start Ups



Although most of us get to participate in the beginning of projects, seldom do we get involved in the start up of a company, association, or—for that matter—publication.

I guess I'm lucky. In 1987, I was asked to add "technical editor" to my engineering duties. With the publication then being bimonthly, I was still able to devote much of my time to engineering. The editing part of my job provided a nice touch of spice to an otherwise engineering-only diet

As the magazine has grown to a monthly publication, my responsibilities have shifted accordingly. In the last week before an issue ships, how I almost long for an engineering-only diet! The engineering is now what keeps me sane.

With the start of *INK's Embedded PC*, Circuit Cellar is opening itself up to another adventure. What started as a quarterly insert in 1995 is already scheduled as a bimonthly insert in 1996. Starting in February's issue, there will be an additional 32 pages devoted entirely to the embedded PC industry.

What changes has this brought in-house? While I stay editor-in-chief of the magazine as a whole, Janice becomes a hybrid of technical editor for *INK* and managing editor of *Embedded PC*. New to the ranks is Carole, joining us as a technical editor, alleviating an overly tight workload and giving room for growth.

The only area we remain a little tight on is the need for *EPC* articles. *Embedded PC* will focus on both PC software and hardware. We'll be covering off-the-shelf motherboards, expansion boards, networking, PCI, other buses, assemblers, compilers, debuggers, multitasking, and operating systems. In other words, assuming your manuscript meets our readership standards, we'll print it. Just send your proposals in.

This issue's *Embedded PC* offers a good mix of topics. Novell introduces their networking expertise to the embedded PC world while Larry Fish shows us how to get the benefits of 32-bit unsegmented architecture under DOS and BIOS. Ken Prada covers PC/104 instruments in oceanography and Russ overviews PC buses.

In the main issue, Stuart Ball takes a close look at PLDs that can be programmed in-circuit, along with some sample applications. David Van den Bout shows us how to build a simple CPLD development system. Finally, Fred Eady overviews the PIC16Cxx family.

For our columns, Ed covers Virtual-86 interrupts from the 32-bit side, Jeff finishes up his two-part article on a carrier current modem, and Tom overviews the conference circuit.

editot@circellar.com

# CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR  
Steve Ciarcia

PUBLISHER  
Daniel Rodrigues

EDITOR-IN-CHIEF  
Ken Davidson

PUBLISHER'S ASSISTANT  
Sue Hodge

EPC MANAGING EDITOR  
Janice Marinelli

CIRCULATION MANAGER  
Rose Mansella

TECHNICAL EDITOR  
Carole Boster

CIRCULATION ASSISTANT  
Barbara Maleski

ENGINEERING STAFF  
Jeff Bachiochi & Ed Nisley

CIRCULATION CONSULTANT  
Gregory Spitzfaden

WEST COAST EDITOR  
Tom Cantrell

BUSINESS MANAGER  
Jeannette Walters

CONTRIBUTING EDITORS  
Rick Lehrbaum  
Russ Reiss

ADVERTISING COORDINATOR  
Dan Gorsky

NEW PRODUCTS EDITOR  
Harv Weiner

ART DIRECTOR  
Lisa Ferry

PRODUCTION STAFF  
John Gorsky  
James Soussounis

CONTRIBUTORS:  
Jon Elson  
Tim McDonough  
Frank Kuechmann  
Pellervo Kaskinen

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to Circuit Cellar INK Subscriptions, P.O. Box 696, Holmes, PA 19043-9613 or call (800) 269-6301.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P O Box 698, Holmes, PA 19043-9613.

Cover photography by Barbara Swenson  
PRINTED IN THE UNITED STATES

For information on authorized reprints of articles, contact Jeannette Walters (860) 8752199.

## HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC  
Barbara Best  
(908) 741-7744  
Fax: (908) 741-6823

SOUTHEAST  
Christa Collins  
(305) 966-3939  
Fax: (305) 985-8457

MIDWEST  
Nanette Traetow  
(708) 357-0010  
Fax: (708) 357-0452

WEST COAST  
Barbara Jones & Shelley Raine  
(714) 540-3554  
Fax: (714) 540-7100

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 6 bits, no parity, 1 stop bit (860) 871-1988; 24001 9600 bps Courier HST, (860) 871-0549, World Wide Web: <http://www.circellar.com/>

All programs and schemata in *Circuit Cellar INK*® have been carefully reviewed to ensure their performance is in accordance with the specifications described, and programs are posted on the Circuit Cellar BBS for electronic transfer by subscribers.

*Circuit Cellar INK*® 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 *INK*® disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK*®.

Entire contents copyright © 1995 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar *INK* is a registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

- 1 2** In-System-Programmable PLDs from Lattice  
*by Stuart Ball*
- 2 0** Building a Low-Cost CPLD Development System  
*by David Van den Bout*
- 2 8** Take Your PIC  
A Look at the PIC 16Cxx Family  
*by Fred Eady*
- 3 6** 7th Annual Circuit Cellar Design Contest Winners  
*by Janice Marinelli*

**7 4** □ **Firmware Furnace**  
Journey to the Protected Land: Behind the Interrupt Curtain  
*Ed Nisley*

**8 4** □ **From the Bench**  
Carrier Current Modem  
Part 2: Alternative Control  
*Jeff Bachiochi*

**9 2** □ **Silicon Update**  
PC Times in Silicon Valley  
*Tom Cantrell*

# EMBEDDED PC

See **pages 39-73** for Our Special Bonus Section

# INSIDE ISSUE 65

- 2** Editor's INK  
Ken Davidson  
Start Ups
- 6** Reader's INK  
Letters to the Editor
- 8** New Product News  
edited by Harv Weiner

**99**  
**ConnectTime**  
Excerpts from  
the Circuit Cellar BBS  
conducted by  
Ken Davidson

**112**  
**Steve's Own INK**  
The Powers that Be

**81**  
Advertiser's Index

# READER'S INK

## A BETTER LEAD

Ed Lansinger's articles on developing an engine control system really interested me, especially since I've gone from having computers as a career and cars as a hobby to the other way around.

I think readers should know about the ignition wires my company imports to Australia from the USA. Magnecor ignition leads are wire-wound leads with 5-20 times more windings than other "heli" leads. They also suppress more RFI and EMI.

While using these wires won't stop your competitors' cars from bringing your ECM unglued due to RFI, the EMI suppression may help when your sensor wiring goes near your plug wires. Contact:

Magnecor Australia Pty. Ltd.  
2000 Oakley Park Rd., Unit 104  
Walled Lake, MI 48390  
(810) 669-6688  
Fax: (8 10) 669-2994

Neil Fisher  
neilf@zeta.org.au

## Contacting Circuit Cellar

We at *Circuit Cellar INK* encourage communication between our readers and our staff, so have made every effort to make contacting us easy. We prefer electronic communications, but feel free to use any of the following:

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.

Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial offices at (860) 875-2199.

Fax: All faxes may be sent to (860) 872-2204.

BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300-14.4k bps, 8N1).

Internet: Letters to the editor may be sent to [editor@circellar.com](mailto:editor@circellar.com). Send new subscription orders, renewals, and address changes to [subscribe@circellar.com](mailto:subscribe@circellar.com). Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article.

For more information, send E-mail to [info@circellar.com](mailto:info@circellar.com).

WWW: Point your browser to <http://www.circellar.com/>.

FTP: Files are available at <ftp://ftp.circellar.com/pub/circellar/>.

The  
only  
8051/52  
BASIC  
compiler  
that is  
100 %  
BASIC 52  
Compatible  
and  
has full  
floating  
point,  
integer,  
byte & bit  
variables.

- Memory mapped variables
  - in-line assembly language option
  - Compile time switch to select 8051/803 1 or 8052/8032 CPUs
  - Compatible with any RAM or ROM memory mapping
  - Runs up to 50 times faster than the MCS BASIC-52 interpreter,
  - Includes Binary Technology's SXA51 cross-assembler & hex file manip.util.
  - Extensive documentation
  - Tutorial included
  - Runs on IBM-PC/XT or compatible
  - Compatible with all 8051 variants
- **BXC51 \$ 295.**

**508- 369- 9556**  
**FAX 508-369-9549**



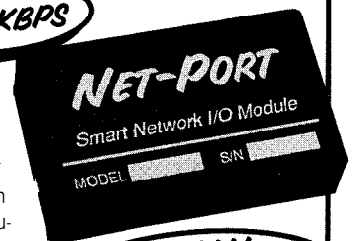
Binary Technology, Inc.  
P.O. Box 541 • Carlisle, MA 0 1741



# NET-PORT™

\$69.00

115 KBPS



Net-Port is a complete serial data acquisition and control system in a 1/2-cubic-inch package. The potted Net-Port contains a variety of digital and analog I/O along with power supply regulation and communication line drivers. Net-Port requires no programming. A simple ASCII command protocol sets and reads all I/O.

## FEATURES

- RS-232A, RS-422, and RS-485 at 300 bps to 115kbps
- Sixteen parallel I/O lines and PC bus
- 4-channel, 8-bit ADC (Net-Port B)
- P-channel, 12-bit ADC and P-channel, 12-bit DAC (Net-Port E)
- PWM output: 2Hz to 3.5 kHz, 5-95% duty cycle
- Simple ASCII command set, requires no programming!
- High-performance, built-in functions: parallel I/O buffering, LCD and keypad control, analog data averaging, data logging
- Sixteen-character ID allows hundreds of Net-Ports
- Small size, encapsulated construction
- Wide power supply input range

**NET-PORT B \$69.00**    **NET-PORT E \$99.00**  
**NET-PORT carrier board w/power supply \$49.00**

Prices do not include shipping  
Features subject to change

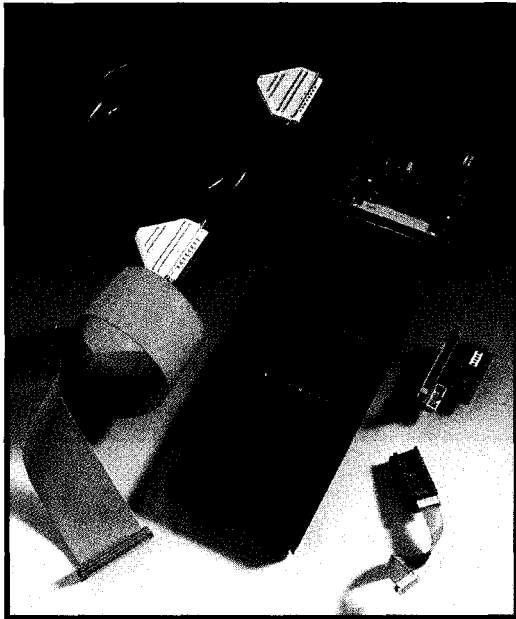


**MICROMINT, INC.**

4 Park Street • Vernon, CT 06066 • (860)871-6170 • Fax (860) 872-2204

# NEW PRODUCT NEWS

Edited by Harv Weiner



## TARGET-CONTROLLER SOFTWARE

Stimgate releases a new productivity tool for developing portable software for embedded microcontroller systems. The **Target Controller for ANSI C** enables software to be written in ANSI C on the PC using well-known tools such as Borland C or Microsoft C. Portability between processor types is ensured by Stimuli-Gateway I/O functions that complete ANSI C with standardized I/O operations for the target microcontroller. By using files and libraries, you can reuse code between different target platforms or C compilers.

The Stimgate Target Controller hardware connects I/O in embedded target processor systems to the PC. It interfaces to the target system by plugging into the EPROM slot and emulates the most popular EPROM, EEPROM, and RAM devices. As with in-circuit emulators, software for different microcontrollers and derivatives of the same processor can be tested without additional personality modules. Prototype software runs out of the target controller memory, which speeds up testing and debugging over traditional program, burn PROM, and edit cycles.

The Stimgate system has built-in hardware stimulation facilities and a library of test functions for test and debugging applications. High-level messages can be sent to the PC to aid in debugging without using the UART. The Stimgate stream windows facilitate message handling from the target system. Once the code has been proven, it can be recompiled and ROMed for the target system using the target microcontroller compiler.

The full development system, which supports embedded controllers such as 8051, 68HC11, 80x86, 683xx, H8/300, H8/300H, and more, sells for \$3950.

CMX Company

5 Grant St., Ste. C • Framingham, MA 01701 • (508) 872-7675 • Fax: (508) 620-6828 • E-mail: [cmx@cmx.com](mailto:cmx@cmx.com)

#500

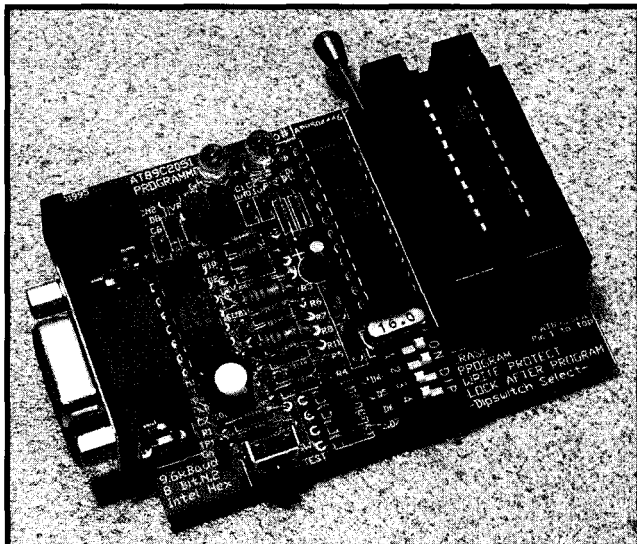
## FLASH MICRO PROGRAMMER

AirBorn Electronics announces a development programmer for the AT89C2051 microprocessor, **Model PG2051**. The AT89C2051 is a 20-pin 8051-compatible microprocessor [including serial port] with 2 KB of flash

memory. The PG2051 erases, programs, and verifies the AT89C2051 chips in 6 s.

The PG2051 may be connected to a PC or other host by a serial cable. According to the settings on its DIP switches, the programmer tests, erases, programs, verifies, and write and security protects as it receives the file. The unit features a test switch which enables the user to check in just one second if the target microprocessor is blank, working, programmed, or failed without needing the PC connected.

The PG2051 Programmer sells for **\$188** and includes data sheet. A complete evaluation kit is available for \$233. It includes the programmer, plug back, two AT89C2051 devices, a small prototype board, sample code, and a shareware assembler and disassembler.



AirBorn Electronics

19-21 Berry St., Ste. 201 • P.O. Box 1491

North Sydney, NSW 2060, Australia

(61) (2) 9925-0325 • Fax: (61) (2) 9925-0297

E-mail: [stevenmQzeta.org.au](mailto:stevenmQzeta.org.au)

#501

# NEW PRODUCT NEWS

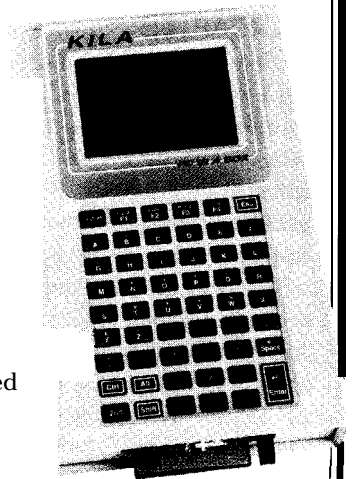
## PC IN A BOX

Kila Systems PC-in-a-Box is a complete DOS system configured as a dedicated controller to run one program for many users. Typical applications are ticketing terminals, POS systems, factory automation, and alarm systems.

PC-in-a-Box is powered by a Chips and Technology F-8680 microprocessor, an AT-equivalent integrated chip with internal peripherals. Its architecture is identical to a PC, except onboard solid-state memory replaces the hard disk. An alphanumeric keypad and graphics LCD provide input and output. DOS applications are developed using high-level compilers and run exactly as they would on a PC. The final code is placed in ROM, flash, or battery-backed RAM. The system runs unattended off an internal rechargeable battery.

The unique features of the PC-in-a-box are an onboard CGA controller and graphic LCD, extensive use of flash and battery-backed SRAM, PCMCIA support, and a customized BIOS for portable applications with extensive power management. Three serial and two parallel ports are also available, and an 8255 chip provides 24 I/O lines. The unit typically requires 100 mA at +5 V and, in suspend mode, power consumption is less than 2 mA.

The PC-in-a-Box is enclosed in a 8.8" x 5.5" x 1.6" custom plastic enclosure. An evaluation kit sells for \$399. The CPU card with 256-KB PSRAM and one serial port sells for \$229 in quantities of 100.



### Kila Systems

2300-C Central Ave. • Boulder, CO 80301 • (303) 444-7737 • Fax: (303) 786-9983

E-mail: kila@rainbow.rmii.com

#502

## SCOPE-TRIGGERING DEVICE

Programmable Designs introduces a family of oscilloscope-triggering devices that provide a wide range of event-triggering options and work with any oscilloscope. With up to four triggering modes, fast input-to-trigger timing, numerous status indicators, and portability, **SuperProbes** are superior to logic analyzers and cost less.

**SuperProbe II** features 18 signal inputs and a clock input. It supports three clock-triggered modes and one combinatorial (pattern match) triggering mode. Flexible logic combinations for specifying trigger events include No-Match triggering for signaling when certain unexpected events occur. The unit has separate "Pattern-Select" and "Don't Care" configuration DIP switches as well as numerous status LEDs for monitoring power, input signals, triggering activity, and clock activity.

SuperProbe II comes with an interface cable that has gold-plated, machined pin contacts designed to fit easily onto IC clips, 0.025" square posts, and the included through-hole or surface-mount component grabbers. SuperProbe II sells for \$649.



**SuperProbe I** is a pattern-match (or word-recognition) triggering device for a logical combination of up to 17 signals. In microprocessor-based systems, this feature enables triggering on a processor write or read operation to a selected memory or I/O device location. SuperProbe I (standard version) sells for \$249 and includes an interface cable with permanently attached grabbers designed primarily for through-hole component leads. SuperProbe I (deluxe) sells for \$395 and includes a SuperProbe II-like interface cable.

**SuperProbe Basic-S** supports pattern-match triggering with up to eight input signals. The Basic-8 includes a cable with permanently attached through-hole grabbers and is capable of meeting the triggering needs of many basic circuit checkout and debugging applications. SuperProbe Basic-8 sells for \$99.

Programmable Designs, Inc.  
41 Enterprise Dr.  
Ann Arbor, MI 48103  
(313) 769-7540  
Fax: (313) 769-7242  
E-mail: design@prog-designs.com

#503

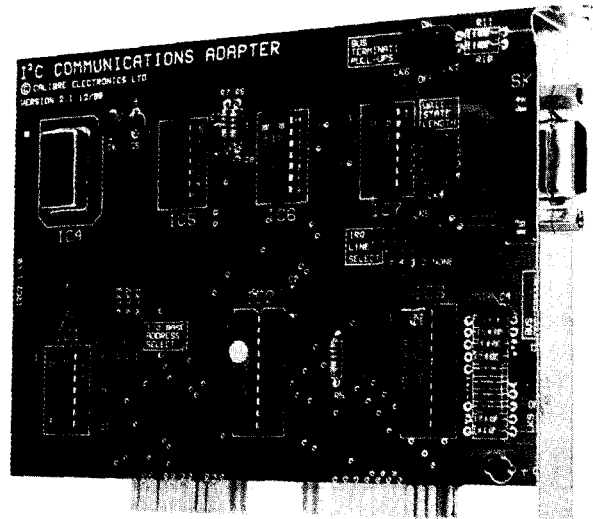
# NEW PRODUCT NEWS

## I<sup>2</sup>C ADAPTER

Saelig is offering a useful tool to save development time for those involved with the I<sup>2</sup>C bus. The I<sup>2</sup>C bus is a two-wire serial bus standard developed by Philips that lets all circuits within a system communicate with each other bidirectionally. It is widely used in television and audio systems and is becoming increasingly common in multiprocessor systems.

Advantages of using I<sup>2</sup>C rather than a parallel architecture include reduced pinout and EMI, simplified wiring and circuit boards, and data rates up to 100 kHz, with communication independent of speed. The bus also offers multimaster capabilities with on-chip collision detection and wire lengths of 12' or more.

The **ICA-90B** kit includes the industry-standard ICA-90 ISA adapter half-card for plugging into your PC, a 3 1/2" disk with an I<sup>2</sup>C function library with many ready-made routines in C and TurboBasic, and a helpful I<sup>2</sup>C instruction manual. All I<sup>2</sup>C functions can be controlled through an adaptable library of routines. The ICA-90B demonstrates I<sup>2</sup>C master/slave modes in receiver and transmitter operations. Both polled and interrupt-driven modes are shown, making further programming for your



application in popular languages easy. It even operates as a transparent I<sup>2</sup>C bus monitor.

The ICA-90B sells for \$299.

The Saelig Company  
1193 Moseley Rd. • Victor, NY 14564  
(716) 425-3753 • Fax: (716) 425-3835

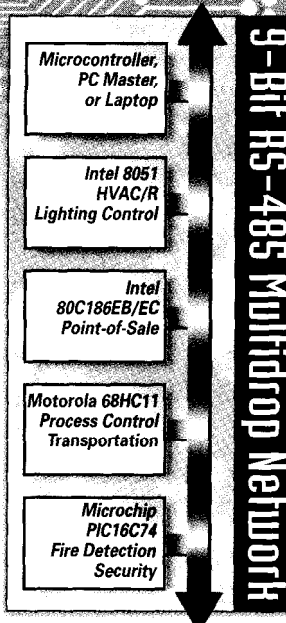
#504

## Microcontroller Networks (μLANs)

With Cimetrics' 9-Bit μLAN you can link together up to 250 of the most popular 8- and 16-bit microcontrollers (8051, 80C196, 80C186EB/EC, 68HC11, 68HC16, PIC16C74).

### The 9-Bit μLAN is:

- ▶ Fast-A high speed (62.58 baud) multidrop master/slave RS-485 network
- ▶ Flexible-compatible with your microcontrollers
- ▶ Reliable-robust 16-bit CRC and sequence number error checking
- ▶ **Efficient**— low microcontroller resource requirements (uses your chip's built-in serial port)
- ▶ **Friendly**- Simple to use C and assembly language software libraries, with demonstration programs
- ▶ Complete- includes network software, network monitor and RS-485 hardware
- ▶ **Practical**- applications include data acquisition and distributed control



Cimetrics Technology • 55 Temple Place • Boston, MA 02111-1300 • Ph 617.350.7550 • Fx 617.350.7552

#105

## !NEW! FROM CIRCUIT CELLAR PROJECT FILE, VOL. II

**\$17.95**

**SAVE 28% OFF  
COVER PRICE OF \$24.95**

**All NEW Projects to SPARK Your Imagination!**

**GPZ8 Audio Sampling System  
Wiring Your House for the 21st Century  
Multiprocessor Architecture using DSP  
ANDMUCHMORE!!!**

VISA, Mastercard, or International Postal Money Order (U.S. funds drawn on U.S. bank only)

## Circuit Cellar Project File

4 Park Street  
Vernon, CT 06066

Tel: (860) 875-2199  
Fax: (860) 872-2204

\*includes domestic delivery. Please add \$6 per copy for delivery to Canada & Mexico, add \$8 per copy for delivery to other non-U.S. addresses.

# NEW PRODUCT NEWS

## REMOTE PC POWER CONTROL

Server Technology introduces an upgraded version of Remote Power On/Off, a telephone-

activated power switch for PCs. The new version adds several reboot enhancements to support remote

computing users who leave their PC on continuously.

Features include: No Answer Automatic reboot (any incoming call that is not answered causes the host PC to automatically reboot); Eight-Ring, No-Answer Forced reboot (a reboot is forced if the call is not answered within eight rings); and Infinite Power On feature (a user can call the host PC and turn it on or off).

Also featured are Locked Modem Safeguard (provides a two-hour connect limit to prevent a stuck modem from locking the line) and One Second re-

Remote Power On/Off lists for \$169.95. It is also available with **pc-ANYWHERE** for \$199.95. It is fully compatible with standard internal or external modems, comm software, remote-control applications, and NetWare. Installation requires three cables, which are provided.

Server Technology, Inc.  
1288 Hammerwood Ave.  
Sunnyvale, CA 94089  
(408) 745-0300  
Fax: (408) 745-0392  
<http://www.powerboot.com/>

#505



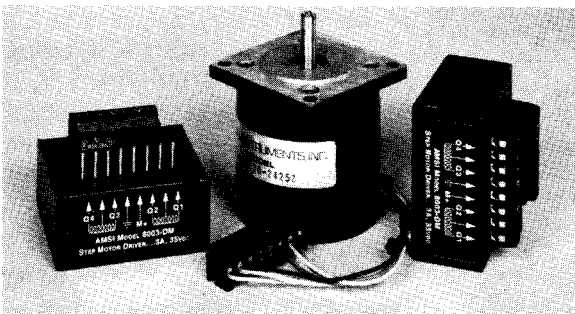
## STEP MOTOR DRIVER

8003-DM Step Motor Driver... \$80  
302SM Step Motor... \$39"

1/2-step, full-step, remote enable, 3A/phase, 35V max, single supply operation, LED's indicate pwr & motion, TTL, parallel printer port compatible.

**FREE SOFTWARE** for running from Printer Port

- 60 oz-in, 12VDC, 200 steps/rev, size 23 (as shown)



Ask for our **FREE Catalog**



American Scientific Instrument Corp  
PO Box 651  
Smithtown, NY 11787  
(516) 361-9499 Tel  
(516) 265-6241 Fax



## B.G. MICRO, INC.

P.O. Box 280298  
Dallas, Tx 75228

Our Internet address is: [BGMICRO@IX.NETCOM.COM](mailto:BGMICRO@IX.NETCOM.COM)



Orders Only 1-800-276-2206  
Tech Support 214-271-9634  
Fax 214-271-2462  
Local Orders 214-271-6546

### SERIOUS STEPPER MOTOR

Serious Stepper Motor!

Original catalog cost \$209.00 each. By Bodine. 8 wires, 200 steps per revolution (1.80). Fourcoils. 3.1V@4A per winding. Easily runs at higher voltages by using series resistors. Massive Torque 300 oz-in. Motor measurements (not including shaft)

4-1/4" L x 3-1/4" D. Mounting plate measurements 3-3/8" square Shaft length 1-1/8" Shaft diameter 3/8" Very limited stock. Bodine Motor #34T2-BEHH No. 2013. With wiring diagram. Shipping on these 7 lb hunks—\$7.00 each.



**Brand new in factory boxes \$34.95 or 2/\$59.95**

### STEPPER MOTOR CONTROLLER BOARD

5 amp stepper motor controller board by Bodine #THD1801 B. Used to drive above stepper at 24Vdc. Uses 4-2N6283-20 amp NPN darlington. Factory boxed with data booklet. Very limited stock!!! Original cat price \$191.00 each. Our Price..... \$24.00



### STEPPER MOTOR RESISTORS CATALOG

Series Power Resistor Pack for Stepper Motor use at 24Vdc. Two 5.1 ohm resistors @ 130W. One 10 ohm resistor @ 15 W. .... \$5.95 (3 pieces)



Please call or write for a free B.G. Micro catalog, loaded with all kinds of surplus goodies.

The above items shipped in the continental U.S. only, with the stated shipping charges!!!  
**TERMS:** (Unless specified elsewhere) Add \$4.00 postage, we pay balance on Order Under 5 lbs. Orders over \$50.00 add .85 for insurance. No C.O.D. Texas Residents add 1/4% Tax. 90 Day Money Back Guarantee on all items. All items subject to prior sale. Prices subject to change without notice. Foreign Orders U.S. Funds Only. We cannot ship to Mexico or Puerto Rico. Canada, add \$7.50 minimum shipping and handling. Countries other than Canada, add \$15.00 minimum shipping and handling.

Call and place your order today! 1-800-276-2206



## FEATURES

12

In-System-Programmable  
PLDs from Lattice

20

Building a Low-Cost  
CPLD Development  
System

28

Take Your PIC

36

7th Annual Design  
Contest Winners

# In-System- Programmable PLDs from Lattice

While PLDs have been around for many years, Lattice has put a new twist on an old workhorse. With their line of ISP parts, the devices can be programmed without taking them out of the circuit.

## FEATURE ARTICLE

**Stuart Ball**



The Program-  
mable Logic  
Device (PLD) has  
become a standard tool  
in the engineer's toolkit. PLDs permit  
us to:

- pack a lot of logic into a single chip
- reduce inventory—the same device can be used in different designs by changing the internal programming
- fix bugs without adding wires to the board.

But, for the experimenter or independent designer, PLDs have been something of a problem. While PLD development software is available, the equipment needed to program the devices is expensive. And, unlike EPROMs, the programming specifications for PLDs are not published by the manufacturers, making it difficult to design your own programmer.

Even for the engineer, PLDs have drawbacks. Because the parts have to be programmed, they must be handled twice, increasing the opportunity for damage from electrostatic discharge or other causes. And, if the parts are soldered to the board for better reliability, the flexibility to fix problems by reprogramming them is lost.

Lattice Semiconductor has changed all that. They have a line of PLDs and larger high-density devices that can be programmed in-circuit. In this article, I'll look at these Lattice parts and show how they can be used to produce a useful debugging tool.

### PLD PRIMER

Imagine that an IC manufacturer has developed a small PLD. The device

fits in an 8-pin miniDIP package. This part has three inputs (on pins 1, 2, and 3) and two outputs (on pins 5 and 6). Figure 1 shows the logic of such a device. Like an actual PLD, the true and complement of each input is available, and the outputs are wrapped back into the fuse array (labeled as columns A-H in Figure 1).

The AND gates represent the product terms of the PLD. Each product term produces the product (logical AND) of any combination of terms in the fuse array. Next, the product terms are summed with the OR gates.

This AND/OR structure is typical of PLDs, and is referred to as a *sum-of-products architecture*. In this hypothetical PLD, each output term has two corresponding AND gates and therefore has two product terms. To put it another way, the output can be the OR of any two AND functions.

Let's say you want to program the output at pin 6 to perform the exclusive-OR function of pins 1 and 2. You can write this logically as:  $P_{in\ 6} = P_{in\ 1} \text{ AND } (\text{NOT } P_{in\ 2}) \text{ OR } ((\text{NOT } P_{in\ 1}) \text{ AND } P_{in\ 2})$ . If you want pin 5 to be high when pin 6 is high and pin 3 is low, you can write:  $P_{in\ 5} = P_{in\ 6} \text{ AND } (\text{NOT } P_{in\ 3})$ .

To make the PLD perform these functions, you'd have to program fuses in the fuse array to direct the appropriate (true or inverted) inputs to the correct product-term AND gates. The circles in Figure 1 indicate the fuse connections. The topmost AND gate generates the  $(P_{in\ 1} \text{ AND } (\text{NOT } P_{in\ 2}))$  term, and the second gate decodes the  $((\text{NOT } P_{in\ 1}) \text{ AND } P_{in\ 2})$  term. The third gate decodes the term for pin 5. The fourth (lowest) gate is unused.

If you work this out, you'll find that the number of connections required in the fuse matrix equals the number of product terms times the number of input and feedback terms. Actual PLDs have several product terms per output, so even the 16L8 (a

relatively simple PLD with 10 inputs, 8 outputs, and 64 product terms) needs an array of 1280 fuses. Most PLDs have additional fuses to control output tristate, registers, and other capabilities.

PLDs come in several flavors. Some have D-type registers at the outputs. On these PLDs, the sum-of-products term drives the D input to the flip-flop. Some PLDs have fixed output polarity, some have programmable polarity, and some have a mix of registered and combinatorial outputs.

Early PLDs used bipolar technology and could only be programmed once. This is where the term "fuse" came from, since there was a silicon fuse that blew open when the part was programmed. Newer parts use flash or other memory technology that permits the part to be reprogrammed. Although people still sometimes talk about fuses in these parts, they are really referring to programmable memory cells.

PLDs are used anywhere a logic function can be reduced to a set of sum-of-terms equations. They permit designers to pack several ICs' worth of discrete logic into one chip. Typical

applications include memory-address decoders, small state machines, and random-logic replacement.

## THE 22V10

The 22V10 has become one of the most popular PLDs. Figure 2 shows the internal structure of the 22V10, the DIP and PLCC pinouts, and a simplified diagram of the OLMC. Designated the GAL22V10 by Lattice (GAL for Generic Array Logic), it has 12 inputs and 10 outputs. The outputs can be combinatorial or registered and can be tristated to drive a bus.

A global-product term can set or reset all the output registers. Unneeded outputs can be used as inputs. And, while early versions of the 22V10 from some manufacturers were one-time programmable, the Lattice parts can be erased and reused. The product-term array has 132 product-term (AND) gates. Each product-term gate has 44 inputs—one for the true and complement of each input pin and one for the true and complement of each output-feedback term.

Each output is driven by an Output Logic Macrocell (OLMC), as shown in Figure 2. Each OLMC has as inputs a unique sum-of-products term, a tristate-control product term, and the global clock, reset, and preset inputs.

Each OLMC contains a D flip-flop. The D input of the flip-flop is driven by the sum-of-products term for that OLMC. The output of each OLMC can be individually programmed to be the true or complement of either the flip-flop output or sum-of-products term. The true and complement of the output is fed back into the product term array, and may be used as a product term itself. If the D flip-flops are used, the clock comes from pin 2 (PLCC, pin 1 on the DIP). All flip-flops are driven by this common clock.

Not all outputs of the 22V10 have the same number of product terms. The product terms for the PLCC pinout are allocated as in Table 1. So, if

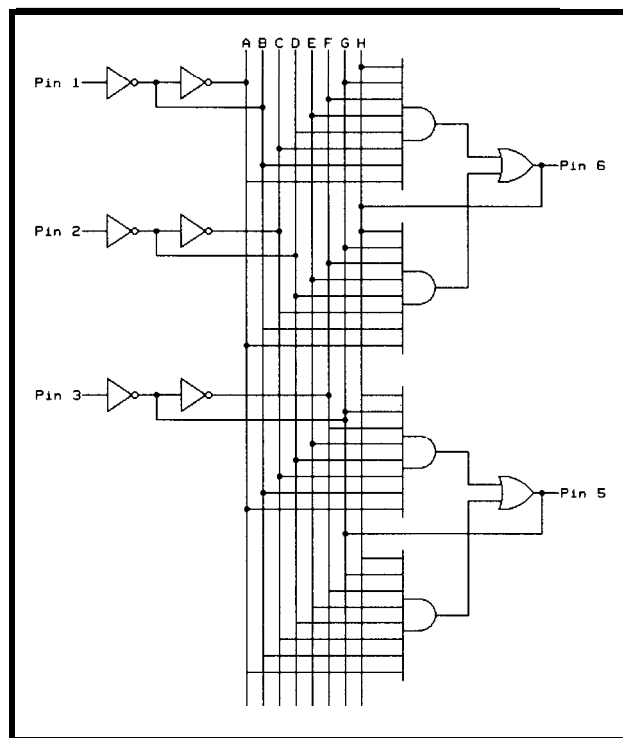


Figure 1—An example of a simple PLD shows how the inputs and output-feedback lines are combined first through AND gates, then OR gates.

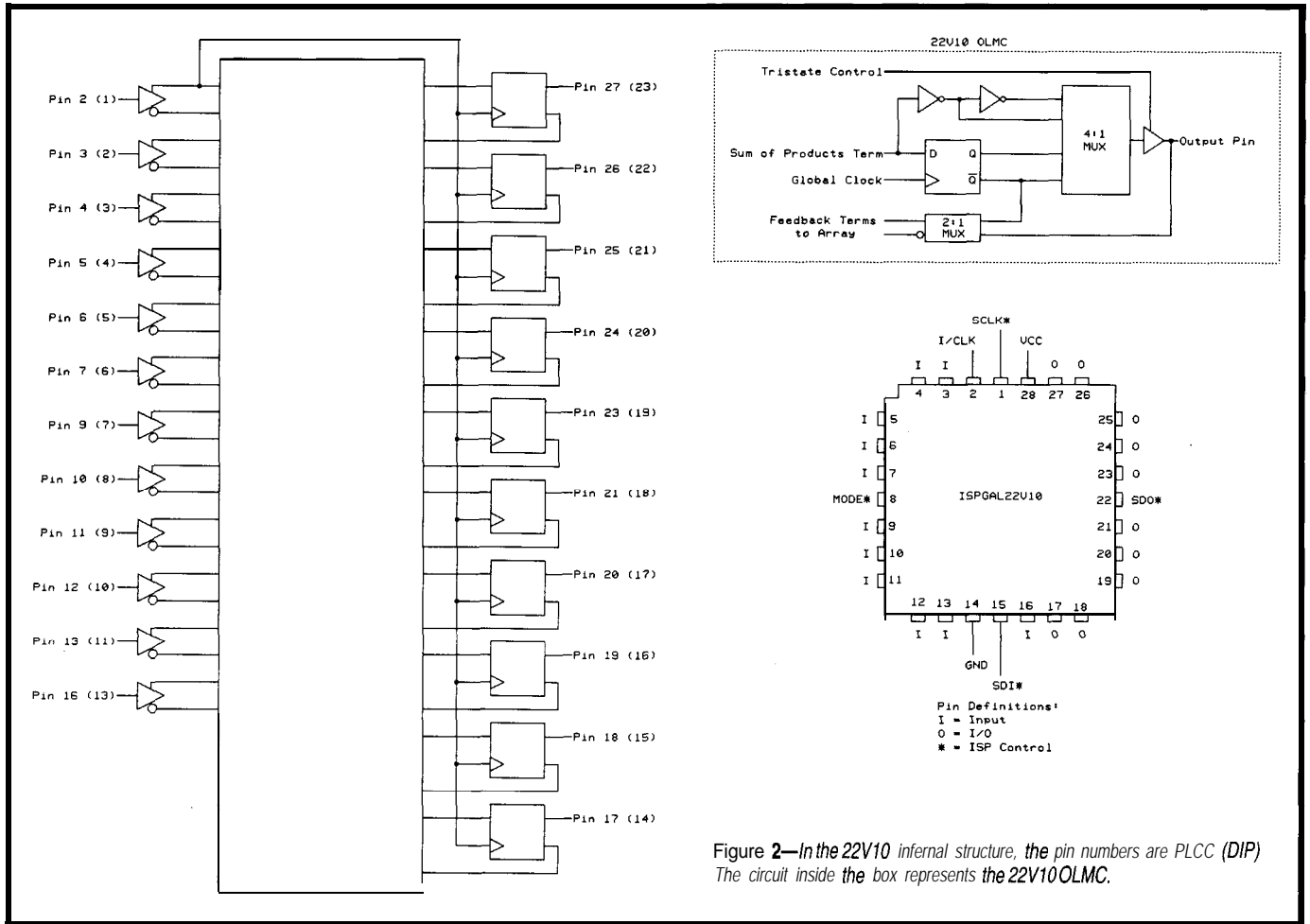


Figure 2—In the 22V10 internal structure, the pin numbers are PLCC (DIP). The circuit inside the box represents the 22V10 OLMC.

you need, say, 15 product terms for a given output pin, you put that function on output pins 21 or 23 since those are the only pins with enough terms.

### ISPGAL22V10

Even with the flexibility of the 22V10, it has the same drawbacks as any other PLD that is programmed in a programmer. However, Lattice has introduced the ISPGAL22V10 (ISP stands for In-System Programmable), one of a family of parts that can be programmed in-circuit. These parts do

not require a special programmer. ISP parts can be programmed:

- in-circuit
- using a cable from a PC
- using a processor [if one exists] in-circuit
- soldered to a board
- using an inline header and a PC cable after the board is stuffed
- with a special configuration to simplify a factory board-test procedure.

ISP parts make PLDs accessible to the experimenter because they can be programmed without using an expensive programmer. The ISPGAL22V10 is only available as a 28-pin PLCC.

Figure 2 shows the pinout of the PLCC part; it isn't available in a DIP package because Lattice took advantage of the four unused pins of the ordinary GAL22V10 PLCC device to add the ISP capability. These pins (indicated by an asterisk) are used as follows:

- Mode defines the mode of the other ISP pins
- SDI moves serial data into the ISP device
- SDO moves serial data out of the ISP device
- SCLK is the serial data I/O clock.

### INTERNAL ARCHITECTURE

A complete description of the ISPGAL22V10, including all programming information, takes several pages in the Lattice data book and is not reproduced here. However, here's an overview of the ISPGAL22V10 programming architecture.

Lattice ISP devices contain a bank of internal shift registers. The SDI pin shifts data and commands into the device, and the SDO pin reads data out. The ISPGAL22V10 contains four shift registers: Device ID, Instruction, Data, and Architecture.

The 8-bit Device ID register verifies the device type before programming. The 22V10's device type is 08

Pin 17	8 terms	Pin 23	16 terms
Pin 18	10 terms	Pin 24	14 terms
Pin 19	12 terms	Pin 25	12 terms
Pin 20	14 terms	Pin 26	10 terms
Pin 21	16 terms	Pin 27	8 terms

Table 1—Not all the device outputs support the same number of product terms. Care must be used when defining the outputs that enough terms are available for the intended function.

The Instruction register is 5 bits long. It allows 10 commands like the following:

- shift data into the Data shift register
- erase the device
- program the addressed row
- load data from the addressed row for readback

The 138-bit Data register is loaded with the address and data to be programmed. Each programmable cell in the device is numbered and individually programmed. An internal state machine directs data to the proper registers and executes commands loaded into the instruction register.

### PROGRAMMING ISP DEVICES

If you want to program ISP devices in-circuit using software you have written, you should get the Lattice ISP manual. Fortunately, if you plan to program the parts from a PC, Lattice provides software that makes it easy.

The first step in programming any PLD is getting a JEDEC file. This file

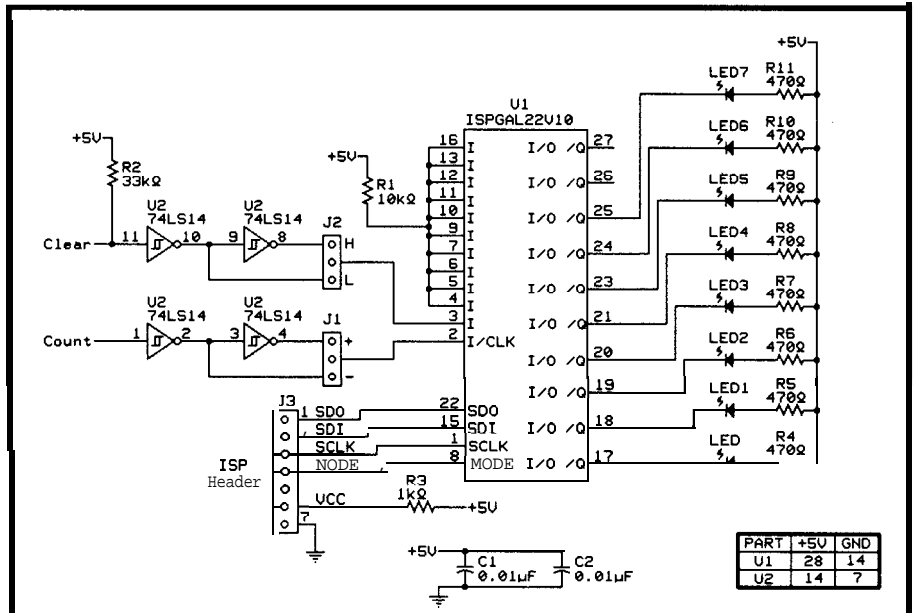


Figure 3-A counter circuit can be made using the ISPGAL22V10. J1 selects *cl* & polarity, J2 selects *clear* polarity, and J3 is the ISP programming connector.

informs the PLD programmer which fuses need to be programmed in the chip. The JEDEC file is a standard format, understood by all PLD programmers and Lattice conversion software.

Several software packages, including CUPL, ABEL, and PALASM, can produce a JEDEC file from input information. While CUPL and ABEL are somewhat expensive, early versions of PALASM often were provided free by

# ENGINEERING PROFESSORS

## LISTEN UP!

The reason you read *Circuit Cellar INK* is obvious.

Like thousands of others, you have come to realize that *Circuit Cellar INK* presents technically relevant computer-based applications in a comprehensive and instructive manner. In fact, our projects are presented just the way you'd do them if only you had the time.

Since 1994, we have provided thousands of college students with industry-quality development tools and information. This program is **FREE** and because of the continued success we have experienced, we'd like to extend our offer to the 1995-1996 school year. The fall semester is upon us...

If you're an engineering professor or teach qualified technical students, we'd like to give you and all of your students free subscriptions to *Circuit Cellar INK*. Please tell us how we can contact you directly so you don't miss a single issue.

**WE ARE HELPING AMERICA REGAIN THE COMPETITIVE EDGE. WE'RE HOPING YOU WILL TOO!**

Rose Mansella, Circulation Coordinator  
Circuit Cellar INK  
4 Park St.  
Vernon, CT 06066

Tel: (860) 875-2199  
Fax: (860) 872-2204  
Internet: [rose.mansella@circellar.com](mailto:rose.mansella@circellar.com)

MMI (Monolithic Memories Inc.), the company that originated PLD devices. You might be able to find an old copy; version 2.23 was the last MMI version, as far as I know. MMI was bought out by AMD a few years ago, and the new PALASM is no longer free. But it is still inexpensive, and it supports PLDs such as the 22V10.

PLD compilers can take input in several forms. The simplest form is Boolean equations, like those used for our hypothetical miniPLD earlier but with symbols for the logical functions. For example, PALASM uses \* for AND, and + for OR. CUPL uses & and #, respectively.

Once you have entered and compiled your equations, use the Lattice ISP software to program the parts. For the 22V10, there are two programs:

- JEDTOISP, which converts the JEDEC file to an ISP file
- I22\_PROG, which programs the ISP format file into the device.

A third program, JEDFIX, can be used to make the JEDEC file compatible with JEDTOISP. While standard, the JEDEC file includes header information that can cause problems for JEDTOISP because different PLD compilers produce different headers. JEDFIX strips the header out as you can do with a text editor.

I'll describe the syntax for using these three programs in just a minute. First, let's look at a circuit that makes use of the ISPGAL22V10.

## THE SUPER LOGIC PROBE

In debugging circuits, I've used everything from a voltmeter to a \$15,000 logic analyzer. One of the most useful techniques I have found to use a counter and an LED bank.

A typical logic probe shows you if a signal is high or low and has a latch to capture state changes. A really good logic probe, like those sold by HP, blinks to show a pulse train. The drawback to these is that you can't always

tell how many or how often pulses occur in a group.

The Super Logic Probe consists of an 8-bit binary counter driving a bank of LEDs. Each bit of the counter drives an LED, so the LEDs represent the binary count. The counter gets a clock input and a clear input. Three-pin shunt jumpers increment the counter on the rising or falling edges of the clock and reset it when the clear input is high or low.

Printer port connector DB-25P		Circuit header	
Fcn	Pin	Pin	Fcn
'ACK	10	1	SDO
D0	2	2	SDI
D1	3	3	SCLK
D2	4	4	MODE
*FLT	15	6	VCC (sense)
GND	20	7	GND
D6	8		Port sense
PE	12		

Figure 4—The cable for in-circuit programming connects the printer port on the PC and the 7-pin header on the circuit board's logic probe.

The circuit (Figure 3) is implemented with an ISPGAL22V10. The 74LS14s buffer the clock and clear inputs to provide polarity selection and to protect against noisy inputs.

A 7-pin header brings the ISP signals to the 22V10 so it can be programmed. Figure 4 shows the wiring of the cable for programming the board. I use a smaller header than the standard Lattice ISP cable. The other end of the cable plugs into the PC's printer port.

Listing 1 shows the PALASM equations for the logic probe. In addition to using \* for AND and + for OR, the symbol := indicates registered output and = indicates combinatorial output. The C L R input uses the 22V10 global clear function to force all the outputs off.

## PROGRAMMING THE 22V10

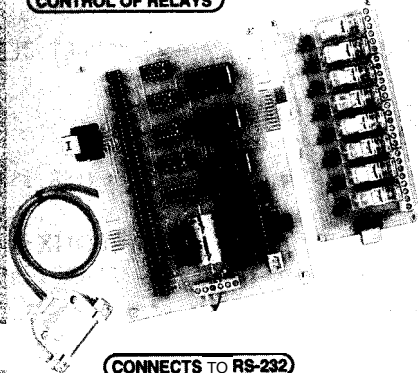
After entering and compiling the equations, the JEDEC file must be created. The command

**JEDTOISP COUNTER.JED COUNTER.ISP**

**creates** the file **COUNTER.ISP** that contains the ISP programming infor-

# RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

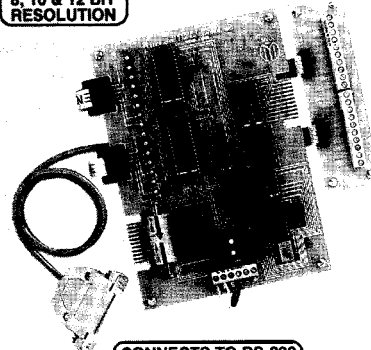


CONNECTS TO RS-232

**AR-16 RELAY INTERFACE (16 channel).....\$ 89.95**  
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.  
**AR-2 RELAY INTERFACE (2 relays, 10 amp)....\$ 44.95**  
**RD-8 REED RELAY CARD (8 relays, 10 VA).....\$ 49.95**  
**RH-8 RELAY CARD (10 amp SPDT, 277 VAC)....\$ 69.95**

# ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

**ADC-16 A/D CONVERTER\* (16 channel/8 bit)....\$ 99.95**  
**ADC-8G A/D CONVERTER\* (8 channel/IO bit)....\$124.90**  
Input voltage, amperage, pressure, energy "sage", joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately).  
**ADC-8E TEMPERATURE INTERFACE\* (8 ch)....\$ 139.95**  
Includes term. block & 6 temp. sensors (-40' to 146' F).  
**STA-8 DIGITAL INTERFACE\* (8 channel) . . . . . \$99.95**  
Input on/off status of relays, switches, HVAC equipment, security devices, smoke detectors, and other devices.  
**STA-8D TOUCH TONE INTERFACE\*.....\$ 134.90**  
Allows callers to select control functions from any phone.  
**PS-4 PORT SELECTOR (4 channels RS-422)....\$ 79.95**  
Converts an RS-232 port into 4 selectable RS-422 ports.  
**CO-485 (RS-232 to RS-422/RS-485 converter).....\$ 44.95**

\*EXPANDABLE...expand your interface to control and monitor up to 512 relays, up to 576 digital inputs, up to 126 analog inputs or up to 128 temperature inputs using the PS-4, EX-16, ST-32 & AD-16 expansion cards.

FULL TECHNICAL SUPPORT...provided over the telephone by our staff. Technical reference & disk including test software & programming examples in Basic, C and assembly are provided with each order.

HIGH RELIABILITY...engineered for continuous 24 hour industrial applications with 10 years of proven performance in the energy management field.

CONNECTS TO RS-232, RS-422 or RS-485...use with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud). Use our 600 number to order FREE INFORMATION PACKET. Technical information (614) 464-4470.

24 HOUR ORDER LINE (800) 842-7714  
Visa-Mastercard-American Express-COD

International & Domestic FAX (614) 464-9656  
Use for information, technical support&orders.

ELECTRONIC ENERGY CONTROL, INC.  
360 South Fifth Street, Suite 604  
Columbus, Ohio 43215-5438

mation. If JEDTOISP rejects the JEDEC format, modify it using JEDFIX before running JEDTOISP:

JEDFIX COUNTER.JED >FIXCOUNT.JED

Then run JEDTOISP this way:

JEDTOISP FIXCOUNT.JED COUNTER.ISP

An alternative to using JEDFIX is to delete everything in the JEDEC file up to the ctrl-B character with a text editor. Ctrl-B usually shows up as a happy face symbol.

To program the 22V10, connect power and ground to the counter circuit, and connect the ISP programming cable to the PC's printer port and the counter circuit's ISP connector. Then, use the following command:

I22\_PROG COUNTER.ISP 0

The 0 specifies which printer port to use. For a printer port other than LPT0, replace 0 with the appropriate number.

Listing 1--PALASM equations for the logic probe.

```
; Pin definition (PLCC pinout)
; PIN 2 3 4 5 6 7 9 10 11 12 13 14
      CLK CLR NC NC NC NC NC NC NC NC NC GND
; PIN 16 17 18 19 20 21 23 24 25 26 27 28
      NC /Q0 /Q1 /Q2 /Q3 /Q4 /Q5 /Q6 /Q7 NC NC VCC GLOBAL
```

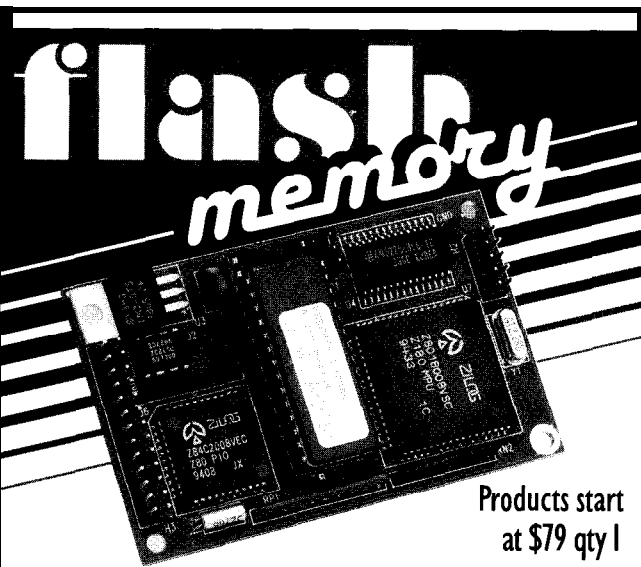
```
; Pin descriptions:
; CLK: Input clock
; CLR: Clears Q0-Q7
; Q0-Q7: An 8-bit binary counter.
      The outputs are true LOW, so a true
      output turns the LED on.
```

EQUATIONS

```
; Note that PALASM uses * for logical AND,
; + for logical OR, = for combinatorial outputs,
; and := for registered outputs. / indicates negation.
GLOBAL.RSTF = CLR
```

```
Q0 := /Q0
Q1 := Q0 * /Q1
      + Q1 * /Q0
Q2 := Q0 * Q1 * /Q2
      + Q2 * /Q0 + Q2 * /Q1
Q3 := Q0 * Q1 * Q2 * /Q3
      + Q3 * /Q0 + Q3 * /Q1 + Q3 * /Q2
```

(continued)



**flash memory**

Products start at \$79 qty 1

Now get C-programmable miniature controllers with non-volatile, "flash" memory. Our **Dynamic C™** development system makes it easy-only \$195. Call our **AutoFAX** today! Dial 916753.0618 from your FAX and request catalog #18.

**ZWORLD ENGINEERING**

1724 Picasso Ave.  
Davis, CA 95616  
916.757.3737  
916.753.5141 FAX



**C COMPILERS  
CROSS ASSEMBLERS  
DEBUGGERS**

68HC08	8051/52
6809	8080/8085
68HC11	8086/186
68HC16	8096/196

**Low Cost!!** PC based cross development packages which include EVERY THING you need to develop C and assembly language software for your choice of CPU.

- MICRO-C compiler, optimizer, and related utilities.
- Cross Assembler and related utilities.
- Hand coded (efficient ASM) standard library (source included).
- Resident monitor/debugger (source included)\*
- Includes Integrated Development Environment and command line tools (incl: editor, telecomm and many other utilities).

\* 68HC08 and 68HC16 kits do not include monitor/debugger.

**Each Kit: \$99.95 US + s&h (please specify CPU)**  
**Super DK (supports all 8 CPU families) \$300.00 US**

**Call or write for our free catalogue of development tools.**  
**FREE Demos available on BBS (or send \$5 for diskette)**  
**We accept Cheque/MO/Purchase Orders/VISA**

**Dunfield Development Systems**  
P.O. Box 31044 Nepean, Ont. K2B8S8 CANADA  
Tel: 613-256-5820 Fax: 256-5821 BBS: 256-6289  
email:76162.3355@compuserve.com

Listing 1-continued

```
Q4 := Q0 * Q1 * Q2 * Q3 * /Q4
      + Q4 * /Q0 + Q4 * /Q1 + Q4 * /Q2 + Q4 * /Q3
Q5 := Q0 * Q1 * Q2 * Q3 * Q4 * /Q5
      + Q5 * /Q0 + Q5 * /Q1 + Q5 * /Q2 + Q5 * /Q3 + Q5 * /Q4
Q6 := Q0 * Q1 * Q2 * Q3 * Q4 * Q5 * /Q6
      + Q6 * /Q0 + Q6 * /Q1 + Q6 * /Q2
      + Q6 * /Q3 + Q6 * /Q4 + Q6 * /Q5
Q7 := Q0 * Q1 * Q2 * Q3 * Q4 * Q5 * Q6 * /Q7
      + Q7 * /Q0 + Q7 * /Q1 + Q7 * /Q2
      + Q7 * /Q3 + Q7 * /Q4 + Q7 * /Q5 + Q7 * /Q6
```

I22\_PROG programs and verifies the device. If errors occur, it tells you. If the circuit won't program, first check the wiring of the ISP pins and the ISP cable. It may also be that the cable is too long, causing noise problems with the 22V10. Try to keep it under 3'.

## TESTING AND USE

The simplest test you can perform is to jumper the clear input to be low true and the clock to be positive edge. The 33-kΩ pull-up resistor on the clear input ensures that it stays high if the wire is unconnected. Touch the clock input to ground. The contact bounce increments the counter and LEDs several counts. Touch the clear wire to ground, and the LEDs all go out.

You can perform a more exhaustive test by connecting a slow signal source to the clock input. If the signal source is slow enough (around 2 Hz), you can see each LED change and verify that the count increments in a binary fashion.

The uses for this circuit are many. I've used one to count the steps going to a stepper motor, the encoder pulses from a servo motor, and even the number of instructions that a balky microprocessor managed to execute before it died.

Since the 22V10 is reprogrammable, you can modify the counter as needed. For example, you could connect one of the unused inputs so it enables and disables counting without clearing the counter. Or, you could wire the inputs to decode a port ad-

dress from a microprocessor, and count the number of times that port is accessed.

## PROGRAMMING

The ISPGAL22V10 need not be programmed from a PC. Inputs can come from a microprocessor in the target circuit. This technique allows the 22V10 function to be changed at powerup, for example, when the microprocessor detects whether a particular option is installed.

While Lattice supplies details in the data book that tell how to do this, they also supply C source code so you don't have to write it all yourself. Lattice sells starter kits that include an ISPGAL22V10, a programming cable, and the appropriate programs. The programs themselves are also available on the Lattice ISP BBS. The files needed are ISP22V10.ZIP and JEDFIX.ZIP, but check to see if there are later versions loaded with different names.

If you use the Lattice starter kit, wire your ISP header to match their cable. The diagram of the Lattice cable is included with the kit.

## OTHER ISP DEVICES

In addition to the 22V10, Lattice makes a number of other ISP devices, including a line of high-density parts. I've used their ispLSI1016 in a number of designs. This part has 2000 gates, 96 D-type flip-flops, and 32 I/O pins. Unlike ordinary PLDs, these larger devices don't have a fixed number of product terms per output. Instead, they

have a global-routing pool, an array of product terms allocated by special software to implement the required functionality.

Creating a design for one of these parts is typically a two-step process. First, the PLD compiler is run to create an intermediate file. Then, a fitter program from Lattice is run. The fitter reads the intermediate file produced by the PLD compiler and allocates the resources on the chip, producing a JEDEC file. Lattice provides DDOWN-LD, a download program for the large devices. Lattice also has a complete development system that does not depend on third-party compilers.

High-density logic devices are available in both ISP and nonISP versions. The nonISP parts are a little less expensive, so I put those on the manufacturing bill of materials. But, I keep a tube of ISP parts in my desk drawer for engineering prototypes.

## WRAPPING UP

The Lattice ISP product line solves many of the problems you may have encountered in current PLDs, putting them within the reach of any designer. You, too, can use PLDs to create more innovative circuits than you did before, thanks to Lattice ISP products. □

*Stuart Ball has spent the last 15 years working on systems as diverse as Global Positioning Systems and single-chip interface translators. He is currently employed as a principal engineer at BancTec Technologies, a manufacturer of document-processing equipment for the banking industry. He may be reached at (405) 354-5042.*

## SOURCE

**GAL22V10, ISPGAL22V10**  
Lattice Semiconductor Corp.  
5555 NE Moore Ct.  
Hillsboro, OR 97124  
(503) 681-0118  
Fax: (503) 681-3037  
BBS: (503) 693-0215

## IRS

401 Very Useful  
402 Moderately Useful  
403 Not Useful

# FEATURE ARTICLE

David Van den Bout

## Building a Low-Cost CPLD Development System

Unlike Steve, David prefers software. You can easily fix errors, document through variables, and reuse modules. Hence, the CPLD fits his bill. He shows us how to build a complete CPLD development system.



often design circuits, but I never seem to get around to building them!

Searching for a breadboard, finding the right chips, cutting wires, and making sure they get in the correct holes takes the fun out of a project.

Then, once it's built, I have to transfer it to a soldered prototyping board to make room for another project. Next, I need to document it so I can fix it if it breaks. And, I have to build another complete copy of the circuit if I want to use it as a part of another project...

Software seems so much easier! I write subroutines and test them with a debugger. If I find errors, a little editing gets rid of them. If I choose reasonable variable and function names, most of the documentation gets done automatically. I can use each subroutine as many times as I want. I can even give copies to other people over the Internet or on diskettes.

That's why I enthusiastically greeted the appearance of complex programmable logic devices (CPLDs) and field-programmable gate arrays (FPGAs) in the mid 1980s. CPLDs and FPGAs make building hardware look like writing software.

CPLDs and FPGAs contain thousands of logic gates that can be rewired

by reprogramming their internal memory. If I want to build a UART, I just write the truth tables or Boolean equations using a hardware description language (HDL), put them in a file, compile it, and download it into a CPLD chip. If I decide I'd rather have a microprocessor, I can change the programming and build one.

It wasn't that easy at first. Early CPLDs and FPGAs didn't contain many logic gates and cost hundreds of dollars each. Worse, the programming software cost thousands!

Luckily, things have changed. Today, you can buy CPLDs and FPGAs with up to 10,000 reconfigurable logic gates for less than \$100, and some of the programming software is free!

In this article, I'll show you how to build a simple but complete CPLD development system for just \$120. But first, let's take a look at the basics.

### WHAT ARE CPLDS?

In the beginning [OK, in the '60s], there was discrete logic. Systems were built from lots of individual chips with a spaghetti-like maze of wiring between them.

Such systems were difficult to modify after you built them. In fact, after a week or two it was difficult to remember what each chip was for! Manufacturing the systems took a lot of time. Each design change meant rewiring, which usually meant building a new printed circuit board.

The chip makers solved this problem by placing an unconnected array of AND-OR gates in a single chip called a programmable logic device (PLD). You could program a PLD with a set of Boolean sum-of-product equations so it would perform the logic functions needed in your system. The ability to internally rewire PLDs lessened the need to redo the circuit board if a design change occurred.

Simple PLDs such as the one shown in Figure 1 only handle 10-20 logic equations, so you can't fit a large logic design into just one. You have to figure out how to break larger designs apart and fit them into a set of PLDs. This process is time-consuming and means you have to interconnect the PLDs with wires.



The wires were a big no-no. Eventually, you made a design change that couldn't be handled just by reprogramming the PLDs, and you had to build a new circuit board.

Again, the chip makers came to the rescue by building much larger programmable chips called *complex programmable logic devices* (CPLDs). With these, you could essentially get a complete system onto a single chip.

A CPLD contains several PLD blocks whose inputs and outputs are connected by a global interconnection matrix. Thus, a CPLD has two levels of programmability. Each PLD block can be programmed as can the interconnections between the PLDs.

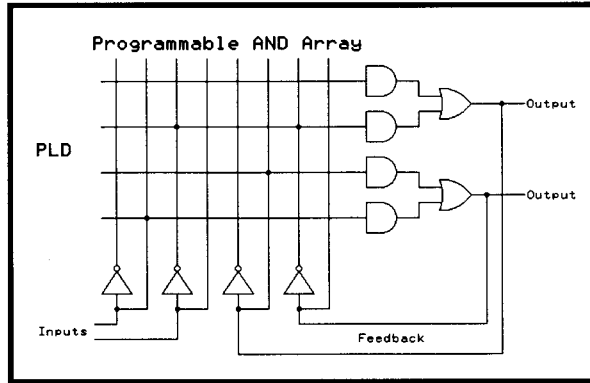


Figure 1—A simple PLD allows the connection of any input or output in any combination to the AND/OR logic.

The EPX780 CPLD shown in Figure 2 is a good example of a CPLD. It contains eight configurable function blocks (CFBs) which are essentially 24V10 PLDs with 24 inputs and 10 outputs. Any output from any CFB can

be connected to any input of any CFB through the global interconnection matrix. Most CFB outputs are accessible externally, but not all of them due to the limitations of the 84-pin PLCC package.

CPLD manufacturers make connections within and between PLD blocks several ways. Some make chips with fuses or antifuses that are programmed by passing a large current through them. These CPLDs and FPGAs are one-time programmable (OTP); you can't rewire them internally if the fuses blow.

Other manufacturers make the connections using pass transistors. A charge, stored on the transistors' gate electrodes with a high-voltage pulse, opens and closes them. This type of programmable device resembles an E(E)PROM.

A special programmer erases or programs the chip before you place it in your circuit. That's fine, unless you have the CPLD or FPGA soldered into a circuit board and then decide you want to change it.

Finally, some manufacturers use static RAM bits to control the pass transistors for each interconnection. By loading a RAM bit with a 1 or 0, you control whether the switch is closed or open and therefore whether two logic elements are connected. You can reprogram CPLDs built with RAM switches without removing them from a circuit board. They are said to be in-circuit reconfigurable or in-circuit programmable. The EPX-780 CPLD falls into this category.

Regardless of the interconnection method, you can see that it would

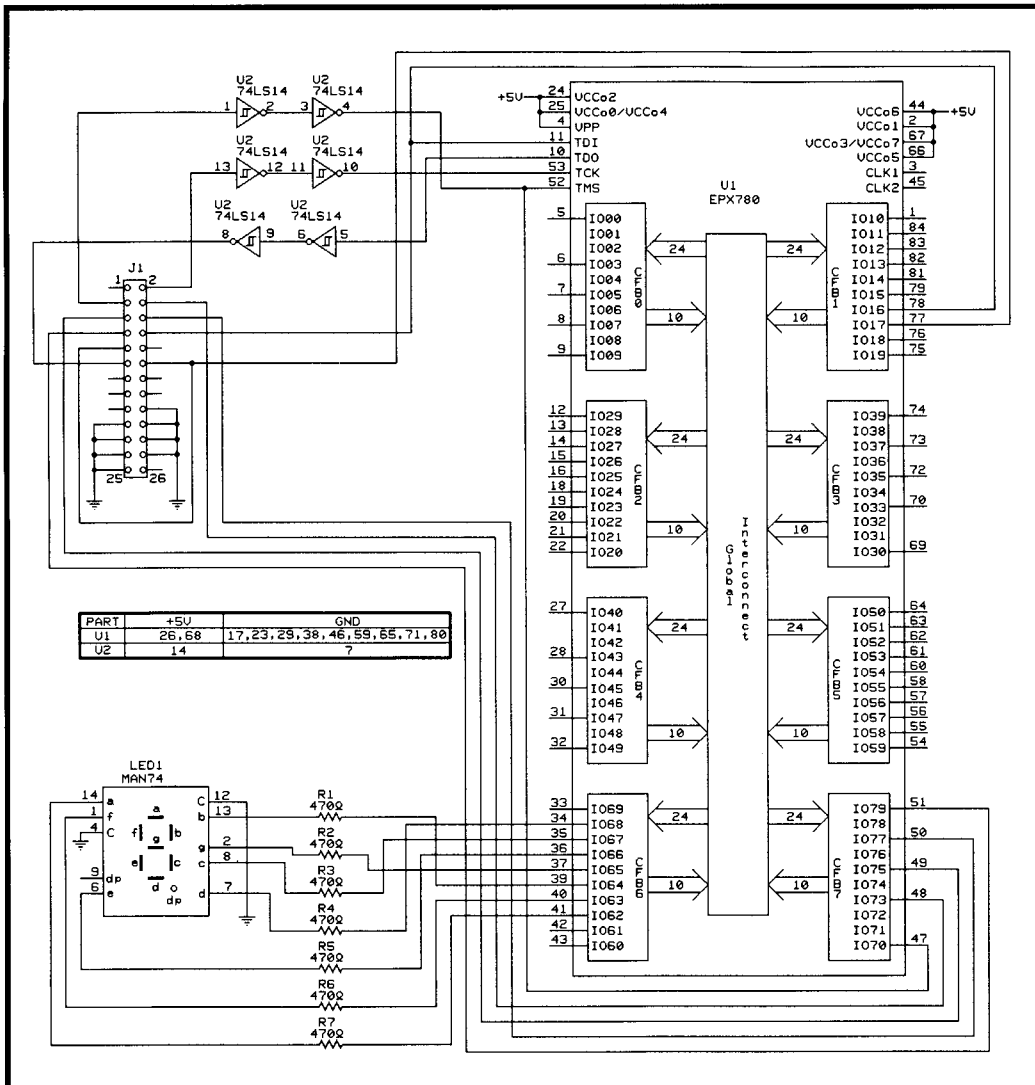


Figure 2—A low-cost CPLD development system consists of little more than some Schmitt triggers to clean up signal edges, an LED display for diagnostics, and a connector.

be quite a chore figuring out which switches to open and close to create a logic circuit. That's why the chip manufacturers provide *device fitters*.

These programs take a description of your logic design as input, compile it, and output a binary file that's downloaded into a CPLD so it acts like your design. Some device fitters compile logic circuits directly from a schematic editor. Other device fitters require you to describe your logic circuit using an HDL like PAL-ASM or ABEL.

When choosing a CPLD, consider both the cost of the CPLD chip and of the programming software and hardware. For most people experimenting with CPLDs, the cost of the actual chip is incidental to the thousands of dollars the programming software costs.

The EPX780 CPLD is a notable exception; its programming software is free of charge. Also, the EPX780 can be programmed by simply connecting it to a PC's printer port, so no expensive programming hardware is needed. And, since the EPX780 stores its configuration in RAM, you can use the same chip on many projects.

## CPLD ASSEMBLY

Any CPLD development system must allow you to:

- download new logic designs into the CPLD
- test the functions of the downloaded designs
- connect the CPLD to other components.

In the EPX780-based development system I describe here, all programming is done through the 4-pin JTAG port (consisting of the TCK, TMS, TDI, and TDO pins). You can control programming easily and cheaply by using the printer port of a PC. You can also use the printer port to apply low-

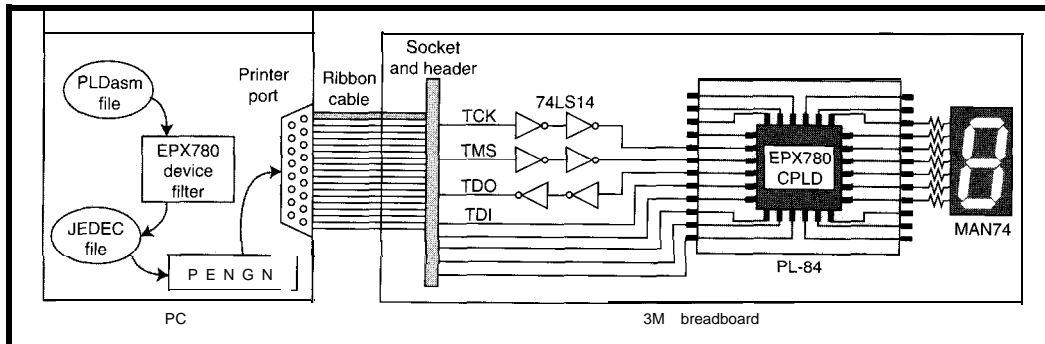


Figure 3—The development system minimizes the hardware necessary by relying heavily on PC-based software.

speed logic levels to some of the EPX780's pins for debugging. A 7-segment LED digit provides visual feedback on CPLD functioning. By building the entire development system on a breadboard, you can change it easily, connecting other chips or components for various projects.

Figure 3 illustrates the basic system. The PLDshell software and pro-

- state-transition statements that you provide using the PLDasm HDL.

The PENGN downloader program converts the JEDEC file into a configuration bitstream and sends it out through the PC printer port. A 25-pin male D-subminiature connector, 26-wire cable, 26-pin socket, and a 26-pin header carry the JEDEC bitstream to a 3M breadboard that holds the development system hardware.

From the header, the clocking signal for the bitstream (TCK) passes through two 74LS14 Schmitt-trigger inverters to prevent erroneous pulses brought on by slow signal transitions on the printer port.

The TMS signal, which controls the state of the EPX780 downloading process, and the TDO signal, which carries status information back to the PC, are also buffered. The TDI signal that carries the actual circuit configuration information from the PC to the '780 doesn't need to be buffered. From the 74LS14, the bitstream is sent to the EPX780.

The other printer port outputs are attached to the EPX780's general-purpose I/O

pins so you can apply test signals. An adapter socket matches the 84-pin PLCC of the EPX780 to the 0.1" pin spacing of the breadboard. A 7-segment LED attached to the CPLD's seven general-purpose I/O pins provides feedback during design debugging.

Current-limiting resistors prevent overloading of the CPLD outputs.

```

DOS/4GW Protected Mode Run-time Version 1.92

INFO PENGN: Interpreting file: fx780.apl

Target Device ID = 10621020h
Device Status Reg = 00001b 0K

Reading from FX780's SRAM
 1 :.....
11: .....
21: .....
31: .....
41: .....
51: . . . .
54: Done

Writing JEDEC test.jed
 0: .....
5120: . . . *.....
10240: . . . . .
15360: . . . . .
20480: . . . . .
25600: . . . . .
30720: . . . . .
31704: Done

```

Figure 4—The PENGN CPU-downloading program provides basic operational feedback.

programming manual can be obtained at no cost from Altera, while on-line software is available from XESS. The software provides a device fitter that generates a JEDEC file for the EPX780 based on:

- truth tables
- Boolean equations

Some electrolytic and nonpolarized bypass capacitors are sprinkled around to prevent noise from interfering with the system. The details of the wiring on the breadboard can be seen in Figure 2.

You should notice several details in the wiring. First of all, the printer port pins that carry the TMS and TDI signals during CPLD downloading also apply signals to the general-purpose I/O pins during debugging. This arrangement is permissible since arbitrary values on these pins cannot send the EPX780 back into the downloading mode.

However, the printer port pin carrying the TCK signal cannot be used during debugging because pulses on this output may cause the EPX780 to return to downloading mode and erase the design being tested. So, you can use only seven of the eight printer-port outputs to apply inputs to the EPX780.

Also, never create a design that uses pins 47, 48, 49, 50, 51, 77, or 78 as outputs. These outputs might conflict with printer-port outputs.

Second, pins 9 and 12 of the printer port must be shorted together. The PENG software uses this connection to test for the attachment of the downloading cable to the printer port. Pin 9 can apply test signals during design debugging because PENG is not active at that time.

Third, note that pin 26 of the 2 x 13 header is left unconnected since the printer port only has 25 pins. Pin 1 of the header connects to pin 1 of the printer port, pin 2 to pin 2, and so on. A straight run of 26-wire flat cable between the male D-subminiature connector and the 26-pin socket, which mates to the 26-pin header, should ensure this.

You've now wired the breadboard, attached the printer-port-downloading cable between the printer port and the breadboard, and connected a 5-V power supply to the breadboard. It's time to test your system.

## TESTING THE SYSTEM

Let's assume you installed PLD-shell on the C: drive of your PC in a directory called `P L D S H E L L` and put it

**Listing 1—The PLDasm code for a simple PLD circuit decodes a 4-bit number into seven signals that drive a 7-segment LED display.**

```
CHIP    leddecod    IFX780_84
; Inputs and outputs for the LED decoder
; the 4-bit input to the LED decoder
PIN    47    d0    ; least-significant bit (LSB)
PIN    48    d1
PIN    49    d2
PIN    50    d3    ; most-significant bit (MSB)
PIN    51    unused0
PIN    77    unused1
PIN    78    unused2

; * pins driving the LED segments
PIN    34    s0    ; +----s6----+
PIN    35    s1    ; |                |
PIN    36    s2    ; s5                s4
PIN    37    s3    ; |                |
PIN    39    s4    ; +----s3----+
PIN    40    s5    ; |                |
PIN    41    s6    ; s2                s1
                    . +----s0----+

; the truth table for driving the LEDs given the 4-bit
; number. A 1 on an output makes the corresponding LED
; segment light up; a 0 will make the segment stay dark.
; The truth table gives the appropriate outputs to light
; the LED segments for the digits 0-9.
T_TAB( d3 d2 d1 d0 >> s0 s1 s2 s3 s4 s5 s6 )
0 0 0 0 : 1 1 1 0 1 1 1 ; 0
0 0 0 1 : 0 1 0 0 1 0 0 ; 1
0 0 1 0 : 1 0 1 1 1 0 1 ; 2
0 0 1 1 : 1 1 0 1 1 0 1 ; 3
0 1 0 0 : 0 1 0 1 1 1 0 ; 4
0 1 0 1 : 1 1 0 1 0 1 1 ; 5
0 1 1 0 : 1 1 1 1 0 1 1 ; 6
0 1 1 1 : 0 1 0 0 1 0 1 ; 7
1 0 0 0 : 1 1 1 1 1 1 1 ; 8
1 0 0 1 : 1 1 0 1 1 1 1 ; 9

; Simulate the LED decoder
SIMULATION
TRACE-ON d0 d1 d2 d3 s0 s1 s2 s3 s4 s5 s6
SETF /d3 /d2 /d1 /d0 ; digit = "0"
SETF /d3 /d2 /d1 d0 ; digit = "1"
SETF /d3 /d2 d1 /d0 ; digit = "2"
SETF /d3 /d2 d1 d0 ; digit = "3"
SETF /d3 d2 /d1 /d0 ; digit = "4"
SETF /d3 d2 /d1 d0 ; digit = "5"
SETF /d3 d2 d1 /d0 ; digit = "6"
SETF /d3 d2 d1 d0 ; digit = "7"
SETF d3 /d2 /d1 /d0 ; digit = "8"
SETF d3 /d2 /d1 d0 ; digit = "9"
```

in your `PATH` variable. The PENG program should be in this directory. The following command tests the general health of your breadboard and the EPX780:

```
C:\> PENG -PART IFX780_84
-PORT 1 -LOC 0 -RS TEST.JED
```

where

- **PART** IFX780\_84 specifies that the chip you're working with is the EPX780 CPLD in an 84-pin PLCC package
- **PORT 1** specifies that the communications between PC and breadboard go through the LPT1 parallel port. Depending on the type of PC you're using, other valid options are - **PORT 2** and - **PORT 3**

- LOC 0 specifies that the EPX780 on the breadboard occupies location 0 in the chain of JTAG devices. Since the breadboard only has one EPX780 CPLD, this is the only reasonable value for this option.
- -RS TEST. JED specifies that PENGN should read the configuration data from the SRAM of the EPX780 and store it in TEST. JED on the PC. It doesn't really matter what's in the SRAM. This operation just exercises the communication paths between the PC and the breadboard.

If your PC screen looks like Figure 4, great! If it looks slightly different but there are no messages containing the word ERROR, you are probably OK - you might be using a newer version of PENGN with different diagnostic messages. In either case, you now possess a working breadboard that can be programmed from the PC to build many types of logic designs.

If you see a message containing the word ERROR, look up the error code

number in the Altera documentation for the PENGN program. Here are some common errors:

- power to the breadboard is turned off
- the cable between the breadboard and printer port is not connected
- pins 9 and 12 of the header are not connected
- the cable between the breadboard and the printer port is not built correctly. Use an ohmmeter to make sure pin 1 of the 26-pin socket is connected to pin 1 of the 25-pin D-subminiature connector.
- the wrong printer port number is used in the PENG N command. If you don't know your printer-port number, try them all: 1, 2, and 3.
- the cable between breadboard and printer port is too long. Cables up to 6' have been used successfully, but a shorter cable is better.
- the 74LS14 is bad

Once you've verified that the breadboard is working, it's time to build a real design.

## A CPLD DESIGN

I'll use a simple LED decoder to demonstrate how to use the CPLD development system. Start by activating the PLDshell programming environment:

```
C: \> PLDSHELL
```

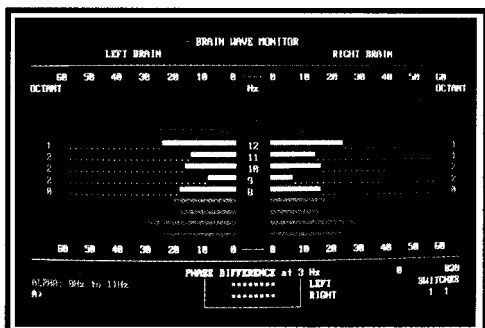
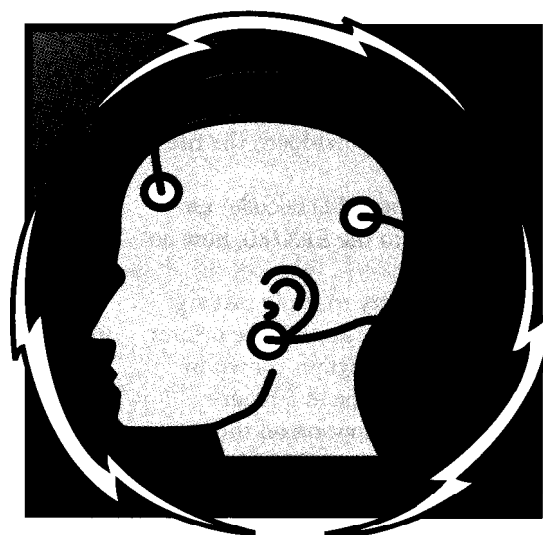
Once in PLDshell, you can use its built-in text editor. For an LED decoder, enter the PLDasm HDL code shown in Listing 1. Notice that the inputs come from the pins of the EPX780 that are attached to the printer port (47, 48, 49, and 50). This procedure tests the LED decoder by passing logic signals to it through the printer port. Notice also that pins 51, 77, and 78 are also declared, even though they are not used, to prevent the PLDshell device fitter from inadvertently assigning outputs to them.

The outputs of the LED decoder are assigned to the pins of the EPX780 that connect to the 7-segment LED. If you don't explicitly specify these pin assignments, the PLDshell device

# HAL - 4

## EEG Biofeedback Brainwave Analyzer

The HAL-4 kit is a complete battery-operated 4-channel electroencephalograph (EEG) which measures a mere 6" x 7". HAL is sensitive enough to even distinguish different conscious states-between concentrated mental activity and pleasant daydreaming. HAL gathers all relevant alpha, beta, and theta brainwave signals within the range of 4-20 Hz and presents it in a serial digitized format that can be easily recorded or analyzed. HAL's operation is straightforward. It samples four channels of analog brainwave data 64 times per second and transmits this digitized data serially to a PC at 4800 bps. There, using a Fast Fourier Transform to determine frequency, amplitude, and phase components, the results are graphically displayed in real time for each side of the brain.



**HAL-4 KIT . . . . . NEW PACKAGE PRICE - \$279 +SHIPPING**  
 Contains HAL-4 PCB and all circuit components, source code on PC diskette, serial connection cable, and four extra sets of disposable electrodes.

to order the HAL-4 Kit or to receive a catalog,  
**CALL: (860) 8752751 OR FAX: (860) 872-2204**

**CIRCUIT CELLAR KITS • 4 PARK STREET  
 SUITE 12 • VERNON • CT 06066**

\*The Circuit Cellar Hemispheric Activation Level detector is presented as an engineering example of the design techniques used in acquiring brainwave signals. This Hemispheric Activation Level detector is not a medically approved device, no medical claims are made for this device, and it should not be used for medical diagnostic purposes. Furthermore, safe use requires HAL be battery operated only!

fitter is free to assign these outputs to any of the 80 CFB outputs in the EPX-780. In this case, that's probably not what you want.

The actual operation of the decoder is specified using a truth table. PLDshell derives the appropriate Boolean equations from this table for you.

Finally, you can embed simulation instructions in the PLDasm file. These are used by the simulator built into PLDshell. The simulator lets you observe the functioning of your design before downloading it to the EPX780 for final in-circuit testing.

Once you enter the PLDasm code, you can activate the COMPILE menu option in PLDshell to create a JEDEC file. (If the PLDasm file is called **LED - DECOD. PDS**, the JEDEC file is **LED - DEC 0 D . J E D**.) Download the JEDEC file into the breadboard using the command:

```
C:\> PENGN -PART IFX780_84
-PORT 1 -LOC 0 -PS LEDDECOD.JED
```

This command is identical to the PENGN command we looked at earlier, except for the **- PS** option. This option programs the static RAM of the EPX780 with the circuit configuration stored in the **LEDDECOD. JED** file. PENGN prints various progress reports on the screen as it downloads the file into the EPX780.

Now that your LED decoder circuit is loaded into the EPX780, how do you test it?

You need a way to apply test signals to the EPX780 through the printer port. The DOS C program shown in Listing 2 lets you type in a binary string which then appears on the printer-port outputs. If you place this code a file called **PORT .C** and compile it, you can make your LED decoder display a "6" by typing the command:

```
C:\> PORT 0000110
```

Your LED decoder responds to the lower four bits of any binary string you pass to the PORT program. (The truth table for the LED decoder is defined only for the numbers "0-9" so you need to extend it to display "A-F" in hexadecimal.)

Listing 2—This simple program lets you apply signals to the CPLD from the PC printer port.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define OUTPORT 0x378 /* printer port address: try 0x278 or */
/* 0x3BC if this doesn't work */
main(int argc, char **argv)
{
    char bits[50]; /* storage for user's binary string */
    int i;
    int port_val; /* value output on printer port */
    int bit-mask; /* mask for bits in port_val */

    sscanf(argv[1], "%s", bits); /* get binary string */
    port_val = 0; /* start with all port bits set to zero */
    bit-mask = 2; /* start with second LSB. The LSB is the
/* TCK and we want to leave that alone. */
/* now start from the end of the user's binary string and */
/* set the bits of port_val that correspond to 1s */
    for (i=strlen(bits)-1; i>=0; i--)
    {
        switch (bits[i])
        {
            case '0': break; /* bit is already zero */
            case '1': port_val |= bit-mask; /* set bit */
                    break;
            default: fprintf(stderr, "ERROR\n");
                    break;
        }
        bit-mask <<=1; /* shift bit-mask to next printer port bit */
    }
    /* finally, output port_val through the printer port */
    outp(OUTPORT, port_val);
}
```

## A MORE COMPLEX CPLD DESIGN

The PLDasm code in Listing 3 combines the LED decoder circuit with a 3-bit counter to build a simple incrementing display. It begins by assigning the inputs and outputs of the combined counter and LED decoder.

For this design, only one input must be driven from the printer port: the clock input that makes the counter change state. I used pin 47 of the EPX-780 because the printer-port output that drives it passes through two Schmitt-trigger inverters. So, the signal should be pretty clean. No other inputs from the printer port are used in this example.

The outputs of the counter (**d 0**, **d 1**, and **d2**) are also the inputs to the decoder. They are not assigned to specific pins of the EPX780, so the PLDshell device fitter assigns them to whatever pins it chooses.

As in the last example, you must specifically assign LED decoder outputs to pins connected to the LED

digit. The example uses vector notation to assign ranges of signal names to ranges of pin numbers.

The truth table for the LED decoder comes after the pin assignments. In Listing 3, the decoder is simplified so that it responds only to 3-bit codes corresponding to the digits 0-8.

The 3-bit counter, described next, uses a simple Moore state machine. Each state is assigned a digital code corresponding to the next digit in the sequence. The default transition between states is used so that the next state in the assignment list becomes the current state when the next clock pulse arrives. The counter increments digital values until the counter rolls over from 111 to 000.

The **EQUATIONS** section defines how the clock connects to the counter flip-flops. Each of the three flip-flops changes state on the rising edge of the clock signal.

Compile and download the circuit as you did in the previous example.

Listing 3—The PLDasm code for a 3-bit up counter displays the current count on a 7-segment LED display.

```
CHIP upcnt IFX780_84
; Inputs and outputs for the counter and display
; the 4-bit input to the LED decoder
PIN 47 clock ; clock signal for counter
PIN 48 unused0 ; unused inputs from printer port
PIN 49 unused1
PIN 50 unused2
PIN 51 unused3
PIN 77 unused4
PIN 78 unused5
PIN d[2:0] ; 3-bit counter outputs
PIN [37:34] s[3:0]; LED decoder outputs assigned to pins
PIN [41:39] s[6:4]; connected to the LED digit
; LED decoder truth-table shortened to 0-7 codes
T_TAB( d2 d1 d0 >> s0 s1 s2 s3 s4 s5 s6 )
0 0 0 : 1 1 1 0 1 1 1 ; 0
0 0 1 : 0 1 0 0 1 0 0 ; 1
010 : 1 0 1 1 1 0 1 ; 2
0 1 1 : 1 1 0 1 1 0 1 ; 3
100 : 0 1 0 1 1 1 0 ; 4
1 0 1 : 1 1 0 1 0 1 1 ; 5
110 : 1 1 1 1 0 1 1 ; 6
111 : 0 1 0 0 1 0 1 ; 7
; state-transition description of a 3-bit counter
STATE MOORE-MACHINE
; when a clock pulse occurs, move to the next state
; in the sequence s0->s1->s2->...->s7->s0.
DEFAULT-BRANCH NEXT-STATE
; the state assignments follow
s0 = /d2 * /d1 * /d0; s0 = 000
s1 = /d2 * /d1 * d0; s1 = 001
s2 = /d2 * d1 * /d0; s2 = 010
s3 = /d2 * d1 * d0; s3 = 011
s4 = d2 * /d1 * /d0; s4 = 100
s5 = d2 * /d1 * d0; s5 = 101
s6 = d2 * d1 * /d0; s6 = 110
s7 = d2 * d1 * d0; s7 = 111
EQUATIONS
; clock the counter flip-flops on the rising edge
d[2:0].ACLK = clock
```

You can then increment the value displayed on the LED digit of your breadboard by sending a clock pulse to the printer port using:

```
C:\> PORT 0000001
C:\> PORT 0000000
```

You can see the entire counter sequence by repeating this set of commands eight times.

## AND THERE'S MORE!

The examples given here only scratch the surface—there's much more you can do with this CPLD development system.

For example, I've used it to design a 4-bit micro that fits entirely in a single EPX780 CPLD (including the

program and data memory). The example file stored on XESS's FTP site contains a set of PLDasm design files that demonstrate some other things you can do with this system.

There are a lot of benefits to using a CPLD to build digital designs. Using a CPLD means you can:

- build designs faster because manual wiring is minimized
- avoid wiring mistakes (which you replace with typing mistakes)
- experiment with many types of digital designs without having to buy more chips
- save your designs in files on your PC and recall them whenever you want
- reuse and modify old designs to build new projects

- let other people use your designs by simply providing a copy of the PLDasm file.

A simple CPLD development system like this won't do everything for you, but it's a handy item to keep in your toolbox for rapid prototyping and testing of digital designs. ☐

*After working at both Bell Laboratories and North Carolina State University, Dave Van den Bout now works at XESS Corporation as a developer of LINUX-compatible software and FPGA-based computing products. He may be reached at (919) 387-1302 or devb@xess.com.*

## SOURCES

### EPX780 CPLD

Wyle Laboratories  
15370 Barranca Pkwy.  
Irvine, CA 92718  
(714) 753-9953

### PLDshell

Altera Corp.  
2610 Orchard Pkwy.  
San Jose, CA 95134-2020  
(408) 894-7144

### XESS Corp.

2608 Sweetgum Dr.  
Apex, NC 27502  
(919) 387-0076  
Fax: (919) 387-1302  
<ftp://ftp.vnet.net/pub/users/xess/PLDshell/pldsh.zip>  
<http://www.xess.com/>

### PLDs

JDR Microdevices  
1850 South 10th St.  
San Jose, CA 95112-4108  
(408) 494-1420

### Digi-Key Corp.

P.O. Box 677  
Thief River Falls, MN 56701  
(800) 344-4539  
Fax: (218) 681-3380

## IRS

404 Very Useful  
405 Moderately Useful  
406 Not Useful

# Take Your PIC

## FEATURE ARTICLE

Fred Eady

### A Look at the PIC16Cxx Family

After taking a close look at the common architecture of the PIC16Cxx family, Fred details each subfamily's unique features. He wraps up his article by touching on how to program PICs.



**Y**ou've seen these amazing devices used everywhere for almost everything—from generating complex video signals to controlling motors of all kinds. This versatile device, originally designed as a Peripheral Interface Controller, is now known as a PIC.

Although many of you have already created some marvelous products with PICs, some of you probably still see this device as a 6" stack of data books with a project waiting to happen. The truth is, however, PIC devices are powerful and easy to use.

The great thing about the PIC family is that if you understand one device, you can easily move from one PIC to another with little difficulty. The key to success lies in understanding the PIC's basic architecture.

In this article, I'll compare the 12-bit baseline and 14-bit midrange PIC families. I'm hoping to present a logi-

cal view of the PIC so that you can make better use of that 6" stack of data books.

Let's start with common PIC architecture.

### PIC16Cxx ARCHITECTURE

The baseline PIC16C5x family consists of the PIC16C54, '55, '56, '57, and the new '58. All five of this family's members are low-cost, 8-bit, EPROM-based CMOS microcontrollers as are the midrange PIC16C6x parts. In the PIC16C6x family, there are eight members: the PIC16C61-'65 and the PIC16C620-'622. The PIC16C7x (PIC16C71, '73, and '74) indicates on-chip A/D conversion. The '8x means the part is EEPROM based. Currently, there is only one device that meets that criteria—the PIC16C84.

However, don't let the term "baseline" fool you, and don't think for a minute that the PIC16C5x devices are inferior. The internal PIC architecture is common across baseline and midrange parts. If your application doesn't need interrupts or special-purpose on-chip peripherals, the PIC16C5x parts perform with the efficiency characteristic of other PIC devices.

When you get right down to it, the core operations of both the midrange and baseline devices are virtually identical. From the view of the programmer, only the specialized on-chip features implemented in the register stack differentiate the devices. Table 1 provides features of the PIC16C5x

	Clock	Memory	Peripherals	Features					
	Maximum frequency of operation (MHz)	Program memory (words)	RAM data memory (bytes)	Timer module(s)					
	EPROM	ROM	I/O pins	Voltage range (volts)					
				Number of instructions					
				Packages					
PIC16C54	20	512	—	25	TMR0	12	2.5–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C54A	20	512	—	25	TMR0	12	2.5–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16CR54	20	—	512	25	TMR0	12	2.0–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C55	20	512	—	25	TMR0	20	2.5–6.25	33	28-pin DIP, 28-pin SOIC, 28-pin SSOP
PIC16C56	20	1K	—	25	TMR0	12	2.5–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C57	20	2K	—	72	TMR0	20	2.5–6.25	33	28-pin DIP, 28-pin SOIC, 28-pin SSOP
PIC16C57A	20	—	2K	72	TMR0	20	2.0–6.25	33	28-pin DIP, 28-pin SOIC, 28-pin SSOP
PIC16C58A	20	2K	—	73	TMR0	12	2.5–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16CR58A	20	—	2K	73	TMR0	12	2.0–6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP

Note: All PIC16/17 family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect, and high I/O current capability.

Table 1—The robust baseline PICs can run at 20 MHz over a wide voltage range.

	Clock				Memory				Peripherals						Features		
	Maximum frequency of operation (MHz)	EPROM	EEPROM	Program memory	Data memory (bytes)	Data EEPROM (bytes)	Timer module(s)	Capture/comparator/PWM module(s)	Serial port(s) (SPI/I <sup>2</sup> C)	Parallel slave port	Analog-to-digital converter (8 bit)	Comparator(s)	Internal reference voltage	Interrupt sources	I/O pins	Voltage range (volts)	Brown-out
PIC16C61	20	1K	—	36	—	—	TMR0	—	—	—	—	—	3	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C62	20	2K	—	128	—	—	TMR0	1	SPI/I <sup>2</sup> C	—	—	—	7	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC, 28-pin SSOP
PIC16C63	20	4K	—	192	—	—	TMR0	2	SPI/I <sup>2</sup> C/SCI	—	—	—	10	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C64	20	2K	—	128	—	—	TMR0	1	SPI/I <sup>2</sup> C	yes	—	—	8	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C65	20	4K	—	192	—	—	TMR0	2	SPI/I <sup>2</sup> C/SCI	yes	—	—	11	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C620	20	512	—	80	—	—	TMR0	—	—	—	2	yes	4	13	3.0-6.0	yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C621	20	1K	—	80	—	—	TMR0	—	—	—	2	yes	4	13	3.0-6.0	yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C622	20	2K	—	128	—	—	TMR0	—	—	—	2	yes	4	13	3.0-6.0	yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C71	20	1K	—	36	—	—	TMR0	—	—	—	4 ch	—	4	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C73	20	4K	—	192	—	—	TMR0	2	SPI/I <sup>2</sup> C/SCI	—	5 ch	—	11	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C74	20	4K	—	192	—	—	TMR0	2	SPI/I <sup>2</sup> C/SCI	yes	8 ch	—	12	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C84	10	—	1K	36	64	—	TMR0	—	—	—	—	—	4	13	2.0-6.0	—	18-pin DIP, 18-pin SOIC

Note 1: All PIC16/17 family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect, and high I/O current capability.  
 2: The PIC16CXX Timer1 has its own oscillator circuit and can operate asynchronously to the device. Timer1 can increment while the device is in SLEEP mode. This allows a Real Time Clock to be implemented.  
 3: PORTB has software-configurable weak pull-ups.

Table 2-A rich set of on-chip peripherals makes the midrange PICs ideal for more complex designs.

devices while Table 2 presents the PIC16Cxx parts.

There are only 33 assembler instructions associated with the PIC-16C5x family and 35 instructions for the PIC 16Cxx devices. Most instructions execute within a single processor cycle with the exception being program-branch instructions, which take two cycles to complete. Each PIC-16C5x instruction word is 12 bits in length with the mnemonic (the opcode) and operand (the register, memory location, or direct data to be manipulated) fully defined within the 12-bit word. The PIC 16Cxx instructions are logically identical but are 14 bits in length.

In reviewing Table 3, the PIC-16C5x instruction set, and Table 4, its 14-bit counterpart, note that from the view of a programmer, the PIC16C5x and PIC16Cxx instruction sets differ only in the literal and control operations area. This variance is due to the added functionality found in the 14-bit devices. Most of the specialized hardware has an associated set of special registers that are manipulated via the instruction set. These registers eliminate the need for different instructions for every individual peripheral.

Most of the baseline and midrange PICs operate with clock speeds ranging from DC to 20 MHz, except for the PIC16C84 which checks in at 10 MHz max. At 20 MHz, the instruction cycle time is 200 ns. Most traditional microcontrollers operate at much lower clock speeds with microsecond cycle times and use instructions that consume multiple bytes of program space per instruction. The PIC's high-speed execution, coupled with the code efficiency offered in the single-word instruction set, boosts performance a magnitude above almost every micro in its class.

The PIC's high microcode execution speed is attained by using Harvard architecture, or the Harvard dual-bus concept, instead of the classic Von Neumann, or single-bus, implementation. Harvard architecture is register file based with a separate bus and memory space allocated for instructions and data. The term "register file based" simply means that all program-controlled objects such as I/O ports, memory locations, and timers are physically implemented as hardware registers.

The PIC16C5x and PIC16Cxx data memory (RAM) bus is 8 bits wide

while the program memory (EPROM) bus is 12 and 14 bits. Using the Harvard dual-bus configuration enables the PIC family to perform high-speed bit, byte, and register operations.

Harvard architecture also inherently overlaps instruction execution. This overlapping of instruction-execution cycles is known as *pipelining*, which is the simultaneous execution of the current instruction as the next instruction is being read from program memory. Traditional Von Neumann architecture fetches instruction and data information over a single shared or multiplexed bus, thereby eliminating the ability to overlap instruction fetch and execution, Figure 1 gives us a physical look at how pipelining is performed within the dual-bus PIC.

## THE REGISTER-FILE CONCEPT

As I mentioned earlier, all PIC program objects are actually implemented as physical registers. Let's begin by looking at the Operational Register Files.

These registers are common to all of the baseline and midrange devices. This collection of registers contains a means for indirect data addressing, real-time clock/counter, program



counter, status word register, file select register, and I/O registers.

f0, the indirect data addressing register (INDF), is not physically implemented. f0 uses the contents of f4, the File Select Register (FSR), to indirectly select any one of the available 32 file registers for a data or pointer register depending on the intent of the instruction that called f0. Listing 1 offers an example.

f1 (or RTCC or TMRO) is read and written just like any other register. It can also be incremented by an external signal applied to the TOCKI pin or by the internal instruction clock. TMRO can also be prescaled using the internal programmable prescaler. TMRO increments as long as clock is applied to it. When FF is reached, TMRO rolls over to 00 and continues counting.

f2, the Program Counter (PCL), generates addresses for EPROM cells containing the user-written program-instruction words. The PC is 9-13 bits wide, depending on the type of PIC. This register depicts the low-order 8 bits of the PC only.

f3, the Status Word Register, contains the arithmetic status of the ALU (carry bit, zero bit, etc.), reset status, and page preselect bits for the larger program memories. f3 is comparable to the PSW (Program Status Word) found in most other microprocessors. Power-down and timeout bits used by the Watchdog Timer (WDT) and sleep instructions are also held in f3.

As previously noted, f4 is the FSR and is used in conjunction with f0 to indirectly select available file registers. If no indirect calls are used in the user's program, this register can serve as a general-purpose register.

f5-f7 are I/O registers for ports A, B, and C, respectively. These registers can be read and written just like any other registers in the register file and are capable of having related I/O pins placed in high-impedance state for isolation or read operations. Any I/O pin can be independently programmed for input, output, or bidirectional operation via the TRIS registers. A binary 1 in a TRIS register bit position corresponds to high impedance or input mode, while a binary 0 gives output of that bit position to the related I/O pin.

Mnemonic	Operands	Description	Cycles	12-bit Opcode		Status Affected	
				MSb	LSb		
ADDWF	f,d	Add W and f	1	0001	11df	ffff	C, DC, Z
ANDWF	f,d	AND W with f	1	0001	01df	ffff	Z
CLRF		Clear f	1	0000	011f	ffff	Z
CLRWF		Clear W	1	0000	0100	0000	Z
COMF	f,d	Complement f	1	0010	01df	ffff	Z
DECf	f,d	Decrement f	1	0000	11df	ffff	Z
DECFSZ	f,d	Decrement f, skip if 0	1(2)	0010	11df	ffff	None
INCF	f,d	Increment f	1	0010	10df	ffff	Z
INCFSZ	f,d	Increment f, skip if 0	1(2)	0011	11df	ffff	None
IORWF	f,d	Inclusive OR W with f	1	0001	00df	ffff	Z
MOVF	f,d	Move f	1	0010	00df	ffff	Z
MOVWF		Move W to f	1	0000	001f	ffff	None
NOP		No Operation	1	0000	0000	0000	None
RLF	f,d	Rotate left f through Carry	1	0011	01df	ffff	C
RRF	f,d	Rotate right f through Carry	1	0011	00df	ffff	C
SUBWF	f,d	Subtract W from f	1	0000	10df	ffff	C, DC, Z
SWAPF	f,d	Swap f	1	0011	10df	ffff	None
XORWF	f,d	Exclusive OR W with f	1	0001	10df	ffff	Z
Bit-Oriented File Register Operations							
BCF	f,b	Bit clear f	1	0100	bbbf	ffff	None
BSF	f,b	Bit set f	1	0101	bbbf	ffff	None
BTFSC	f,b	Bit test f, skip if clear	1(2)	0110	bbbf	ffff	None
BTFSS	f,b	Bit test f, skip if set	1(2)	0111	bbbf	ffff	None
Literal and Control Operations							
ANDLW	k	AND literal with W	1	1110	kkkk	kkkk	Z
CALL	k	Call subroutine	2	1001	kkkk	kkkk	None
CLRWDT	k	Clear watchdog timer	1	0000	0000	0000	TO, 'PD
GOTO	k	Unconditional branch	2	101k	kkkk	kkkk	None
IORLW	k	Inclusive OR literal with W	1	1101	kkkk	kkkk	Z
MOVLW	k	Move literal to W	1	1100	kkkk	kkkk	None
OPTION	k	Load OPTION register	1	0000	0000	0000	None
RETLW	k	Return, place literal in W	2	1000	kkkk	kkkk	None
SLEEP	—	Go in standby mode	1	0000	0000	0011	TO, 'PD
TRIS	f	Load TRIS register	1	0000	0000	0fff	None
XORLW	k	Exclusive OR literal to W	1	1111	kkkk	kkkk	Z

Table 3—The PIC16C5x instruction set is easy to learn as it consists of only 33 mnemonics.

PICs using 18-pin packages do not have a physical C port.

The second set of registers, known as the General-Purpose Registers, is

most commonly used as internal user RAM available for program variable storage. The number of these registers depends on the type of PIC.

Mnemonic	Operands	Description	Cycles	14-bit Opcode		Status Affected		
				MSb	LSb			
ADDWF	f,d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z
ANDWF	f,d	AND W and f	1	00	0101	dfff	ffff	Z
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z
CLRWF		Clear W	1	00	0001	0xxx	xxxx	Z
COMF	f,d	Complement f	1	00	1001	dfff	ffff	Z
DECf	f,d	Decrement f	1	00	0011	dfff	ffff	Z
DECFSZ	f,d	Decrement f, skip if 0	1(2)	00	1011	dfff	ffff	Z
INCF	f,d	Increment f	1	00	1010	dfff	ffff	Z
INCFSZ	f,d	Increment f, skip if 0	1(2)	00	1111	dfff	ffff	—
IORWF	f,d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z
MOVF	f,d	Move f	1	00	1000	dfff	ffff	Z
MOVWF	f,d	Move W to f	1	00	0000	1fff	ffff	Z
NOP	—	No Operation	1	00	0000	0xxx	0000	None
RLF	f,d	Rotate left through carry	1	00	1101	dfff	ffff	C
RRF	f,d	Rotate right f through carry	1	00	1100	dfff	ffff	C
SUBWF	f,d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z
SWAPF	f,d	Swap nibbles in f	1	00	1110	dfff	ffff	Z
XORWF	f,d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z
Bit-Oriented File Register Operations								
BCF	f,b	Bit clear f	1	01	00bb	bfff	ffff	None
BSF	f,b	Bit set f	1	01	01bb	bfff	ffff	None
BTFSC	f,b	Bit test f, skip if clear	1(2)	01	10bb	bfff	ffff	None
BTFSS	f,b	Bit test f, skip if set	1(2)	01	11bb	bfff	ffff	None

Table 4—The PIC16Cxx instruction set is logically identical to the PIC16C5x instruction set and consists of two additional instructions that provide access to the interrupt capability of the midrange devices.

Literal and Control	Operations				
ADDLW	k	Add literal to W	1	1111x kkkk kkkk	C, DC, Z
ANDLW	k	AND literal to W	1	11 1001 kkkk kkkk	Z
CALL	—	Call subroutine	2	10 0kxx kkkk kkkk	
CLRWDT	—	Clear watchdog timer	1	00 0000 0110 0100	'TO, 'PD
GOTO	k	Go to address	2	10 1kxx kkkk kkkk	—
IORLW	k	Inclusive OR literal to W	1	11 1000 kkkk kkkk	Z
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk	—
RETFIE	—	Return from interrupt	2	00 0000 0000 1001	
RETLW	k	Return with literal in W	2	11 0lxx kkkk kkkk	
RETURN	—	Return from subroutine	2	00 0000 0000 1000	
SLEEP	—	Go into standby mode	1	00 0000 0110 0011	*TO, 'PD
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C, DC, Z
XORLW	k	Exclusive OR literal to W	1	11 1010 kkkk kkkk	Z

Table 4—continued

Special-Function Registers (SFR) are dedicated to the CPU for the purpose of controlling a particular device. For instance, the CMCON register within the PIC 16C620 is manipulated by the CPU to control the comparators. All of the advanced 14-bit parts have SFRs that are associated with the specialized functions available on the silicon.

## NEW FEATURES

Now that you really know what a PIC is, let's look at some of the new

innovations, beginning with the PIC-16C61.

The PIC 16C61 is really a "bridge" part. It is the basic PIC16C5x part with enhanced features such as interrupt

capability and a separate RETURN instruction that operates on an eight-level hardware stack. The PIC16C5x parts don't have interrupts and sport a two-level hardware stack. This part is a direct descendant of the PIC16C71, without the A/D converter circuitry.

The PIC16C62, '63, '64, and '65 differ in program and data memory size and are loaded with goodies. The PIC16C62 and '63 boast 28-pin packages with 22 I/O lines while the 40-pin PIC16C64 and '65 parts offers 33 I/O lines. All four parts include capture input, compare output, and PWM output.

Other features include an additional timer (TMR1) that runs during sleep mode. This feature enables the

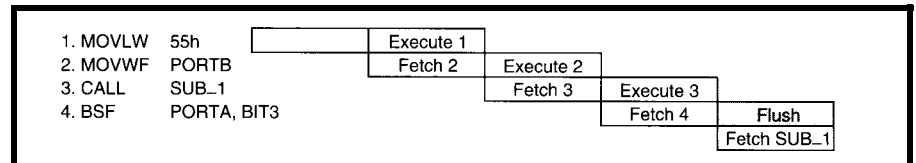


Figure 1—All PIC devices employ an instruction pipeline technique that overlaps fetch and execution cycles. All instructions are single cycle, except for any program branches. These branches take two cycles since the fetch instruction is flushed from the pipeline while the new instruction is fetched and then executed.

# PIC ADAPTERS



In Stock  
\$60.-\$179.

Programming adapters for PIC chips  
SOIC, SSOP, QFP, or PLCC socket to DIP plug

**LOGICAL**  
SYSTEMS

(315) 478-0722 Tel (315) 479-6753 Fax  
73760.2723@compuserve.com  
PO. Box 6184, Syracuse NY 13217 USA

## PIC16Cxx C COMPILER

- Integrated software development environment including an editor with interactive error detection/correction. A DOS command line compiler is also included.
- Access to all PIC hardware features from easy to use C functions.
- Built in libraries for RS232 serial I/O (all chips) and precision delays are included.
- Includes example drivers for an LCD, keypad, Serial E<sup>2</sup> and real time clock.
- Efficient function implementation allows call trees deeper than the hardware slack.
- Features such as bit variables are optimized for the unique hardware capabilities.
- Functions that call one another frequently are grouped together in the same page and calls across pages are handled automatically by the twl transparent to the user.
- Assembly code may be inserted in the SOURCE and may reference C variables.
- Constants (including strings and arrays) are saved in program memory.
- Hex and debug file formats are readable by most programmers and emulators.

Complete example program to read A/D and send value to a PC via RS-232 I/O:

```
#include <PIC16C71.h>
#include <stdio.h>
#define Delay (Clock=1500000)
#define RS232 (Baud=9600, Xmit=PIN_10, Rcv=PIN_7)
#include <hex.c>

main() {
    int value;

    setup_port_a( ALL_ANALOG );
    setup_ADC( ADC_CLOCK_INTERNAL );
    set_ADC_channel( PIN_18 );

    printf("Sampling pin 18:\r\n");

    do {
        value = Read_ADC();
        printf("A/D Value: ");
        puthex(value);
        printf("\n\n");
        delay_ms(1000);
    } while (TRUE);
}
```

PCB compiler \$99 (all 5x chips)  
PCB compiler \$99 (61-84 chips)  
Shipping, COD \$10 (2 day, USA Only)

CCS, PO Box 2452  
Brookfield, WI 53008  
414-781-2794 x30

implementation of a real-time clock. In addition, a third timer (TMR2), an 8-bit-wide parallel slave port, and support for SPI and I<sup>2</sup>C are common to the PIC 16C64 and '65 devices. The PIC-16C63 and '65 extend this list of features by adding a pin that can be configured for capture input, PWM output, compare output, and an on-chip USART.

Program memory for the PIC-16C62x ranges from 5 12 words in the PIC16C620 to 2K words in the PIC-16C622. Data memory is 80 and 128 bytes, respectively. The PIC16C62x devices are unique in that they each carry a set of on-chip analog comparators. I could talk pages about this, but Figure 2 is really what you need. The figure shows the eight possible comparator modes. Note that this set of parts also includes an on-chip programmable voltage reference.

Earlier I described the PIC 16C61 as a '71 without analog-to-digital functions. The PIC16C71 differentiates itself by including four channels of analog-to-digital conversion. This was one of the original 14-bit parts.

The PIC16C73 is a first cousin to the '63. It adds 4K words of program memory and an additional interrupt source and five channels of A/D conversion. This is also true for the eight-A/D-channel 40-pin PIC 16C74.

The PIC 16C84 was introduced shortly after the PIC16C71. The PIC-16C84 register file is almost identical to that of the PIC16C71, with the exception of the special registers, which let you use the EEPROM data memory. The PIC 16C84 has 64 8-bit data EEPROM cells that can be read and written during normal operation.

When a byte is written to the EEPROM data area, microcode within the PIC16C84 automatically erases the location before writing the new data. Write cycle time is 10 ms and is controlled by an on-chip timer. The programmer can choose to poll a write complete bit or simply wait out the 10-ms period. Reading PIC16C84 user EEPROM data memory is accomplished in Listing 2a. As you can see in Listing 2b, writing PIC16C84 user EEPROM data memory is a bit more involved but no real problem.

Listing 1—The five simple instructions in this code snippet use indirection to add the contents of register 8 to the working register W. At 20 MHz, this takes only 1 μs.

```

; Initialize FSR with 08h
movlw 0x08    ; Load W with 08h
movwf fsr    ; Load f4 with 08h

; Load register 8 with 09h
movlw 0x09    ; Load W with 09h
movwf 8       ; Load register 8 with 09h

; Perform an indirect operation
addwf indf, w ; Add contents of register pointed to
              ; by the FSR to the contents of the
              ; W register and place the result
              ; in the W register
  
```

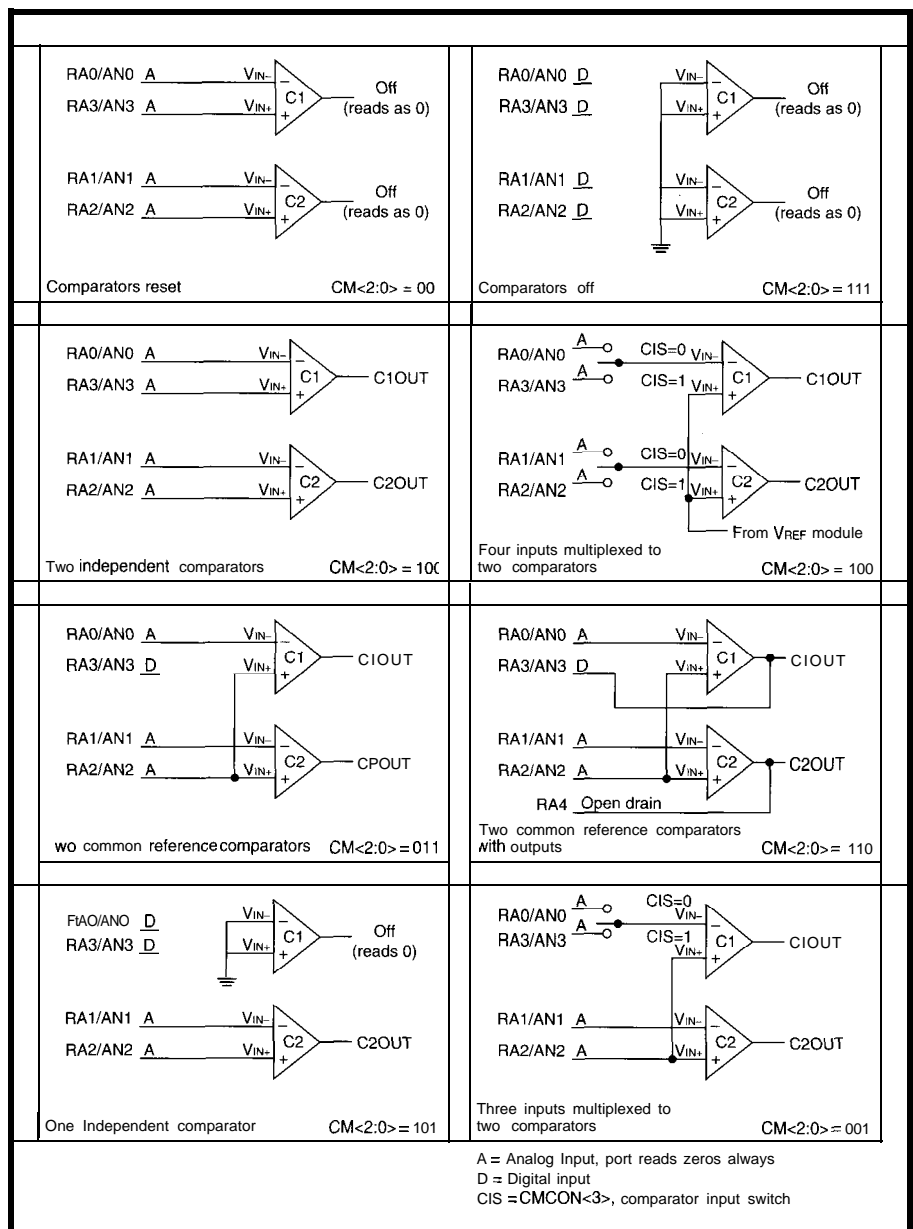


Figure 2—The PIC16C62x devices have eight modes of comparator operation and an on-chip programmable voltage reference.

The endurance of the user EE-PROM is typically 1,000,000 cycles with a data retention time in excess of 40 years. The PIC16C84 has 1 K words of program memory and reprogrammable EEPROM.

## PROGRAMMING PICs

The baseline devices are designed to be programmed in parallel mode. That entails presenting 12-bit words to the target PIC in addition to the other program control requirements. The newer midrange devices use what is called the Microchip in-system serial programming process. This technique only requires five connections:

- clock line
- data line
- Vdd
- ground
- +13-V programming voltage

This process enables subsystems to be assembled and programmed with a blank PIC16Cxx device onboard and in-circuit. The most recent level of

firmware can then be programmed into the product just before shipping it. This feature results in a simple and low-cost programming method. In-system serial programming specifications can be found in the Microchip data book.

The 14-bit core target PIC device is placed into program and verify mode by holding the RB6 and RB7 pins low while raising the MCLR pin from ground to +13 VDC. RB6 is the programming clock and RB7 transfers data during the programming process. A real-world example of this algorithm using a PIC 16C54 is available on the EDTP BBS.

## FURTHERHELP

Obviously, the PIC has evolved into a more powerful device that is finding its way into worlds normally confined to the traditional microcontroller paradigm. To learn more about the PIC, take a Microchip seminar in your area. I recently did the Florida tour with engineers from Atlanta, Georgia, and Chandler, Arizona. A

wealth of information is presented and you get one-on-one with real PIC specialists.

In addition to seminars and data books, Microchip offers a complete line of development tools including programmers and emulators. The company also supports a third-party market, consisting of consultants and businesses offering PIC-compatible software and development tools. You may find these third-party resources in the **Microchip Third-Party Guide**.

It's up to you to get the details you need for your application. Here are some manuals you should get:

- **Microchip Data Book**
- **Microchip Embedded Control Handbook**
- **Microchip Serial EEPROM Handbook**

The data book is essential as it contains all of the technical data for the full line of PICs. The embedded control handbook is a handy reference for beginning or experienced PIC users



Listing 2—EEPROM in the PIC16C84 provides *nonvolatile storage capability*. These code segments show how easy it is to read (a) and write (b).

```

a)  clrw           ; Clear W
     movwf        eeadr      ; Load address 0
     bsf          status,rp0 ; Go to page 1
     bsf          eecon1,rd  ; Do an EEPROM read from address 0
     bcf          status,rp0 ; Return to page 0
     movfw        eedata, w  ; Load W with EEDATA contents
     movlw        0x01      ; Load address of 0x01
     movwf        eeadr      ;
     bsf          status,rp0 ; Go to page 1
     bsf          eecon1,rd  ; Do an EEPROM read from address 1
     bcf          status,rp0 ; Return to page 0
     movfw        eedata, w  ; Load data just read into W

b)  writeaddr0
     clrw           ; Clear W
     movwf        eeadr      ; Load EEADR with address 0x00
     bsf          status,rp0 ; Select register bank for EECONX
     bsf          eecon1,wren ; Enable EEPROM write enable
     bcf          eecon1,eeif ; Make sure EEIF is clear

     movlw        0x55      ; This sequence must be performed
     movwf        eecon2    ; in this order to write to
     movlw        0xaa      ; EEPROM data memory
     movwf        eecon2    ;
     bsf          eecon1,wr  ;

writing0
     btfss        eecon1,eeif ; Check for end of write
     got0
     bcf          eecon1,eeif ; Clear EEIF before leaving
     bcf          eecon1,wren ; Disable EEPROM write enable
     bcf          status,rp0 ; Select register bank 0
; *** Your program continues here..

```

and contains various PIC application notes. The serial EEPROM handbook is a great source of information concerning the Microchip line of serial EEPROM devices.

The Microchip assembler and various other NC-related software products are available free of charge from the Microchip BBS. The Microchip BBS is a no-charge service that can be accessed through CompuServe. Details for connecting are contained within the pages of the **Microchip Data Book**. You may also get PIC application and development tool information by dialing the EDTP Reader Service BBS at (407) 454-3198. ☒

**Fred Eady is a registered Microchip third-party developer and has authored a number of PIC-related articles His company, E D Technical Publications, specializes in low-cost PIC development tools. Fred may be reached at edtp@ddi.digital.net.**

**All tables and charts are reprinted with permission from Microchip Technology. Information contained in these drawings and charts is intended for suggestion only and may be superseded by updates.**

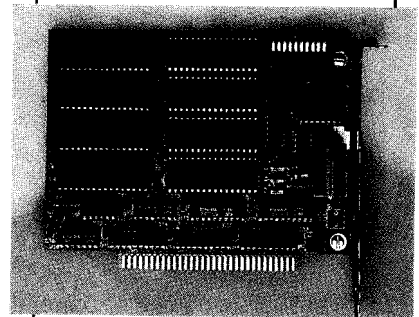
## CONTACT

Microchip Technology, Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224  
(602) 786-7200  
Fax: (602) 786-7277

E D Technical Publications  
P.O. Box 541222  
Merritt Island, FL 32954-1222  
(407) 454-9905  
Fax: (407) 454-9905

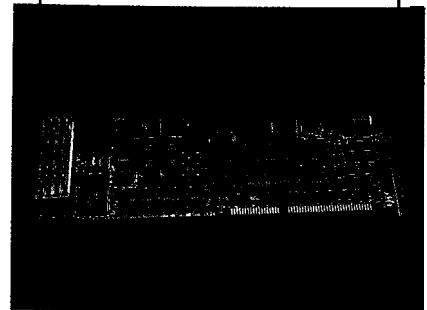
## IRS

407 Very Useful  
408 Moderately Useful  
409 Not Useful



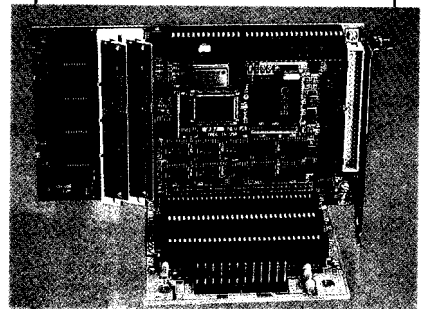
### ENHANCED SOLID STATE DRIVE — \$164\*

4M Total, Either Drive Bootable  
1/2 Card 2 Disk Emulator  
Flash System Software Included  
FLASH & SRAM, Customs too



### 486 SLAVE PC — \$895\*

Add up to 4 Boards to One Host PC  
Fast Data Transfer and I/O  
PC-104 Port, IDE & Floppy Control  
Independent Processors on One Bus  
No Special Compilers Needed



### TURBO XT w/FLASH DISK — \$266\*

To 2 FLASH Drives, 1 M Total  
DRAM to 2M  
Pgm/Erase FLASH On-Board  
CMOS Surface Mount, 4.2"x6.7"  
2 Ser/1 Par, Watchdog Timer

All Tempustech VMAX® products are  
PC Bus Compatible. Made in the  
U.S.A., 30 Day Money Back Guarantee  
\*Qty 1, Qty breaks start at 5 pieces.

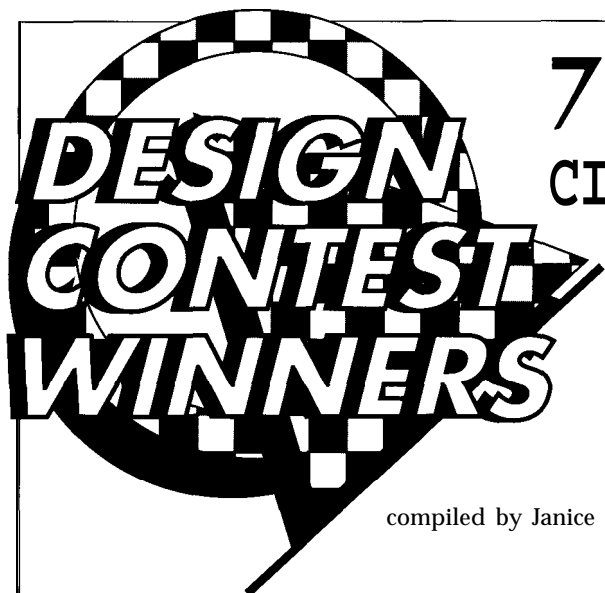
**TEMPUSTECH, INC.**

TEL: (800) 634-0701

FAX: (941) 643-4981

Fax for fast response! 295 Airport Road  
Naples, FL 33942

# 7th ANNUAL CIRCUIT CELLAR DESIGN CONTEST



compiled by Janice Marinelli

## 1ST PLACE DAVID GADDIS

### BATTERY CHARGE/DISCHARGE ANALYZER

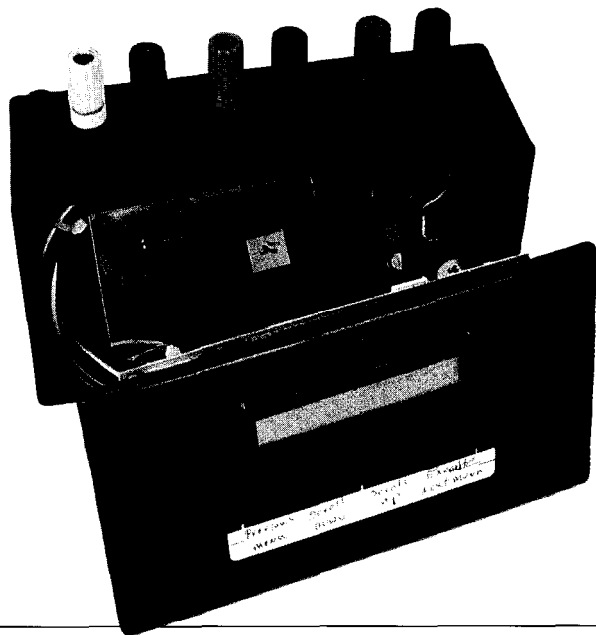
The battery charge/discharge analyzer assists in designing, predicting, planning, and evaluating battery performance on battery-powered equipment. It measures load profile, operating time, and charging characteristics and can provide repetitive charge or discharge cycling.

The analyzer operates as a stand-alone test instrument with limited internal storage of basic instruments or can be connected to the PC for long-term, real-time measurements.

The hardware is built around Motorola's MC68HC705C8 and includes a single-supply rail-to-rail amplifier, A/D converter, A/D interface, and power supplies. Software menu selections can be selected manually or automatically.

Using the battery charge/discharge analyzer, battery run time and life can be accurately measured and predicted under operating conditions.

David may be reached at [gaddis@ix.netcom.com](mailto:gaddis@ix.netcom.com).



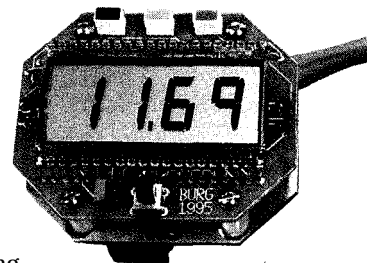
## 2ND PLACE BRUCE WILBER FOUR GAUGE

Bruce set out to design and implement a multipurpose engine-monitoring gauge. He wanted it to fit into a standard instrument hole (2.25" or 3.5"), be rugged electrically and mechanically, have an interface to a remote computer for data logging and display, and use easily available parts. Primarily, though, it had to adapt to various measurements and sensors specific to an engine.

Microchip's PIC16C71 was chosen for its size, availability, and onboard A/D converter. Microchip's three-wire driver combines with a static LCD to display readings. RS-485 provides a noise-resistant, half-duplex, multidrop data link.

Although initially implemented in an automobile, Bruce's true aim is to implement the sensor in an airplane he is building with his father. Currently, it monitors fuel and nitrous oxide pressures. Eventually, it will monitor voltage, oil and fuel pressure, and oil temperature.

Bruce may be reached at [wilber@packet.net](mailto:wilber@packet.net).



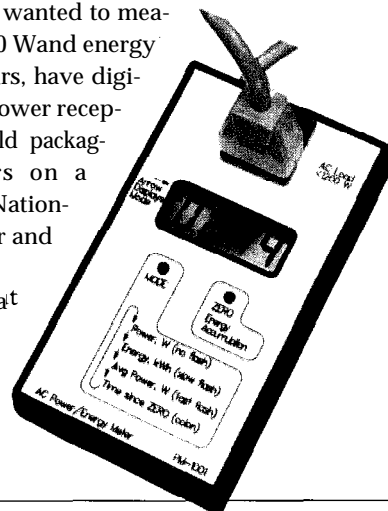
## 3RD PLACE RICK MAY A PIC-BASED AC POWER METER

When there's no way to test power consumption, it's easy to get fed up with claims that a product is "green" or a real "energy saver." To protect the consumer and give manufacturers a tool to prove their claims, Rick set out to develop a power-energy meter.

Using low-cost parts, he wanted to measure AC power from 0 to 1200 W and energy consumption in kilowatt-hours, have digital readout, easy hookup to power receptacles and plugs, and handheld packaging. Rick's design centers on a processing combination of National's ADC083 1 A/D converter and Microchip's PIC 16C6 1.

Initial testing reveals that results are accurate within 10 w.

Rick may be reached at [71241.2673@compuserve.com](mailto:71241.2673@compuserve.com).



# H O N O R A B L E M E N T I O N S

## DAVID GADDIS COLORIMETER

The colorimeter is a color-sensing instrument that provides color matching and identification. Red (660 nm), green (558 nm), and blue (470 nm) light-emitting diodes serve as light sources while a blue-sensitive (human-eye response) photodiode acts as a sensor. The optical outputs of the three LEDs are mixed and the intensity of each varied by a PWM DAC regulator to determine any color wavelength between 470 and 660 nm. The heart of the entire system is a Motorola MC68HC705J1A microcontroller that puts a HC05 core in a 20-pin package.

The colorimeter stores color names and characteristics in EEPROM and compares them against unknown colors. The name of the closest match and its measured color characteristics can be displayed and compared to other known colors.

The wavelength mixing is calibrated to determine the proper current ratios for each wavelength.

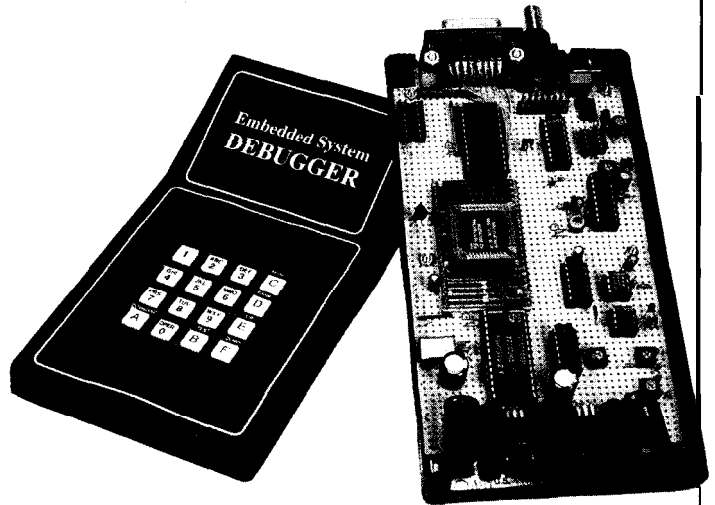
David may be reached at [gaddis@ix.netcom.com](mailto:gaddis@ix.netcom.com).

## GENNADYPALITSKY, MARKNAIDITCH, and DAVID GREEN EMBEDED SYSTEM DEBUGGER

The embedded system debugger debugs embedded and mixed signal circuits. The system is particularly useful in situations when an in-circuit emulator for the target processor is not available. A simple menu-driven system offers video display of the voltages of up to 8 A/D converter channels. Analog information is visualized in bar graphs.

An SAA1101 serves as a system clock, synchronizing both the 87C550 microcontroller and the video signal. The device can use 128 registers of dual-port RAM to communicate with a target microcontroller. Another register sets the target control into test mode while another sends up to 256 commands to the microcontroller.

Gennady may be reached at [gennp@ix.netcom.com](mailto:gennp@ix.netcom.com).



## ROGER GIPSON LED SCOREBOARD

The LED scoreboard displays two team scores using 4-5" displays. A quizmaster sets the value of the question (i.e., 10, 20, or 30). If the answer is correct, the scorekeeper only has to select the responding team and press the plus (+) key. An incorrect answer penalizes the team half the points. The scoreboard automatically makes this calculation when the scorekeeper presses the minus (-) key.

One unique aspect of the project is the manner Roger uses to drive the large LEDs. LM317 voltage regulators serve as current limiting devices in the driver section. As well, a universal driver board accommodates both common cathode and anode displays just by moving jumpers.



## FORMOREINFORMATION

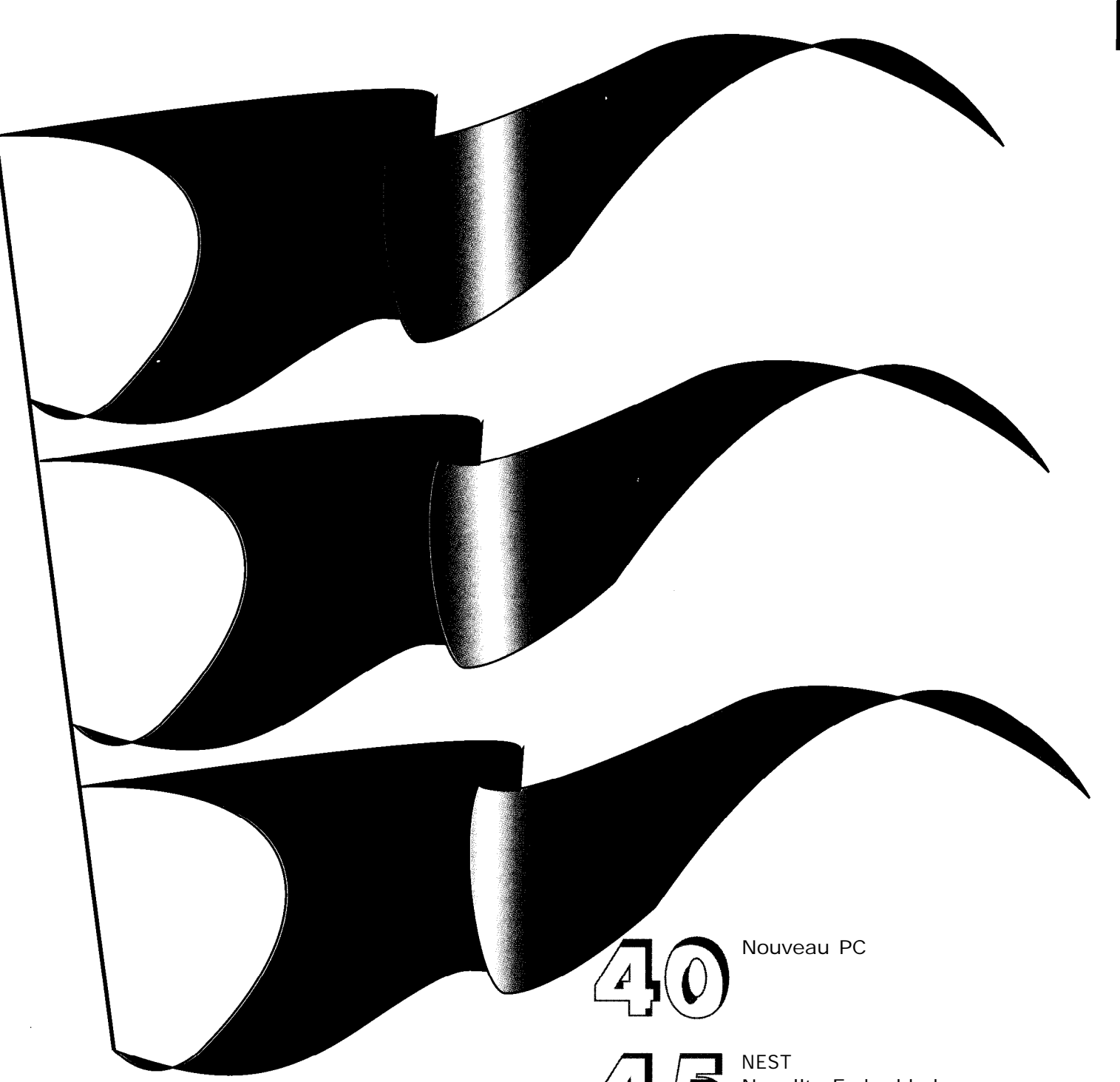
Congratulations to all the winners. It's a pleasure to see the designs based on such a well-rounded mix of processors.

We encourage all Design Contest winners and entrants to write complete articles about their projects. Design Contest articles are highlighted with the finish-line logo seen at the start of this article.

If you'd like more information about a project, you may contact any author who has given their E-mail address. Otherwise, you must patiently wait for the full article to appear in an upcoming issue. We will not give out the phone numbers or addresses of the project designers.

If you don't have E-mail access, we will forward your letter to the designer. Just send it in care of us at:

Design Contest Winner  
Circuit Cellar INK  
4 Park St.  
Vernon, CT 06066.



**40** Nouveau PC

**45** NEST  
Novell's Embedded  
Networking Solution  
Dennis Fredette

**52** Thirty-two-bit Tricks for  
Embedded Controllers  
Larry Fish

**60** PC/ 104 Quarter  
PC/ 104 Embedded Systems  
in Oceanographic Instruments  
Ken Prada

**EMBEDDED PC**  
**EMBEDDED PC**

**67** Applied PCs  
Embedded PC Buses and  
CPU Boards  
Russ Reiss



## CPU BOARD WITH PC/104 PORT

The ICH-486DX contains all the basic elements found in a standard IBM PC/AT-compatible desktop computer along with a PC/104 expansion port on a half-sized ISA-bus-compatible card. This combination is ideally suited for embedded applications.

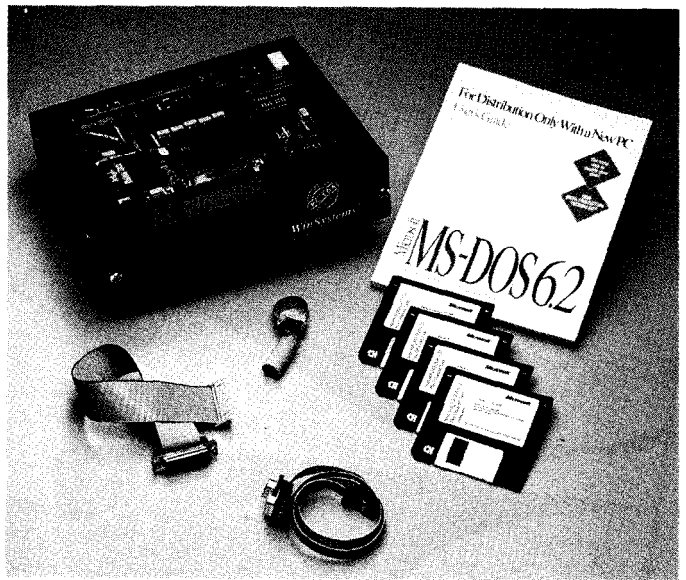
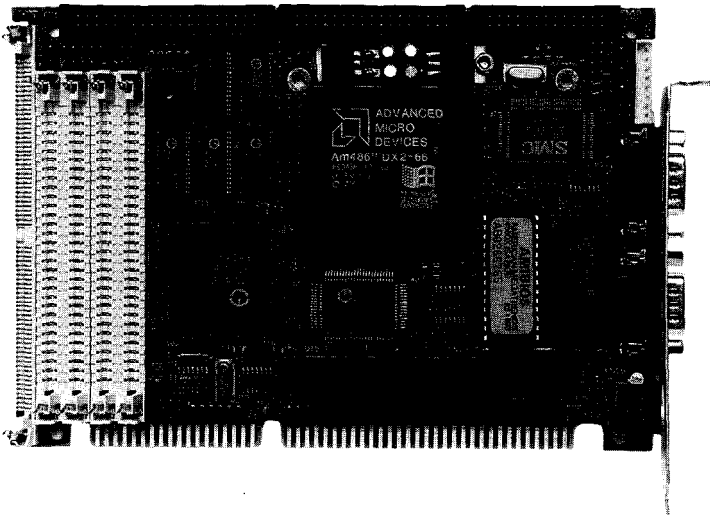
The ICH-486DX contains a full-featured passive-backplane CPU operating at 100 MHz and performs at a Landmark V2.0 rating of 363 MHz. It includes two serial ports, a bidirectional parallel port, a dual floppy disk port, an IDE hard-disk port, a PS/2 keyboard port, an onboard speaker, watchdog timer, and up to 96 MB of DRAM. In addition, each board has a standard PC/104 expansion port for boards such as an EPROM/RAM disk emulator, a video controller, or digital I/O. Since the unit was designed for embedded applications, the BIOS boots even without a keyboard or monitor. A power connector is provided onboard to allow direct connection to an external power supply for use in standalone applications.

The watchdog timer makes the board ideally suited for controlling critical processes where unattended operation is essential. In the event of an I/O timeout delay or external failure, the watchdog timer can be programmed to generate a nonmaskable interrupt or system reset. The timeout delay can be adjusted from 0.5 to 5 s.

The ICH-486DX SBC sells for \$695 without processor and includes a user's manual.

Microcomputer Specialists, Inc.  
2598 Fortune Way  
Vista, CA 92083  
(619) 598-2177  
Fax: (619) 598-2450

#510



## EMBEDDED SYSTEMS DEVELOPMENT KIT

The WinSystems SDK-LBC-104 System Developers Kit (SDK) makes software and hardware development convenient and reliable and eliminates the clutter of stringing a power supply, disks, cables, and computer boards together. Many embedded system designs use off-the-shelf single-board computers with PC/104 expansion modules that operate both as the development system and as the final target system running the application software. The SDK provides the necessary hardware to ease program development and a "known-good environment" to reduce risk and development time.

The SDK enables the single board computer (and any PC/104 expansion boards) to be mounted on top of the enclosure with the peripherals packaged inside. The kit consists of Microsoft's DOS 6.2 (or higher), a 400-MB hard disk, a 1.44-MB high-density 3.5"

floppy drive, a triple-output power supply plus keyboard, power, disk, COM, and LPT cables to interface with WinSystems LBC-486DX embedded SBC. The SBC is purchased separately.

The power supply, floppy disk, and hard disk are mounted in a low-profile, black anodized enclosure for convenience and easy access. The power supply is a 50-W universal switcher that accepts input voltages from 85 VAC to 264 VAC. Output voltages are +5 V at 5 A, +12 V at 2.0 A, and -12 V at 0.5 A.

The SDK-LBC-104 sells for \$895.

### WinSystems

715 Stadium Dr.  
Arlington, TX 76011  
(817) 274-7553  
Fax: (817) 548-1358

#511

# Nouveau PC

edited by Harv Weiner

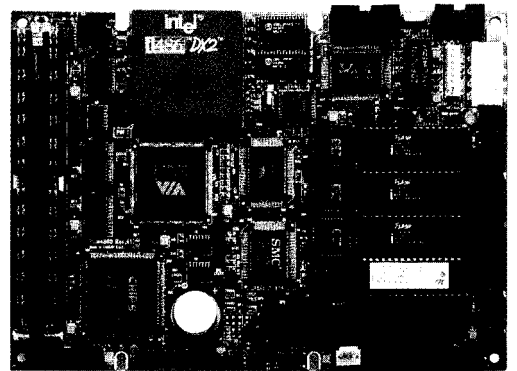
## '486 CPU WITH PC/I 04 EXPANSION

The PCM-4860 is an all-in-one single-board '486 computer with an **onboard** Ethernet interface and VGA CRT/flat-panel controller. The card offers all the functions of a compatible industrial computer on a single board, but it fits in the space of a 5¼" floppy drive (5.75" x 8"). The board is 100% PC/AT-compatible, so programs run without modifications.

**Onboard** features include two serial ports (RS-232 and RS-232, -422, or -485), one parallel port, an IDE hard-drive controller, a floppydrive controller, and a PS/2 mouse interface. The board's watchdog timer automatically resets the system if it stops due to a program bug or EMI.

An **onboard** solid-state disk (SSD) emulates a floppy drive using EPROM or flash memory devices. The system can boot from the SSD, which is accessed using standard DOS commands or BIOS I/O. Capacity is up to 1.44 MB, depending on the size of the memory chips. With flash memory, reads and writes of the disk act just like a floppy. With EPROM, the disk is read-only, and the chips must be programmed with an EPROM programmer. Up to six industry-standard PC/I 04 expansion modules can be added.

The PCM4860 features an **80486SX**, DX, or DX2 processor with selectable clock speed, and supports up to 32 MB of **onboard** DRAM. An Award **128-KB** flash-memory BIOS with power management is included, and the chip set is the VIA VL82C486. The card runs from a single +5-V power supply.



Amdex Industrial Computers

One Trefoil Dr. • Trumbull, CT 06611 • (203) 268-8000 • Fax: (203) 268-2538

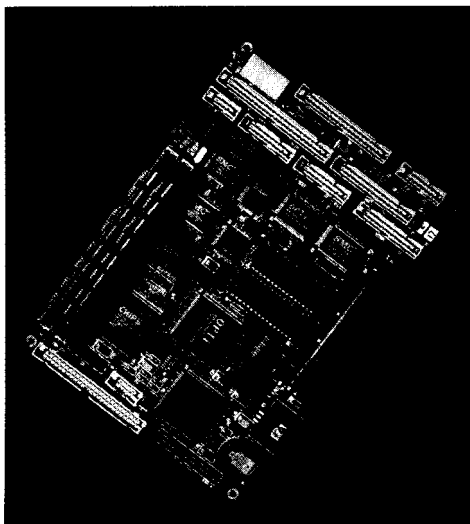
**#512**

## EMBEDDED PC DEVELOPER'S REFERENCE

Ampro is offering its new embedded-PC catalog. This free **100-page** reference includes extensive information on the company's line of **PC/I 04 CPU** and expansion modules (**8088-486SLC2**), **PC/I 04-expandable** single-board computers (**386SX-486DX4**), and a variety of **PC/I 04-oriented** accessories.

The catalog also provides invaluable reference information for the embedded-PC system designer including where to find remote debugger support, real-time and embedded operating system support, a discussion on the advantages of remote versus self-hosted development, as well as a handy embedded-PC system developer's reference list which includes how to obtain industry specifications and guides, white papers on such topics as designing with **PC/104**, using the PC architecture in embedded applications, and more.

**Ampro** Computers, Inc.  
990 **Almanor** Ave.  
Sunnyvale, CA 94086  
(408) 522-2 100  
Fax: (408) **720-** 1305



**#513**

## THIRD-GENERATION SBC

Ampro's **LittleBoard/486i** is the first in a series of **third-generation** PC/AT-compatible single-board computers. It offers a high level of integration as well as high CPU performance.

The **LittleBoard/486i** features a **50- or 100-MHz** Intel '486DX CPU, up to 64 MB of system DRAM, an embedded-PC BIOS, keyboard and speaker interfaces, four buffered serial ports, an IEEE-1 284 (EPP/ECP) parallel port, and floppy and enhanced IDE drive controllers. Also featured are a local bus LCD/CRT display controller, SCSI-II hostadapter, and an Ethernet LAN interface. The

board also contains an array of extensions and enhancements that optimize it for embedded-system applications. Included among these are a watchdog timer, a powerfail NMI generator, and an **onboard** bootable solid-state disk capability.

System operation is based on a single +5-V power source and offers power-saving modes under support of special advanced-power-management BIOS functions. Additional system expansion is accommodated by an **onboard PC/104** expansion-stack location which offers compact, self-stacking modular expandability.

The **LittleBoard/486i** sells for \$899 (50-MHz CPU) and \$999 (100-MHz CPU) in OEM quantities.

**Ampro** Computers, Inc.

**#514**

# Nouveau IPC

## EMBEDDED CONTROLLER

**Total486** is an enhanced controller that provides a complete standalone system for users requiring high performance, low power consumption, and a choice of multiple functions on a compact board.

Total486 uses a **66-MHz '486/DX2** processor and associated chip set to provide a range of configurations on a 9.2" x 6.3" board and mounts directly behind a range of standard LCD display panels. A built-in analog touch-screen interface offers easy implementation of a space- and power-efficient package.

The unit operates from a single +5-V supply and can be configured to provide 1-32 **MB** of system RAM, up to 8 MB of flash memory, up to four RS-232 serial ports (with one selectable as an isolated **RS-485**), an AT-keyboard interface, speaker port, software-controlled watchdog timer, **128-byte** EEPROM for configuration data storage, and a battery-backed real-time clock. Other features of the unit include three additional **32-pin** memory sockets, local bus **SVGACRT** and LCD display interfaces with **onboard** bias voltage generator, and ISA and PC/104 expansion connectors.

You can interface the Total486 through a 24-line TTL I/O interface (Opto-22 solid-state-relay compatible), a scanned matrix keypad, eight-channel A/D converter, **LPT1-compatible** printer port, floppy and hard disks, and a full NE2000 Ethernet interface including twisted pair and Thin Ethernet connectors.

The unit is supplied ready for immediate software development by including a **preconfigured** enhanced version of Datalight's **V6.0** ROM-DOS (MS-DOS V6.2 compatible) together with the BIOS in EPROM. Software utilities enable the user to directly build ROM disk images containing the application program using a standard desktop PC. These images can be downloaded to the **onboard** flash memory for immediate execution on **powerup**.

Dexdyne Ltd.

15 Market Pl. • Cirencester, Glos. **GL7 2PB**, U.K.  
**+(44) 1285-658122 • Fax: +(44) 1285655644**

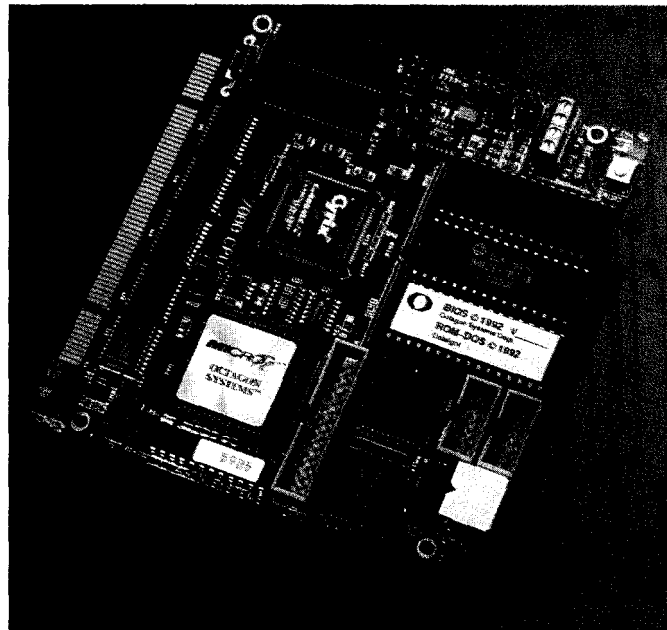
**#515**

## EMBEDDED PC

Octagon Systems presents a feature-rich, **16-bit**, ISA-compatible industrial computer. Whether installed in a passive ISA bus backplane, used standalone, or operated **side-by-side** with **micro-PC-expansion** cards, the 4.5" x 4.9" **Model 7000** is ideal for embedded applications.

The 7000's **25-MHz '486SLC** processor features built-in primary cache to maximize performance. The **16-bit** data bus doubles throughput over the previous generation of **8-bit** ISA-bus-compatible cards. A series of card cages and backplanes accommodates **8-** and **16-bit** micro-PC cards simultaneously.

The 7000 includes three solid-state disks which can be configured with up to 2.5 MB of total storage capacity and fulfill distinct system functions. The first disk contains the **AT-compatible BIOS** with indus-



trial extensions, **utility software**, and DOS 6.0 in ROM. The second disk stores the application program and can be configured with either **1 MB of EPROM** or **512 KB of flash** memory. The flash programmer is built-in for **reprogram-**

ming through a serial port. The third disk is multifunctional and can be used for data conversion tables, multilanguage support, or other operating systems.

The 7000 also contains 4-8 MB of DRAM, a coprocessor socket for an **80387SX**

coprocessor, two **16C550-compatible** serial ports with an RS-232, -422, or -485 interface, an LPT bidirectional parallel port, watchdog timer with a timeout of **1.2 s**, on **AT-style** calendar clock, and keyboard and speaker ports.

The unit is built to withstand extreme temperatures (-400 to 70°C). The card also has very low power requirements (operating at 5 V) and is rated for a **MTBF of 1 12,985** hours. To guard against potential loss of data and time-consuming reinitialization, setup information is stored in non-volatile EEPROM.

The 7000 sells for under \$700 in OEM quantities.

Octagon Systems  
 6510 W. 91st Ave.  
 Westminster, CO 80030  
 (303) **430-1500**  
 Fax: (303) 426-8 126

**#516**

# Nouveau PC

HIGH-SPEED PC DATA LINK

Wideband announces several compact, hybrid, thick-film modules for high-speed serial communication.

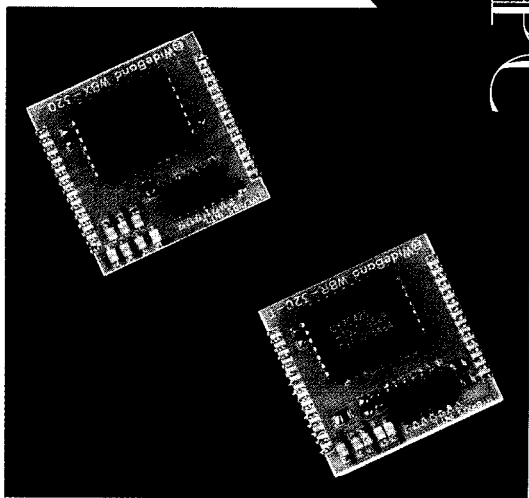
The WBX-320-T transmits serial data over a twisted-pair cable at a bit rate of 320 Mbps. The 30-pin part, which occupies one square inch of board real estate, uses embedded 8- or 1 O-bit encoding resulting in a useful data rate of 256 Mbps over cable lengths of up to 150' (up to 300' with optional extender module).

Data to be transmitted over the serial link is input through an 8-bit parallel interface into a 1-KB onboard FIFO. At the other end of the serial link, the WBR-320-T converts the serial data back to its original 8-bit format where it is stored in an onboard FIFO until read by the user.

The WBX-320-C and the WBR-320-C are identical to the -T components, except they are terminated for 75-Ω coax.

By using four of the transmit modules in parallel, a 1-GB data transfer link can be created over a standard category 5 cable.

PC interface boards are available to originate and receive serial data streams compatible with these modules. Modules may also be directly coupled to the parallel printer ports of conventional computers. The modules operate on 5 V and are available in quantity for \$79 each.



**WideBand Corp.**

26900 East Pink Hill Rd. • Independence, MO 64057 • (8 16) 229-5300 • Fax: (8 16) 229- 1000

#517

# Nouveau PC

YOU DON'T GET TO THE TOP BY THINKING SMALL, SO WHY MESS AROUND WITH INFERIOR TOOLS. ADOPT **TEAM PARADIGM** AND TAP INTO THE FINEST TOOLS, BACKED BY UNLIMITED **FREE TECHNICAL SUPPORT.**



Team Paradigm has the ability to deliver all the embedded system pieces from Intel to AMD, C/C++, or assembly language, and all the Borland/ Microsoft development tools. Plus **Paradigm DEBUG** for your favorite in-circuit emulator and real-time operating system.

Team Paradigm for your current or next x86 project. We deliver the finest embedded x86 development tools, backed by unlimited free technical support. No one is more serious about your success than Paradigm Systems. Call today and get the details before you waste any more of your precious time.

## PARADIGM

*'Nuff said.*

*Proven Solutions for Embedded C/C++ Developers*

**1-800-537-5043**

Paradigm Systems,  
3301 Country Club Road, Suite 2214, Endwell, NV 13760  
(607) 748-5966 / FAX: (607) 748-5968  
Internet: 73047.3031@compuserve.com

**TO BE CONTINUED...**

## OPTOISOLATED PC/ 104 CARD

Saelig's TP024 card provides low-level optoisolation in PC/I 04 systems for up to 12 digital

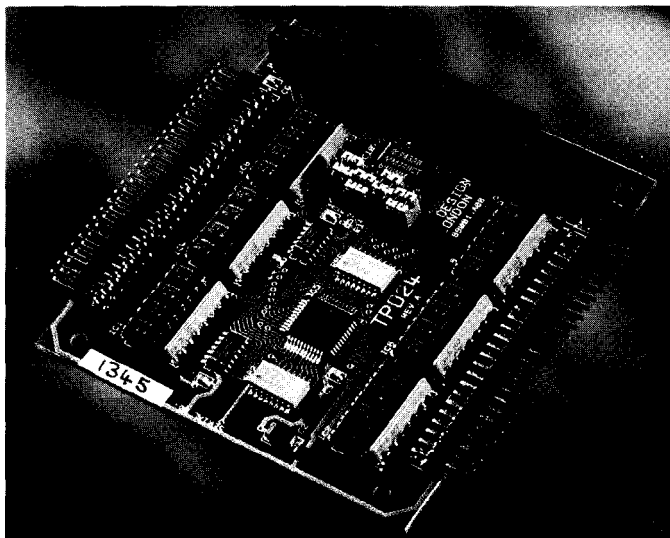
inputs and 12 digital outputs. Although not meant for hazardous voltages, it is useful for breaking ground loops while sourcing or sinking 160 mA. Parallel I/O functions are provided by a 71055 chip with all input and output pins independently isolated from each other.

The 12 optoisolated outputs are rated for 180 mA at 35 V while the 12 inputs are rated for 35 V. Input current-limiting resistor networks are fitted in SIL sockets and can be configured to match voltage-input requirements (no voltage applied to an input is

read as a logic 0 by the software; voltage applied to the input reads a logic 1). The TP024 is I/O mapped with jumper selection for base

address and runs on +5 V only at 55 mA when all optoisolators are latched on.

The stackable TP024 complies fully with PC/I 04 specification Rev. 2.2 and measures just 3.8" x 3.6". Sample software for driving the board is also provided. The unit sells for \$256.



The Saelig Company  
1193 Mosely Rd.  
Victor, NY 14564  
(716) 425-3753  
Fax: (716) 425-3835

#518

# Nouveau PC

## REMOTE POWER CARD!

3 VERSIONS:

### RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

### PHONE

TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

### TIMER

WAKEUP OR SHUTDOWN PC, LATE NITE BACKUP / MODEM, CONTROL AC APPLIANCES



**95\$** 27\$ OEM

ALL MVS CARDS INCLUDE ASM, BASIC, AND C SOURCE FOR PC OR SBC

## 8 CHAN ADC

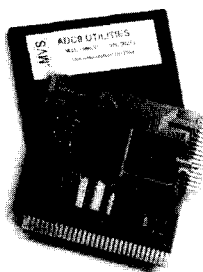
DATA ACQUISITION, SERVO CTL, AUDIO 8-BIT RESOLUTION 22KHZ SAMPLE RATE SHARP CUTOFF ANTI-ALIAS FILTER CREATE STEREO BLASTER(.VOC) FILES

**95\$**

## 2 CHAN DAC

VOICE MAIL, MUSIC, ALARMS, CTL VOLT 8-BIT RESOLUTION 44KHZ SAMPLE RATE PLAYS MONO / STEREO BLASTER FILES FUNCTIONS AS DIGITAL ATTENUATOR TOO

**75\$**



MVS BOX 850  
MERRIMACK, NH  
(603) 792-8507

5 YEAR LIMITED WARRANTY  
MADE IN USA

# CIARCIA DESIGN WORKS

Does your Big-Company marketing department come up with more ideas than the engineering department can cope with?

Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?

Steve Ciarcia and the Ciarcia Design Works staff may have the solution. We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems.

Whether you need an on-line solution for a unique problem, a product for a startup venture, or just experienced consulting, the Ciarcia Design Works is ready to work with you. Just fax me your problem and we'll be in touch.

REMEMBER A  
CIARCIA DESIGN WORKS  
FAX: (860) 871-8986

# NEST

## Novell's Embedded Networking Solution

*With NEST, Novell extends their desktop networking expertise into the embedded market. After overviewing the NEST architecture, Dennis delves into what makes up each specific layer of the network.*

**W**ith the release of Novell Embedded Systems Technology (NEST), Novell is pushing their networking envelope into embedded computing. The client software developer's kit (SDK) gives developers a tool that quickly and easily builds NetWare-ready embedded system devices.

Novell designed NEST to enable developers to minimize their development efforts and the size of their code. How easy is it to network such devices? Join us to see how NEST works. Then, judge for yourself.

### BUILDING NEST

The falling cost of computing power has caused the proliferation of embedded systems controlling a wide variety of devices. Embedded systems developers have added more sophisticated functions and interfaces to intelligent devices.

Many of these devices are now powerful enough to support a direct network connection to a host or other devices, replacing the slow or proprietary connection they traditionally used.

A good example is a printer. In the past, a serial or parallel connection linked a user workstation or server. This connection was slow and in most cases unidirectional. They were not suitable for printing large complex documents or reporting errors such as out of paper or low toner.

A network connection gives these devices fast access to their data and easily reports errors. The device also distributes some of the load from the server. A networked printer, for instance, typically services its own print queue, reducing the overhead required on the server.

In general, any device which communicates with a host or user workstation benefits from a network connection. The high bidirectional data rate offered by the network lets the device be more interactive, offering a better user interface.

The device advertises its presence on the network and communicates either directly with the user or via a server. In addition, the device can also use services from the network. It can read its **configura-**

tion from a file on the server or spool data to the server's file system.

The combination of embedded system devices and NetWare technology isn't new. However, before the release of NEST, developers built NetWare support into such devices by reverse engineering NetWare.

Reverse engineering can be tricky, requiring years of development. For third-party developers, reverse engineering was further complicated by a lack of documentation. Licensing only provided rudimentary detail on how NetWare transport and core protocols worked. With limited help, third-party developers had to figure out how Novell engineered code and provide the same functionality in their own way.

NetWare 4.1 made this process virtually impossible. Reverse engineering could no longer unfold its encryption algorithms, security codes, and advanced network services.

The NEST client SDK eliminates the need to reverse engineer NetWare code. NEST is a ready-made, embedded device

connection to NetWare networks, enabling OEMs to bring NetWare-compatible products to market within months, rather than years.

Because development times are short and include current NetWare code, OEMs can support the most current version. NEST's SDK is based on NetWare 4 code, which Novell engineered to be backward-compatible with version 3. Because NEST is derived from proven NetWare code, NEST-based products are reliable.

The NEST architecture is open and modular. Developers can network any embedded, multitasking, real-time operating system. Also, you only need the modules required to develop a particular system, minimizing the size of their code.

## FITTING NEST INTO NETWORKING

Figure 1 illustrates how the NEST architecture fits into network computing. The Open Systems Interconnection (OSI) model, written by the International Standardization Organization, consists of seven layers that define the functions necessary for computers to communicate with each other. You can also see how the various NetWare protocols fit into the OSI model.

The layered design of NEST corresponds to the NetWare protocol layers. As Figure 1 shows, the full spectrum of NetWare networking capabilities is built into NEST. Essentially, NEST is NetWare, squeezed and optimized for embedded system use.

Notice that the NEST architecture contains the following major modules:

- portable operating system extension (POSE) interface

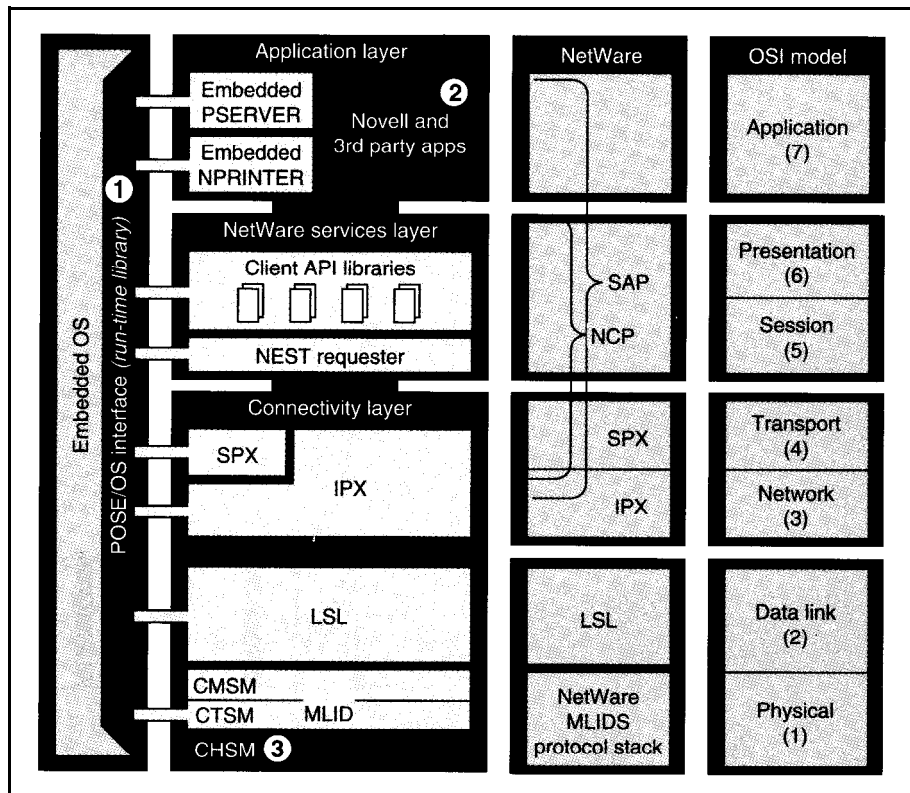


Figure 1: The full spectrum of NetWare networking capabilities is built into NEST. Numbers 1-3 (highlighted in red) identify the pieces developers may have to create.

- application layer
- NetWare services layer
- connectivity layer

The following sections explain each of these modules, beginning with the embedded operating system and the POSE interface.

## THE POSE INTERFACE

The embedded operating system controls the function of embedded device hardware. POSE is a specification (included as part of the client SDK) that developers use

to build an interface between the functions ordinarily provided by an embedded operating system and NetWare.

With the POSE specification, developers create an interface between NetWare and the embedded operating system. It defines all of the embedded operating system services required to support NEST architecture as a set of function calls.

The POSE interface specification defines all the embedded operating system services required to support NEST architecture as a set of function calls. The specification includes the parameters and return codes that each function call must pass to NetWare. POSE functions let the developer control various system-level operations such as memory management, task switching, synchronization, and timing (see Table 1).

Of the 31 POSE functions, 17 conform to POSIX standards, increasing its compatibility with existing embedded operating systems. This compatibility makes it easier for developers to use the embedded operating system of their choice.

To link NetWare and a chosen embedded operating system, a developer either creates a POSE interface or obtains one from the embedded operating system vendor.

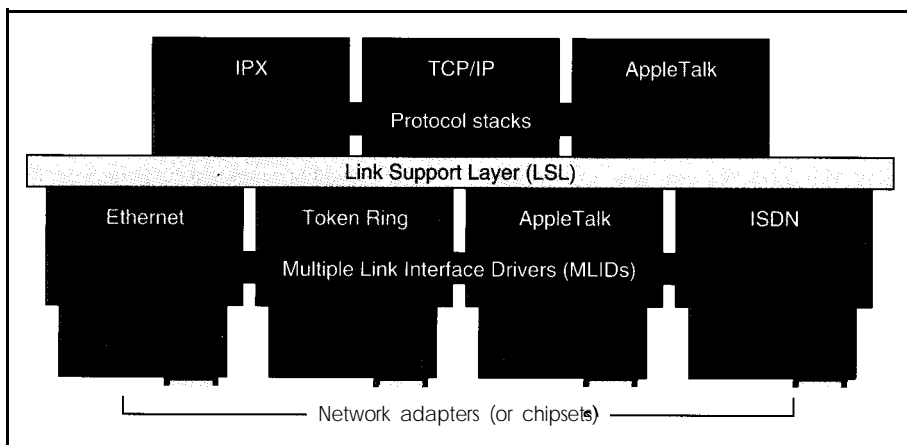


Figure 2: Novell's open data-link interface is the foundation for the NEST.

Function Name	Description
POSEMemoryFree	Deallocate memory
POSEMemoryAllocate	Allocate memory
POSEClockGetResolution	Get system clock resolution
POSEClockGetTime	Get the system clock
POSEClockSetTime	Set the system clock
POSESemaphorePost	Unlock a semaphore
POSESemaphoreTryWait	Conditionally lock a semaphore
POSESemaphoreWait	Lock a semaphore
POSEUnnamedSemaphoreDestroy	Destroy an unnamed semaphore
POSEUnnamedSemaphoreInitialize	Initialize an unnamed semaphore
POSEThreadCreate	Create an execution thread
POSEThreadExit	Terminate the calling thread
POSEThreadJoin	Wait for a thread to terminate
POSEThreadYield	Yield to another thread
POSETickInterruptChain	Establish an interrupt service routine
POSETickInterruptUnChain	Remove an interrupt service routine
POSEInterruptProtect	Suspend hardware interrupts
POSEInterruptUnProtect	Restore hardware interrupts
POSEInterruptVectorSet	Specify an interrupt handler
POSEInterruptVectorClear	Remove an interrupt handler
POSEMessageLog	Log message
POSEProcessIDGet	Return the caller's process ID
POSEInternalInitialize	Initialize POSE subsystem (must call first)
POSEInternalDeInitialize	Deinitialize POSE subsystem (must call before exit)
POSENanoSleep	Delay for a specified period
POSEPrivilegedProtect	Begin a critical period
POSEPrivilegedUnProtect	End a critical period
POSEPrivilegedCreate	Create a privileged thread
POSESchedPriorityGetMin	Get minimum scheduling priority (lowest/worst)
POSESchedPriorityGetMax	Get maximum scheduling priority (highest/best)
POSEPrivilegedExit	Terminate the calling privileged thread

Table 1: The POSE specification allows NEST to interface to any operating system.

dor. As part of the NEST 1.1 client SDK, Novell supplies a ready-made POSE interface to FlexOS, an operating system from Integrated Systems.

A developer using FlexOS as the embedded operating system doesn't need to create a POSE interface. If the developer doesn't want to use FlexOS, they can use the POSE interface for FlexOS as a template to develop an interface for another operating system. The POSE interface spec and the FlexOS POSE interface simplify the process of networking devices.

Other operating systems and their accompanying POSE interfaces will soon be available.

#### APPLICATION LAYER

At the application layer, manufacturers provide the applications that enable NEST devices to perform the functions users need. Manufacturers provide one or more applications that enable their particular device to perform its intended functions on a network.

NEST applications give users varying degrees of control over devices. For example, if a print queue server utility is embedded in a printer, users can use the utility as if it were running as a server-based application.

Manufacturers also develop Windows or DOS applications that communicate with a NEST device. A developer can write a Windows-based network management product to manage NEST devices across a company-wide network.

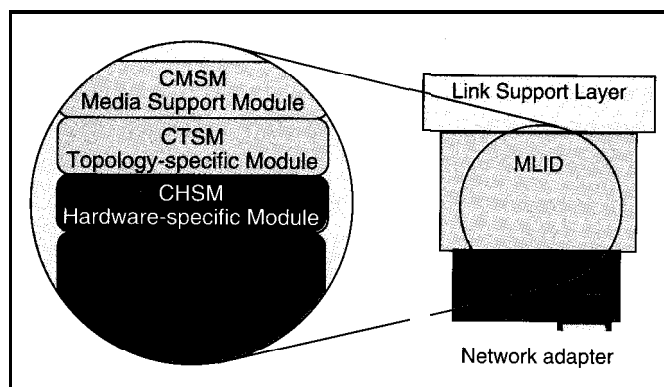


Figure 3: MLIDs handle the sending and receiving of packets to and from the physical network.

In other cases, user control of NEST devices might be unnecessary or unwanted. An environmental control application regulates the temperature of each part of a building by monitoring a network of thermometers. After initial configuration, control of heating and air conditioning devices proceeds as needed. Typically, a company doesn't want building occupants to affect temperature.

NEST applications request network services through the NetWare client Application Programming Interface (API) library. As a set of more than 700 individual libraries (function calls), the client API library is part of the NetWare services layer. The collective client library includes function calls for managing data migration, directory services, messaging, and so on.

For lower-level functions, applications use a set of transport service APIs to directly access transport-level services, which are part of the connectivity layer. Transport service function calls enable developers to open an SPX socket, establish a connection with a listening socket or a remote partner, send or receive sequenced packets, and close connections and sockets.

The client SDK includes embedded device versions of two NetWare printer utilities: Embedded PSERVER (EPS) and Embedded NPRINT (ENP). In NEST printers, EPS is optional, but ENP must be included. EPS manages print queues and routes print jobs, while ENP communicates with EPS and manages the physical printing of jobs. ENP includes the printer control module, which is the code that interfaces ENP with the print driver. ENP receives print jobs from EPS and then routes them through the PCM to the printer driver, which, in turn, communicates directly with the printer engine.

By embedding EPS and ENP directly into their devices, printer manufacturers eliminate the need for printer applications running on a separate network client or server. This step saves memory space in clients and servers and, in some cases, eliminates the need to purchase dedicated print-server hardware.

#### NETWARE SERVICES LAYER

As Figure 1 illustrates, the NetWare services layer acts as a



bridge between the application and connectivity layers. It contains the client API libraries and NEST requester. The client API libraries are modules of code that provide access to more than 700 NetWare services, including bindery, directory, printing, connectivity, file, and messaging services.

These libraries enable NEST devices to perform network client functions such as:

- access and manipulate files as a PC equipped with NetWare client software does. For instance, a NEST-enabled fax machine can log onto a server and then

open, read, fax, and close the file. NEST devices can access accounting information such as client connection time or disk space.

- data migration services automatically move old files to off-line storage devices, preserving main storage capacity
- services unique to a particular device. Developers can minimize the size and complexity of their applications and leverage existing network services.

At the bottom of the NetWare services layer is the NEST requester (see Figure 1). Like the traditional NetWare client re-

quester, the NEST requester manages application requests and server responses. To issue a request, the requester builds packets, adding the packet signature and using RSA authentication services to encrypt the account name and password before transmission.

To send packets to servers, the NEST requester calls the transport services provided in IPX. The requester error checks the data flow by using features such as auto reconnect (restores dropped connections), resend (resends unacknowledged packets within a specified time), and packet burst (supports efficient bulk data transfer).

CONNECTIVITY LAYER

The NEST connectivity layer contains the transport mechanisms and other software needed to move packets across the network wire, allowing network nodes to communicate and exchange data.

The layer's architecture is based on Novell's open data link interface model, which supports multiple transport protocol stacks and link interface drivers via an intermediary layer called the link support layer. The ODI model is shown in Figure 2.

NEST provides a complete NetWare connectivity layer, including separate and complete IPX and SPX protocol modules. If developers use only the IPX and SPX protocol stack, they don't need to create code at this layer. This savings eases management of data streams, a difficult low-level programming task.

Some applications require only IPX transport services. In such cases, developers can omit the SPX module from their product. The developer needs the SPX module in the product only if packet acknowledgment and sequencing are necessary.

If developers want their devices to support multiple transport protocols (such as IPX/SPX, TCP/IP, and AppleTalk), the ODI model enables them to easily implement multiple transport protocol stacks in the same embedded system. Although the client SDK includes the IPX/SPX protocol stack, developers can supply other protocol stacks such as TCP/IP and AppleTalk.

The connectivity layer also contains the LSL, which manages communications between the transport protocol stacks and the MLIDs (see Figure 2). When an application sends a packet, the LSL accepts the packet from whichever protocol is handling the

**Let Your Development Fly!**

Choose Hitex professional tools for your embedded microprocessor design and get your project off the ground ahead of schedule. Hitex builds all types of microprocessor development tools, from sophisticated in-circuit emulators to remote debuggers, monitors and simulators. Complete solutions are available for:

- ✓ the complete 8051 family
- ✓ 80C86/88, 80C186/188EA/EB/EC/XL, 80286, V20/V30/V40/V50
- ✓ 80386DX/SX/CX/EX
- ✓ 80C165/166/167
- ✓ 68HC11 family
- ✓ 6800x, 6830x, 6833x, 68340

Call Hitex for your free demo package and learn how smooth and efficient development with professional tools really can be!

**HITOLS Inc.**  
 2055 Gateway Place  
 Suite 400  
 San Jose, CA 95110  
 ☎ (408) 451-3986  
 ✉ 1-800-45HITEX  
 Fax: (408) 441-9496

**hitex**

CALL US NOW!  
 1-800-45HITEX

teletest 32

New TX886  
 Entry-level 8086X  
 in-circuit  
 emulator

packet and assigns it to the proper MLID. When a client receives a packet, this scenario reverses.

The MLIDs send and receive packets to and from the physical network (i.e., network adapters and wire). As Figure 3 indicates, MLIDs contain three modules:

- C language media-support module—contains general functions common to all drivers.
- C language topology-specific module—manages operations unique to specific topologies such as Ethernet, Token Ring, and Fiber Distributed Data Interface (FDDI). It supports multiple frame types, which can be defined for a given topology. (A frame is a discrete package of information ready for transmission over the network wire. Typical frames contain a header, which specifies handling instructions, and a segment of data.)
- C language hardware-specific module (CHSM)—handles all interaction with the network interface hardware and con-

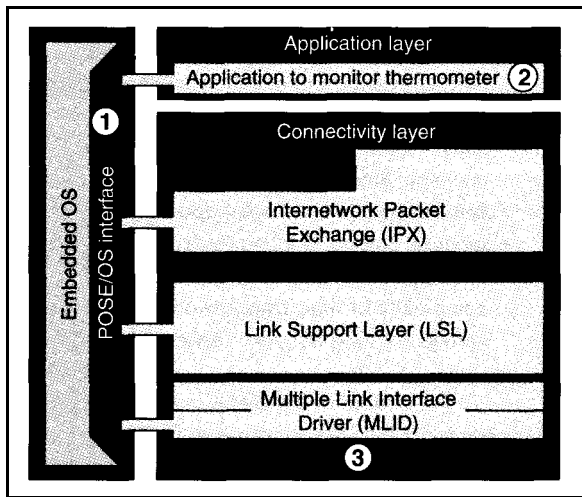


Figure 4: Only certain NEST pieces are needed to create an intelligent thermometer. Those numbered 1-3 may need to be written by the developer.

tains various hardware-specific drivers. The CHSM's functions include adapter initialization, adapter reset and shut-down, and packet reception and transmission.

The only MLID module that developers might have to create is the CHSM. The client SDK includes the CHSM for Novell's NE2 100 Ethernet network adapter specifi-

cation. In this case, developers do not need to program the connectivity layer. They can also use the NE2 100 driver as a template, modifying it to build a driver that works with the network adapter they intend to use.

### PROGRAMMING

Because of NEST's pick-and-choose modular architecture and because it provides almost all necessary embedded system-to-NetWare connectivity, developers find making networkable devices simple. At most, a developer must build three small pieces of architecture:

- a POSE interface
- one or more applications to enable the device to perform functions
- a CHSM module for an MLID

To show how easy it is to create a network embedded device, let's look at the steps involved in building NetWare connectivity into an intelligent thermometer and a NEST printer.

The intelligent thermometer would be a simple NEST device. The thermometer regularly broadcasts its status and current temperature reading. Figure 4 shows the necessary architectural pieces. Notice that several elements—the NetWare services layer and SPX protocol—are not required. The three pieces the developer has to create are shown in red and labeled 1-3.

For the thermometer, as for any device, a developer might have to create the POSE interface to the chosen embedded operating system (see number 1 in Figure 4). The developer can avoid creating the POSE interface by using FlexOS and its POSE interface provided in the client SDK.

So the thermometer can perform its intended function, the developer creates a simple application (number 2 in Figure 4). The application includes the instructions the thermometer needs for broadcasts. Because the thermometer does not need to confirm its broadcasts are received (i.e., two-way communication), the developer can use the transport APIs to directly call IPX. In other words, the NetWare services layer and the SPX protocol module (part of the connectivity layer) can be omitted.

If developers do not use the ready-made Novell NE21 00 network adapter driver

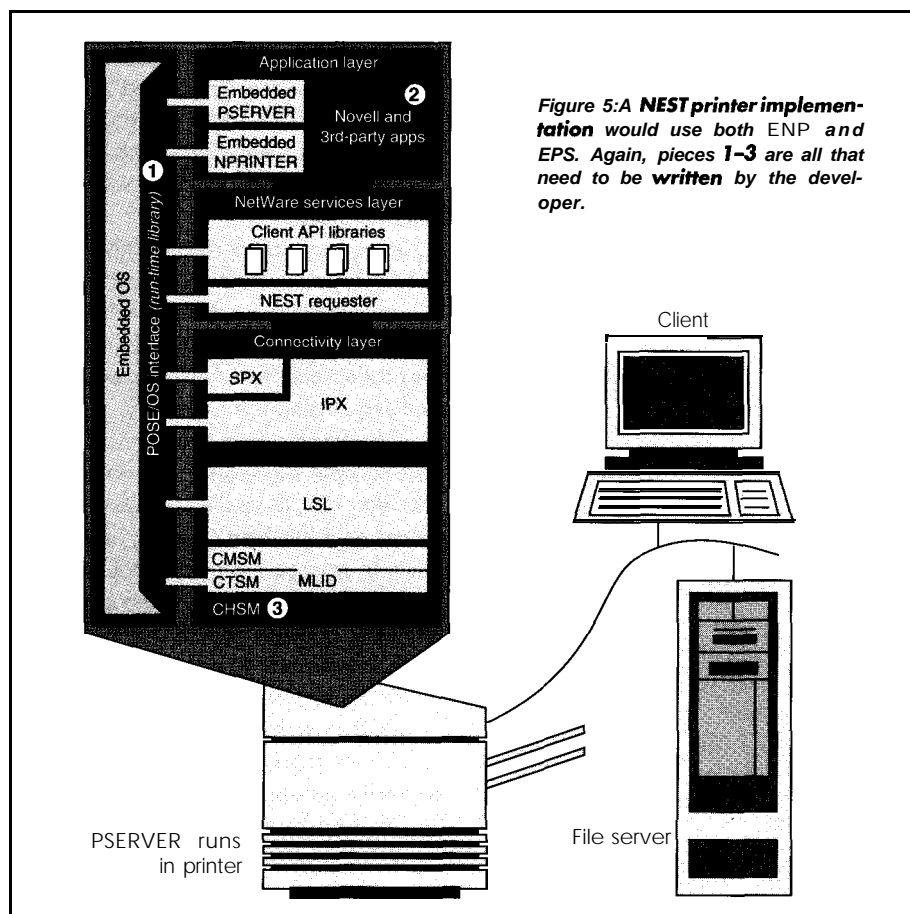


Figure 5: A NEST printer implementation would use both ENP and EPS. Again, pieces 1-3 are all that need to be written by the developer.

code, they have to create their own MLID. The development process is fairly simple:

- modify the existing CHSM or create a new one (see number 3 in Figure 4)
- if the topology is not Ethernet, create a new CTSM
- combine the new modules with the CMSM and other required NEST modules

If the manufacturer of the intelligent thermometer uses FlexOS and the Novell NE2 100 driver code (CHSM) for the physical network connection, then a developer

only creates a small application and combines it with ready-made NEST pieces.

If the developer later wants to enable network users to control the temperature, the developer needs to build NEST-based thermostats instead of thermometers.

If the thermostats needed two-way communication with guaranteed message delivery, the developer needs to include the SPX protocol module. The developer then modifies the application and recombines it with the necessary NEST modules.

As mentioned earlier, printer manufacturers can use the NEST EPS and ENP

utilities to embed NetWare print services into printers. Networking an embedded system printer requires all NEST layers. The printer requires all the services in the connectivity and services layers, requester, and selected client API libraries.

In the application layer, the developer chooses to use only ENP or both ENP and EPS. If the developer includes only ENP, NetWare's PSERVER application manages the printer's queues from somewhere else on the network. If the developer includes the EPS application, the printer has direct access to print server queues. Any embedded printer with EPS can manage queues for itself and any other network printers.

Regardless of whether a developer includes only ENP or both ENP and EPS, the developer must write the printer driver that enables the PCM to communicate with the printer. Figure 5 shows a NEST printer implementation using both ENP and EPS.

#### READY-MADE AND WAITING

NEST is a nearly complete embedded systems interface to NetWare networks. Although developers must create at most three pieces of their product, the rest is already built into NEST.

NEST developers can use any embedded operating system, including any proprietary operating system they have already developed. From POSE to the application layer to the MLIDs in the connectivity layer, NEST's architecture is modular, portable, operating-system independent, network-media independent, and reliable. EPC

*Dennis Fredette is the owner of the Niche Agency, which specializes in technical writing and editing. He is a frequent contributor to computer publications. For more information on NEST, contact Nick Webb at [nwebb@novell.com](mailto:nwebb@novell.com).*

#### CONTACT

Novell, Inc.  
122 East 1700 South  
Provo, UT 84606  
(801) 429-5348  
Fax: (801) 429-3424

#### I R S

- 4 10 Very Useful
- 41 1 Moderately Useful
- 412 Not Useful

# Anatomy of a Great Frame Grabber

**Low Price** .....  
Forget high priced on-board processing. Our designs exploit the host CPU without sacrificing performance.

**Small Board** .....  
Half slot ISA, PC/104 and STD boards for embedded and compact designs.

**Precision** .....  
**Image** processing  
Industry leading sampling jitter less than  $\pm 3nS$ . Video noise less than 1 LSB.

**Easy Interface** .....  
Image processing library, C library including source, Windows DLLs for C and Visual Basic.

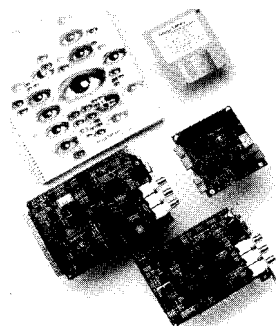


## Call 800-366-9131

Call and find out how ImageNation's quality products, responsive technical support and elegant software get your application quickly to market.

**ImageNation™**  
*“Vision Requires Imagination”™*

I-800-366-9131 • Phone: 503-641-7408 • Fax: 503-643-2458  
E-mail: [info@ImageNation.com](mailto:info@ImageNation.com)  
Homepage: <http://www.ImageNation.com/~image>  
P.O. Box 276, Beaverton, OR 97075-0276



Larry Fish

# Thirty-two-bit Tricks for Embedded Controllers

*Larry shows us how to get all the benefits of a 32-bit unsegmented architecture and still operate happily under DOS and BIOS. You can get at the power of the 80386 processor With conventional OS code.*

The 80386 processor is used in an increasing number of embedded controllers. Why?

Because it:

- offers a 32-bit processor with 4 GB of memory space
- has hundreds of hardware and software products available to support applications
- is essentially just a miniature PC. You can do all the software development and testing on a PC.

There's only one problem: It can be a real challenge getting all the power you can out of a '386.

To understand the problem, let's look at the '386 architecture more carefully. The '386, '486, and Pentium processors have two basic modes of operation: real and protected. In real mode, the processor works like a fast 8086, but it also has all the limitations of the 8086, including 1 MB of memory space and 64-KB segments.

In protected mode, the processor becomes a full 32-bit processor with a 4-GB memory space and sophisticated memory-management features. Unfortunately, DOS and BIOS are not compatible with protected mode.

Since you probably want to do most of your software development under DOS, not being able to run in protected mode is a real problem. Also, most of the commercially available '386 controllers use BIOS and DOS, so they have trouble running in protected mode.

Actually, there is a way to get all the benefits of a 32-bit unsegmented architecture and still operate happily under DOS and BIOS. In this article, I'll give you some suggestions that will help you get at the power of the '386 with conventional DOS and BIOS.

## WHAT'S THE PROBLEM?

Let's look at some of the problems you encounter if you run in protected mode under DOS and BIOS. To begin with, when

you switch to protected mode, interrupts change drastically. In real mode, interrupt tables reside in low memory, but in protected mode, they can be located anywhere.

What's worse, interrupts use 32-bit addresses instead of 16 bit. Neither DOS nor BIOS can handle this type of interrupt scheme. The first interrupt crashes the system. So, before you can even switch to protected mode, you have to write a set of routines that intercept and deal with each and every protected-mode interrupt.

Another problem you encounter is that DOS cannot properly load protected-mode programs. When DOS loads a program, it puts the program anywhere in the 640-KB main-memory block.

After the program is loaded, DOS adjusts certain addresses so they reflect the actual location where the program was loaded. Protected-mode programs have 32-bit addresses, which causes a real problem since DOS doesn't know how to adjust 32-bit addresses.

There are several solutions to these problems. You could run OS/2 or Windows NT (not very practical for a microcontroller). You could set up your own interrupt tables and write your own DOS loader. Or, you could buy a commercial DOS extender.

DOS extenders have their own interrupt tables and program loaders, and provide a host of support functions for **protected-mode** programs. Unfortunately, they are expensive and usually require a royalty if you sell your application.

So what's the best solution?

Use some simple techniques that take advantage of features hidden in the '386 and '486 architecture. These methods work because the '386 can do 32-bit operations even when it is in real mode.

Doing **32-bit** operations in real mode means that you don't need:

- a DOS extender
- to deal with interrupts
- to contend with the arcane machinery of the '386 in protected mode.

All you need is an assembler which is capable of assembling **32-bit** instructions (such as Microsoft's MASM or Borland's TASM).

Just a word of warning before you start experimenting with 32-bit operations. Memory managers like QEMM, EMM386, and HIMEM sometimes put the processor in V86 mode. V86 mode causes problems with the following experiments, so remove all memory managers before trying them out.

## ASSEMBLING 32-BIT INSTRUCTIONS

I'd like to take you through a series of experiments to explore the '386 architecture. Everything is done on the PC because it's easy to test the software and experiment. But remember, everything on the PC is directly transferable to the '386

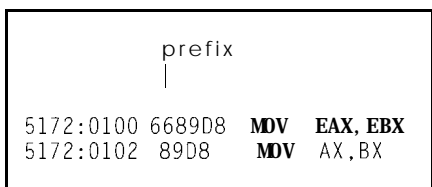


Figure 1: A portion of the **Codeview** display shows both **16-** and **32-bit** instructions. You can see the prefix 66h in front of **32-bit** instruction.

microcontroller. You need an IBM PC-compatible '386, '486, or Pentium, an assembler, and a debugger.

I use assembly language because it is easier to see how everything works. Of course, the same methods can be applied to higher-level languages like C and Pascal.

One of the difficult parts of writing **32-bit** programs in real mode is getting the assembler to assemble the code properly. This code fragment exposes this difficulty.

```
.386

CSEG    SEGMENT
        ASSUME  CS:CSEG
        ORIGIN  0100H

START:  MOV     AX, BX
        MOV     EAX, EBX

CSEG    ENDS
        END     START
```

The program has two instructions: MOV AX, BX (16 bit) and MOV EAX, EBX (32 bit). If you assemble the program and look at it with a debugger like **Codeview** (Microsoft's debugger), you see something strange:

```
MOV     EAX, EBX
MOV     AX, BX
```

The instructions are swapped! The **16-bit** instruction is now a 32-bit instruction, and the 32-bit instruction is now a **16-bit** instruction!

On the '386, both instructions have identical opcodes. Three things determine whether the instruction is 16 or 32 bit. The first is the processor mode. If the processor is in real mode, it automatically defaults to **16-bit** instructions.

But if the processor is not in real mode, it looks at the D bit in the descriptor for the current segment. (Descriptors are special tables that are used by the '386 to control memory access.) If the D bit is set, the **processor executes** all instructions as 32-bit instructions. If the D bit is cleared, the processor treats all instructions as 16-bit instructions.

Finally, each instruction can have a prefix byte which changes the way the instruction works. The prefix byte doesn't set the mode-it changes it.

If you are in 32-bit mode, the prefix byte causes the operation to be 16 bit. If you are in 16-bit mode, it causes the operation to be **32 bit**. Thus, the same prefix byte has different effects depending on the mode you're in.

All of this becomes even more confusing when the assembler comes into play. The assembler needs to know what mode the processor is in when the code executes.

If the processor is in 32-bit mode, the assembler must put a prefix byte in front of a **16-bit** instruction to force a **16-bit** operation in the **32-bit** environment.

If the program runs in real mode, the assembler must force 32-bit instructions to be 32 bit by putting a prefix byte in front of the opcodes.

It's now easy to see why the code fragment behaves so strangely. The `.386` at the start of the program makes the assembler think the program is running in **protected mode**, so **32-bit** operations are the default. As a result, the assembler puts a prefix byte in front of the **16-bit** instruction and not in front of the 32-bit instruction.

But, when **Codeview** actually runs the program, it's in **16-bit** mode, so the prefix byte is in the wrong place. If you look more closely at the **Codeview** display shown in Figure 1, you can see the prefix byte 66h in front of the 32-bit instruction.

The `.386` directive at the start of the program instructs the assembler to accept '386 instructions, but it also tells the assembler that the program runs in **protected mode**. If you want to assemble 32-bit instructions in real mode, tell the assembler that the program runs in **16-bit** mode.

You can do this with the `USE 16` directive:

```
.386

CSEG    SEGMENT    USE16
        ASSUME  CS:CSEG
        ORIGIN  0100H

START:  MOV     AX, BX
        MOV     EAX, EBX

CSEG    ENDS
        END     START
```

This code fragment is identical to that shown earlier, except for the USE16 parameter in the code-segment declaration.

This parameter tells the assembler that the code is running in real mode so it makes the proper assumptions about how to prefix the opcodes.

## ACCESSING 32-BIT ADDRESSES

Even though you can assemble 32-bit instructions, you still need to know how to access data using 32-bit addresses if you want to use the full 4 GB of memory space on a '386. Otherwise, the processor gives you an error if you try to exceed a segment boundary.

This small program loads a value from memory using the EBX register as an indirect pointer:

```
.386

CSEG      SEGMENT  USE16
          ASSUME   CS:CSEG
          ORG      0100H

START:    MOV      EBX, OFFFOH
LABEL:    MOV      EAX, [EBX]
          INC      EBX
          JMP      LABEL

CSEG      ENDS
          END      START
```

The best way to test and execute this program is to single step through it with a debugger like Codeview. If you execute it as a stand-alone program, it crashes your computer.

The EBX register is 32-bit, so it should load from any location within the processor's 4-GB memory space. The program first loads EBX with the value FFF0h. This address is just a few bytes short of the end of the segment. Each time the program goes through the loop, it increments EBX and accesses new memory locations.

Within a few cycles, EBX points to an address beyond the end of the segment. Normally, the processor hangs or reboots when this happens because the processor's protection features limit segment size to 64 KB in real mode. When the address exceeds 64 KB, a general-protection error is generated. General-protection errors are 32-bit faults and neither BIOS nor DOS can deal with them.

To get around this problem, reset the segment limit from 64 KB to 4 GB. It is

*Listing 1: This program tests 4-GB memory addressing in real mode.*

;The program should be assembled as follows:

```
        MASM      TEST4G;
        LINK      TEST4G;

; The program should be tested under a debugger like Codeview
; or Turbo Debugger. If you use Turbo Debugger, don't use the
; '386 version, "TD386". You cannot single step through the
; protected-mode portion of the code with most debuggers. You
; can single step through the main loop, but don't single step
; the subroutine labeled "SETUP." Step over this routine
; using the F10 command in Codeview or the F8 command in Turbo
; Debugger.

        .386P

CSEG      SEGMENT  USE16
          ORG      0100H
          ASSUME   CS:CSEG,DS:DSEG,ES:CSEG

START:    MOV      AX,SEG DSEG      ; Point to data segment
          MOV      DS,AX

          CALL     SETUP           ; Reset segment limits

          MOV      EBX,OFFFOH     ; Test segment limits
START1:   MOV      EAX,[EBX]
          INC      EBX
          JMP      START1

; This macro builds segment descriptor using supplied arguments
; Arguments are:
          LIMIT: size limit of the segment (20 bits)
          BASE:  starting location of the segment (32 bits)
          GRAN: granularity of the segment, byte or 4 K (1 bit)
          DEF:  default address of the segment 16 or 32 bits (1 bit)
          PRS:  The present bit, indicates segment is valid (1 bit)
          DPL:  The descriptor privilege level (2 bits)
          DTP:  The descriptor type (1 bit)
          TYP:  The memory type (4 bits)

DESCRPT MACRO LIMIT,BASE,GRAN,DEF,PRS,DPL,DTP,TYP
          LOCAL  ATTRIB,A1,A2,A3,A4

ATTRIB = (GRAN SHL 15) OR (DEF SHL 14) OR (PRS SHL 7) OR
          (DPL SHL 5)
ATTRIB = ATTRIB OR ((LIMIT SHR 8) AND 0F00H) OR (DTP SHL 4) OR TYP

A1      = LIMIT AND OFFFHH      ; LIMIT
A2      = BASE AND OFFFHH      ; BASE
A3      = (BASE SHR 16) AND OFFH ; More BASE
A4      = BASE SHR 24           ; Rest of BASE

          DW      A1
          DW      A2
          DB      A3
          DW      ATTRIB
          DB      A4
          ENDM

DSEG      SEGMENT  USE16

;Global descriptor table

GDT      DW      0,0,0,0
          DESCRPT OFFFFFFFH,0,1,0,1,0,1,2
TSIZE    =      $      GDT
```

(continued)

normal to increase the limit when you enter protected mode, but you are supposed to reset the value to 64 KB when you go back to real mode.

But, if you leave the 4-GB limit in place, the processor runs in real mode with a 4-GB memory limit-which is exactly what you want. Now our little program happily increments past 64 KB.

listing 1 shows a program that adjusts the segment limit for a real-mode program. As you can see, it tests the segment limit like the previous program by incrementing EBX past 64 KB. The subroutine SETUP sets the range limit to 4 GB. Here's how it works.

To reset the memory limits, I first create a descriptor which specifies how memory is configured. Although there are several ways to do this, the easiest is to use a global descriptor. To do this, I build a Global Descriptor Table (GDT) in memory that contains all the necessary data.

Each entry or descriptor in the GDT is 8 bytes long. The first descriptor has all bytes set to zero. The second entry controls the memory block's size and attributes. Because the format of a descriptor is convoluted, a macro builds it. Here, I build a descriptor whose base is zero and whose limit is 4 GB.

Once the descriptor table is built, I need to point the Global Descriptor Table register (GDTR) at it. The GDTR requires two pieces of information: a pointer to the table and the table size. The pointer must be a linear **rather than** segmented address. Since the program can be loaded anywhere in the linear address space, I can only get the actual address at **runtime**. I then calculate the linear address of the table.

Before going to protected mode, I turn off interrupts. Without a special set of interrupt routines and tables, the processor crashes on the first interrupt in protected mode. Here, I turn off both regular and nonmaskable interrupts (NMI).

Once the processor is in protected mode, I set one or more segment registers to point to the new descriptor. In protected mode, segment registers are not a part of the memory address. Instead, they point to a descriptor.

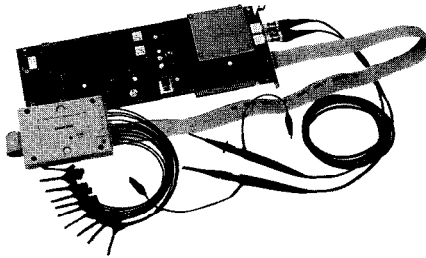
In the example program, the DS register points at the descriptor. Since DS is associated with data transfers, all data transfers that normally use it have a 4-GB range.

Since I don't point ES at the new descriptor, operations associated with this register

## PC-Based Instruments

### 200 MSa/s DIGITAL OSCILLOSCOPE

**HUGE BUFFER  
FAST SAMPLING  
SCOPE AND LOGIC ANALYZER  
C LIBRARY W/SOURCE AVAILABLE  
POWERFUL FRONT PANEL SOFTWARE**



**\$1799 - DSO-28204 (4K)  
\$2285 - DSO-28264 (64K)**

#### DSO Channels

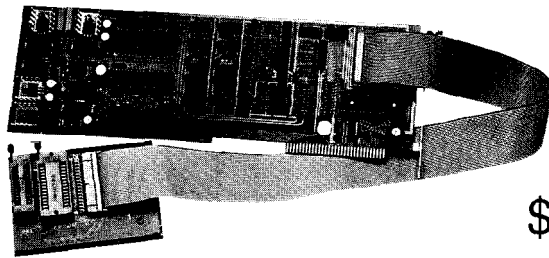
2 Ch. up to 100 MSa/s  
or  
1 Ch. at 200 MSa/s  
4K or 64K Samples/Ch  
Cross Trigger with LA  
125 MHz Bandwidth

#### Logic Analyzer Channels

8 Ch. up to 100 MHz  
4K or 64K Samples/Ch  
Cross Trigger with DSO

## Universal Device Programmer

PAL  
SAL  
EPROM  
EEPROM  
FLASH  
MICRO  
PIC  
etc..

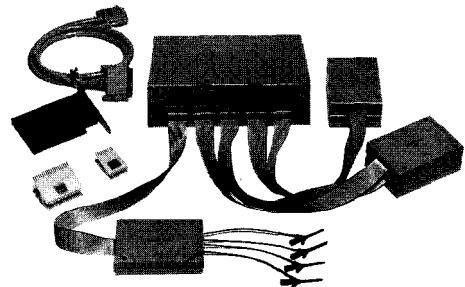


**\$475**

Free software updates on BBS  
Powerful menu driven software

## 400 MHz Logic Analyzer

- up to 128 Channels
- up to 400 MHz
- up to 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



\$799 - LA12100 (100 MHz, 24 Ch)  
\$1299 - LA32200 (200 MHz, 32 Ch)  
\$1899 - LA32400 (400 MHz, 32 Ch)  
\$2750 - LA64400 (400 MHz, 64 Ch)

**Call (201) 808-8990**



**Link Instruments**

369 Passaic Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786



# Your Complete x86 Solution!

## Multitasking in Real and Protected Mode



**smx?** Full-featured, high-performance, preemptive kernel. Customized to x86 processors ideal for demanding applications. Real mode works stand-alone or with DOS, 16-bit, segmented protected mode works with *pmEasy16* or 286 | DOS Extender? 32-bit, flat protected mode works with *pmEasy32* or TNT.

## Protected Mode Environment



**pmEasy.**™ 16- or 32-bit protected mode entry, DPML services, application loaders, *Periscope/32*® and *Soft-Scope*® debugger support

## DOS-Compatible File System



**smxFile.**™ Full-featured file manager. IDE, floppy, and RAMdisk drivers available.

## Dynamically Loadable Modules



**smxDLM.**™ Runs independent executables as tasks which may be downloaded or loaded from disk.

## Networking



**smxNet.**™ TCP/IP stack. Fast UDP. Packet driver interface. NE2000, SMC, and SLIP drivers available.

## Task-Level Debugging



**smxProbe.**™ Provides tracing and symbolic debugging. Works with or without code debuggers.

## C++ Classes



**smx++.**™ Class library built upon smx. Provides fully OOP-compatible kernel interface.

## Extended Memory, Real Mode



**smxEMS.**™ Allows copying data between real memory and extended memory buffers or accessing extended memory via a window.

## User Interface



**smxWindows.**™ Text windowing Dresses up user interfaces.

## Low Cost & Easy to Use



Royalty-free licenses. Supports common, low-cost C compilers and debuggers. Great manuals Source code available.

## Reliable



On market 6 years. 100's of applications. Extensive error checking. 30-day free trial.

**I - 800 - 366 - 2491**  
mdi @earthlink.net

**MICRO DIGITAL, INC**

fax 714-891-2363 Garden Grove, CA USA  
<http://www.earthlink.net/~mdi>

listing 1: continued

;Pointer to the global descriptor table

```
GDTPTRDW    TSIZE-1    ; Define limit
GDTLIN DD    ?          ; Linear
DSEG ENDS
```

;Routine to reset segment limits

```
SETUP: XOR   EAX, EAX    ; Calculate linear address of GDT
      MOV   EBX, EAX
      MOV   AX, SEG GDT
      MOV   BX, OFFSET GDT
      SHL  EAX, 4
      ADD  EAX, EBX
      MOV  GDTLIN, EAX
```

```
LGDT  FWORD PTR GDTPTR ; Load descriptor table
```

```
CLI          ; Disable interrupts
IN  AL, 070H ; Disable NMI
OR  AL, 080H
OUT 070H, AL
```

```
PUSH DS      ; Save DS
MOV  EAX, CRO ; Go to protected mode
OR  AL, 1
MOV  CRO, EAX
JMP  SHORT SETUP1 ; Purge instruction pipeline
```

```
SETUP1: MOV  BX, 8      ; Point to second GDT entry
      MOV  DS, BX      ; Set DS
```

```
MOV  EAX, CRO ; Go to real mode
AND  AL, 0FEH
MOV  CRO, EAX
POP  DS      ; Restore data segment
IN  AL, 070H ; Enable NMI
AND  AL, 07FH
OUT 070H, AL
```

```
STI          ; Enable interrupts
RET
```

```
CSEG ENDS
      END START
```

still have the 64-KB limit. Any segment register except CS can be pointed at the new descriptor, allowing it to access 4 GB of information.

After returning to real mode, I set the modified segment registers to some meaningful value.

Why is this done?

In real mode, the value in the segment register is still added to the offset to form the memory address. If, for example, the registers are set to zero, you get a memory map that starts at zero and runs to 4 GB.

In the example program, I set DS back to its original value. This resetting gives a memory model in which everything is relative to the base address of the current segment. You can still access 4 GB of

memory-it just starts in the middle of memory and wraps around the end.

Because the assembler generates addresses that are relative to a segment base, this technique enables you to access variables created by the assembler without having to convert the segmented address to a linear address.

For the average program, you probably want some segment registers set to zero and some set to the base of the current segment. This way, you can access local variables in the normal way and far data using a linear address.

## D R A W B A C K S

There are a few drawbacks to the techniques described here. For one thing, pro-



**Listing 2:** This program tests protected mode under Windows using the built-in **DPMI**. It must run in a DOS box under Windows running in enhanced mode on a 386.

; The program should be assembled as follows:

```
MASM WINDPMI
LINK WINDPMI
EXE2BIN WINDPMI.EXE WINDPMI.COM
DEL WINDPMI.EXE
```

; To test this program, first go into Windows. Windows must be running in enhanced mode on a '386. From Windows, go to DOS using the "DOS PROMPT" icon. Execute the program by typing ; WINDPMI from the DOS prompt.

```
.386
CSEG SEGMENT USE16
ORG 0100H
ASSUME CS:CSEG,DS:CSEG,ES:CSEG

START: LEA DX,RLMSTR ; Display start-up message
MOV AH,9
INT 21H
CALL DISSEG ; Display current segments

; Release memory back to the DOS memory pool
MOV BX,PRGEND-START+256; Get program size, incl. PSP
MOV CL,4 ; Convert to paragraphs
SHR BX,CL
ADD BX,1 ; Plus 1
MOV AH,4AH ; Set function
INT 21H ; Call DOS

; Test for DPMI installation. If so, get the DPMI information
; 32-BIT MODE MUST BE AVAILABLE FOR OUR TEST
MOV AX,1687H ; Get DPMI function
INT 2FH
OR AX,AX ; DPMI installed?
JNZ NODPMI ; Exit if not
AND BX,1 ; 32-bit mode?
JZ NODPMI ; Exit if not
MOV WORD PTR DPOFF,DI; Save protected-mode switch addr
MOV WORD PTR DPOFF+2,ES

; Allocate a scratch memory block for the DMPI
MOV BX,SI ; Get number of paragraphs needed
MOV AH,48H ; Set up to allocate memory
INT 21H ; Call DOS and allocate memory
JC NODPMI ; Exit if we cannot allocate

; Goto protected mode
MOV ES,AX ; Get base of allocation
MOV AX,1 ; Select 32-bit program
CALL DWORD PTR DPOFF ; Turn on protected mode
JC NODPMI ; Exit if can't go to protected mode

; Protected mode code starts here
LEA DX,PTMSTR ; Display protected mode message
MOV AH,9
INT 21H
CALL DISSEG ; Display current segments

; Create a 4-GB descriptor
; Allocate a local descriptor
MOV AX,0 ; Allocate a local descriptor
MOV CX,1 ; One descriptor
INT 31H
JC PROEXT ; Exit if we cannot allocate
MOV NEWSSEL,AX ; Save selector for new descriptor

; Set descriptor base to zero
MOV AX,7 ; Get function code
```

(continued)

IF YOU DO

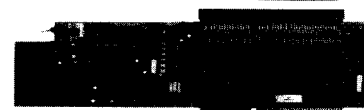
# FUNCTIONAL TESTS

YOU NEED

## HOT SWAPPING ELECTRONIC EXTENDERS

For PCI, ISA, VXI, EISA, VESA, Micro Channel & NuBus. Custom Designs available.

Electronic ISA Extender / PC Mini Extender



Electronic PCI Extender / PCI Mini Extender



Insert/Remove Cards With PC Power On!  
Save Time Testing And Developing Cards:  
Save Wear On Your PC From Rebooting  
Adjustable Overcurrent Sensing Circuitry  
NO Fuses, All Electronic For Reliability  
Single Switch Operation W/Auto RESET  
Optional Software Control Of All Features!  
Breadboard Area For Custom Circuitry  
And More...

Full line of passive extenders:  
PCI, ISA, VXI, EISA, VESA,  
Micro Channel & NuBus

Passive PCI Extender Passive EISA Extender



Passive MC32 Extender Passive ISA Extender



# AZ-COM

3343 Vincent Rd., Ste. D,  
Pleasant Hill, CA 94523

TEL: 1-800-209-2418

FAX: 510-947-1900

CALL OUR 24 HOUR  
FAX ON DEMAND SYSTEM  
510-947-1000

grams written this way are about 20% larger because so many prefixes have to be attached to the 32-bit opcodes. Additionally, the programs may run slightly slower for the same reason.

Finally, although Intel documents the loophole we used to get 32-bit addresses in real mode, it's probably not the way they intended the processor to be used. Even though it works in all versions of the '386, '486, and Pentium, it may not work on future processors.

## WINDOWS AND VIRTUAL MODE

These techniques don't work in some situations. To access the full 4 GB of memory space, you must build newdescriptor tables. Loading pointers to descriptor tables is a privileged operation. It requires that the processor operate at a privilege level of zero, the highest level possible.

Under MS-DOS, the processor is usually in real mode and operating at the highest privilege level. But when DOS runs under Windows enhanced mode, programs execute in virtual '86 mode.

In virtual mode, the processor always operates at privilege level three, the lowest level, so you can't directly load a new descriptor if you are running under Windows. If you try, Windows aborts your program and tells you that system integrity has been violated. For this reason, you cannot use the memory expansion technique with Windows.

This is not an insurmountable problem because Windows has a built-in DOS protected-mode interface (DPMI). DPMI is a standard interface that lets application programs run in protected mode.

In addition, Windows has its own built-in DOS extender. Although the DOS extender is not documented, it handles interrupts and simulates DOS calls. If you need 32-bit processing under Windows, it is relatively easy to take advantage of the built-in DPMI and DOS extender.

### DUAL-MODE PROGRAMS

If your program must run under both Windows and DOS, you can test for the Windows DPMI at the start of the program. If you find the DPMI, the program runs in protected mode. If there is no DPMI, the program runs in real mode using the techniques outlined earlier.

### Listing 2: *continued*

```

MOV    BX,NEWSEL    ; Get selector
XOR    cx,cx        ; Set base to zero base
MOV    DX,CX
INT    31H          ; Set. descriptor base
JC     PROEXT

; Set descriptor limit to 40 GB
MOV    AX,8         ; Get function code
MOV    BX,NEWSEL    ; Get selector
MOV    CX,0FFFFH    ; Set limit to 4 GB
MOV    DX,CX
INT    31H
JC     PROEXT

; Test protected-mode memory limits by accessing beyond a segment
; boundary
LEA    DX,NSLSTR    ; Display message
MOV    AH,9
INT    21H

MOV    AX,DS        ; Get current selector
MOV    OLDSEL,AX    ; Save it

MOV    AX,NEWSEL    ; Get the new selector
MOV    DS,AX        ; Use with DS
MOV    EBX,0FFFFFFH ; Point beyond 64K
MOV    AX,[EBX]

MOV    BX,CS:OLDSEL ; Restore old selector
MOV    DS,BX
CALL  WRDOUT        ; Display memory value
CALL  SPACE
MOV    AX,NEWSEL    ; Display new selector
CALL  WRDOUT
CALL  CRLF

; Exit from protected mode using DOS exit
PROEXT: MOV    AX,04C00H ; Get exit function
INT    21H          ; Call DOS and exit

; Execution comes here if we are unable to go to protected mode
; for any reason
NODPM: LEA    DX,NPMSTR ; Display error message
MOV    AH,9
INT    21H
RET

; Variable storage for the program
DPOFF  DD    ?        ; Far address of protected-mode switch
ENTRY  POINT
NEWSEL DW    ?        ; New protected-mode selector
OLDSEL DW    ?        ; Old protected-mode selector

```

You now know how to make 32-bit operations work in real mode. But, there are a few things you must do to make the same techniques work in protected mode.

First, you need the DPMI to build protected-mode descriptors that allocate and define the memory your program needs.

Second, the descriptor for your program must default to 16-bit operations. Otherwise, the prefix byte for 32-bit instructions has the wrong effect.

Listing 2, for example, puts you into protected mode using a DPMI. It runs under Windows 3.1. The program:

- tests for a DPMI
- allocates memory for the DPMI
- goes into protected mode.

Once it is in protected mode, it creates a 4-GB descriptor and verifies that the memory limit has been expanded by loading from

memory location `OFFFh` (well beyond the real-mode 64-KB boundary).

The program also prints the value of the CS and DS registers in both real and protected mode. When you run the program, you discover that the values of these registers are different in the two modes.

Why?

In protected mode, segment registers are not part of the address—they are pointers to descriptors. This difference makes it easy to verify that the program is truly running in protected mode.

Notice that the program calls two DOS functions from protected mode. This operation would be impossible without the DOS extender built into Windows. It intercepts and handles all protected-mode calls to DOS and BIOS. It is probably safe to use these functions, but since they are undocumented, there is always the risk that they could be changed down the road.

When you write protected-mode programs, debugging can be difficult. If you make the slightest error, Windows aborts **your program**, saying only that it has violated system security. As well, most debuggers don't work in protected mode.

If, for example, you try to debug the program in Listing 2 using a real-mode debugger, the real-mode portions of the program work fine. But strange and unpredictable things happen when you try to go to protected mode.

The solution?

Find a protected-mode debugger or program the protected part of the software very carefully.

Once you are in protected mode, the **DPMI** provides several support functions for the interface between protected-mode programs and DOS. The features of the **DPMI** are described in detail in the DOS Protected Mode Interface (DPMI) Specification, available free from Intel.

## MEMORY MANAGEMENT

Memory managers like **HIMEM** or **QEMM** can cause problems with the techniques we're using.

Under some circumstances, a memory manager may run in protected mode while DOS is running in V86 mode. It can then use the memory-management features of protected mode to put blocks of RAM into memory above the 640-KB boundary.

But, when the processor is in V86 mode, our programs can't switch to protected

mode to expand the segment limits. To make this work, simply avoid using a memory manager or carefully configure the memory manager so it doesn't use V86 mode.

Normally, a memory manager switches DOS to V86 mode when it loads a program to high memory. You may be able to avoid problems by not allowing the memory manager to load any programs into the **memory space** between 640 KB and 1 MB.

You can also use a memory manager that supports **DPMI** or **VCPI** (Virtual Control Program Interface). **VCPI** is another protected-mode interface for DOS that is similar to **DPMI**.

If the memory manager supports either the **DPMI** or **VCPI** specification, you can use the same techniques used with Windows.

## READY TO GO?

Learning to program in protected mode can be difficult. I hope the techniques discussed here help you overcome some of the rough spots.

The sample programs in this article should give you a starting point for writing both real- and protected-mode programs. Even if you never use the techniques outlined in this article, you should have a better understanding of the intricacies of the 386. **EPC**

*Larry Fish* has been designing hardware and software for more than **twenty** years. Currently he works as a consultant designing embedded systems and CAD **software**. *He may be reached at [lfish@nyx.cs.du.edu](mailto:lfish@nyx.cs.du.edu).*

### SOURCES

**DOS Protected Mode Interface Specification**  
Intel Order Number 240763.001

Intel Literature JP26  
3065 Bowers Ave.  
P.O. Box 58065  
Santa Clara, CA 9505 1-8065  
(800) 548.4725

### REFERENCES

Crawford, John H. and Patrick P. Gelsinger. *Programming the 386*. Sybex: CA. 1987.

Duncan, Roy, et al. *Extending DOS*. Ed. Roy Duncan. Addison Wesley: Reading, MA. 1990.

## IRS

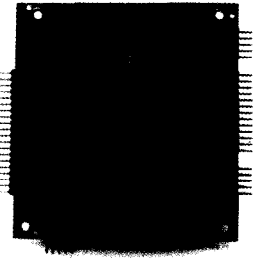
- 413 Very Useful
- 4 14 Moderately Useful
- 415 Not Useful

# rttd® Sets the Pace in Low Power, High Performance PC/104 Technologies



50 MHz  
486 SXLC2  
**\$543**

**Features:**  
486SXLC2  
50 MHz  
387SX  
8K cache  
2MB DRAM  
3W typical  
SSD  
EEPROM  
WDT  
IDE & FDC  
2 Serial  
EPP  
PS/2 mouse  
Keyboard  
Quick Boot  
Virtual devices



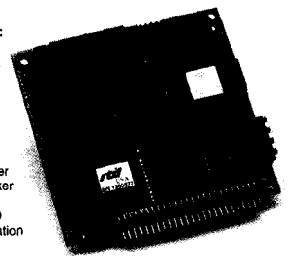
### CMI486SXLC2-1 Fully Integrated PC-AT with Virtual Device Support

When placing your order, mention this ad and receive a 387SX math coprocessor FREE!



200 kHz  
12-bit DAS  
**\$498**

**Features:**  
200 kHz  
12-bit A/D  
16S/E/8D  
Scan  
Burst  
Multiburst  
1K CGT  
Triggers  
FIFO buffer  
Data marker  
12-bit D/A  
16-bit D/O  
+5V operation



### DM5408-2 200 kHz Analog I/O Module with Channel-Gain Table

Make your selection from:

#### 9 cpuModules™

SuperXT™, 386SX, 486SXLC2, and 486DX2 processors. SSD, 8MB DRAM, RS-232/422/485 serial ports, parallel port, IDE & floppy controllers, Quick Boot, watchdog timer, power management, and digital control. Virtual devices include keyboard, video, floppy, and hard disk.

#### 7 utilityModules™

SVGA CRT & LCD, Ethernet, keypad scanning, PCMCIA, intelligent GPS, IDE hard disk, and floppy.

#### 18 dataModules®

12, 14 & 16-bit data acquisition modules with high speed sampling, channel-gain table (CGT), sample buffer, versatile triggers, scan, random burst & multiburst, DMA, 4-20 mA current loop, bit programmable digital I/O, advanced digital interrupt modes, incremental encoder interfaces, opto-isolated digital I/O & signal conditioning, opto-22 compatibility, and power-down.

## rttd® Real Time Devices USA

200 Innovation Boulevard • P.O. Box 906  
State College, PA 16804-0906 USA

Tel: 1 (814) 234-8087 • Fax: 1 (814) 234-5218  
FaxBack®: 1 (814) 235-1260 • BBS: 1 (814) 234.9427

### RTD Europa RTD Scandinavia

Budapest, Hungary Helsinki, Finland  
Fax: (36) 1 212-0260 Fax: (358) 0 346.4539

RTD is a founder of the PC/104 Consortium and the world's leading supplier of PC1104 CPU and DAS modules

Ken Prada

# PC/104 Embedded Systems in Oceanographic Instruments

**In** ancient days, before PCs, oceanographic research used embedded systems. Ken traces the evolution of those systems to today when many are based on PC/104 architecture.

Embedded systems are not new to oceanographers. Before microprocessors, embedded systems were used extensively in oceanographic instruments. Because of isolated locations, many oceanographic systems must be autonomous and operate for extended periods without support. Embedded architectures enable data sampling, recording, and telemetry.

Oceanographic systems are of two basic types: those used aboard ships or similar large platforms, and those used autonomously, such as buoys, ocean-bottom instruments, and untethered vehicles. The two groups differ significantly in design and operation.

Aboard ship, a power cord is available and usually someone monitors operations, makes changes to procedures, changes recording media, monitors data quality, and spills coffee on the keyboard.

In a buoy or underwater instrument, however, there is no AC cord, no operator, no keyboard, and the coffee is weak and salty. These are truly autonomous units.

They require very low-power embedded systems.

When deployed, they usually remain unattended for long periods. Throughout their operational life, these instruments make strategic sampling decisions, handle large volumes of storage or telemetry, and monitor and adjust power consumption, while accomplishing complex control and data acquisition tasks.

A variety of embedded processors and systems are used in ocean instruments, and many are commercially available. Processor and system choices are based on individual instrument needs and power limitations. Most selections provide reliable, competent, and low-power operation.

However, as the requirements for autonomous systems expand, the extended capabilities and features found in the PC/104 architecture provide distinct advantages. CPU, I/O, software, and operating systems are an easy link to the desktop environment.

PC/104 has become critical in the design of extended modern ocean instruments because of its:

- low development costs
- performance growth in processor capability and memory size
- compatibility with standard storage devices
- availability of off-the-shelf functions
- software development environment

In this article, I'll start by listing standard sensor systems and their tasks to give you a flavor of the broad range of oceanographic embedded applications. Bear in mind that this list represents only a small sample of the instrument types used in oceanography. I'll then describe a specific system which emphasizes how PC/104-based embedded systems enhance ocean research.

Unlike most embedded systems, in oceanographic instruments power consumption is a critical issue. Many ocean-

bound sensor systems must operate for extremely long periods without servicing, and in some cases, the systems are expendable. Due to size and weight restrictions within each instrument, battery stacks are limited. Yet, PC/104 typically requires more power than many other embedded architectures.

Hence, to take advantage of the PC/104 architecture, special attention must be devoted to power-consumption considerations. I also discuss one solution to the power problems.

### EMBEDDED SENSOR SYSTEMS

Few, if any, modern oceanographic instruments exist that do not use some sort of embedded intelligence. In addition to commercially available instruments and sensor systems, engineers and scientists have designed many one-of-a-kind systems for specialized tasks. The wide variety of such unique applications includes:

- buoys which measure surface-meteorological variables such as air temperature, humidity, barometric pressure, incident and reflected radiation, and precipitation
- buoys or moorings which have instruments attached to their mooring cables that measure and record water temperature, conductivity, and current flow at various depths
- buoys which use acoustic signals over a broad frequency range (38-1 000 kHz) to measure biological activity of various-sized organisms from small plankton to large fish
- ocean-bottom systems which measure and record variations in bottom sand or sediment that is caused by animals or currents sweeping the ocean floor. (These

measurements are made using high-frequency acoustics and photography.)

- ocean-bottom systems which record seismic activity
- autonomous small vehicles which expand spatial sampling by carrying sensors to places not easily reached by samplers lowered or towed from ships.

### BEFORE PC/104

The RCA COSMAC 1802 was among the first early embedded-system controllers. Though limited in capability, this flea-power processor produced a new and exciting generation of intelligent instruments. The Motorola series of low-power controllers later expanded instrument capabilities. These controllers are still an integral part of many ocean instruments systems.

In the early 1980s, instrument users needed greater arithmetic capabilities, more complexity in control and sampling operations, and increased information-storage space or telemetry bandwidth. Capabilities beyond simple microcontrollers were clearly needed.

In 1982, we developed a system at Woods Hole which measured and recorded real-time ambient-noise spectra in the ocean. Its controller was a National NSC800, and it was based on the CP/M operating system. The control program was written in BDS C. Frequency spectra were produced with an Intel 8086/8087 combination as a DSP unit.

This project shaped many of the goals for future systems. It showed the benefits of working with more capable microprocessors, the advantages of an embedded operating system, and the wonders of C as a language for embedded applications.

However, even this system, and certainly the newer 16-bit processors, were power-hungry creatures waiting for an opportunity to stop the Energizer bunny. We needed a low-power solution...

### THE 80x86 PC CONNECTION

In the mid 1980s, several things sparked greater interest in embedded PCs. These improvements included:

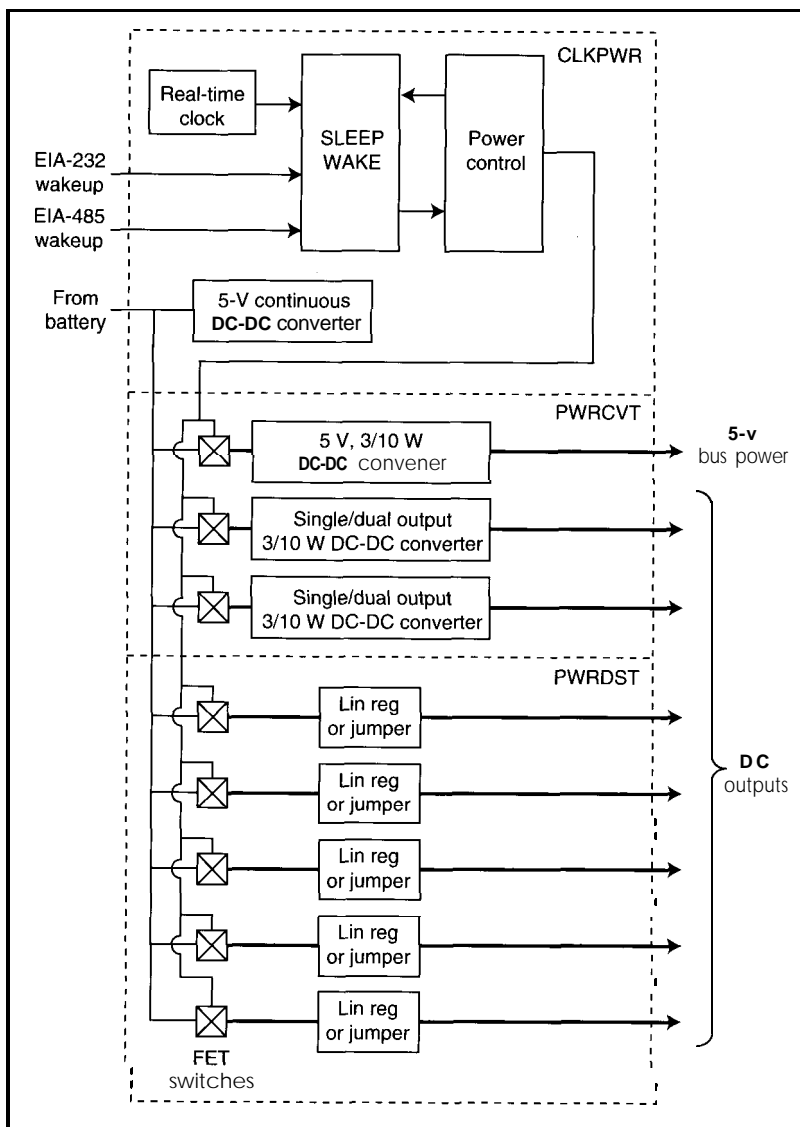


Figure 1: The power-control boards provide several switched single or dual voltages for system or peripheral support. Sleep and wake functions are also available.

Embedded applications include vehicle control, data sampling and logging, and video-frame control and capture.

All of these systems depend on embedded microprocessors and modern storage technology or intelligent telemetry. In fact, in the last two decades, embedded intelligence has provided the most important enabling technology for advances in ocean sensors, systems, vehicles, and platforms.

- the availability of CMOS replacements for standard 74xx TTL functions
- the introduction of the Harris line of CMOS ICs which included substitutes for the 8088 family
- the expanded availability of other CMOS products including EPROMs and static RAM
- the growth of MS-DOS as a well-supported single-user operating system
- the appearance of mass-storage products compatible with the PC architecture and DOS
- the appearance and growth of competent C compilers for application development

These events produced an ideal environment for advanced low-power, sensor-recording systems.

## THE VERY LOW-POWER PC

In 1986, I found an 8088 single-board computer that plugged into a passive PC

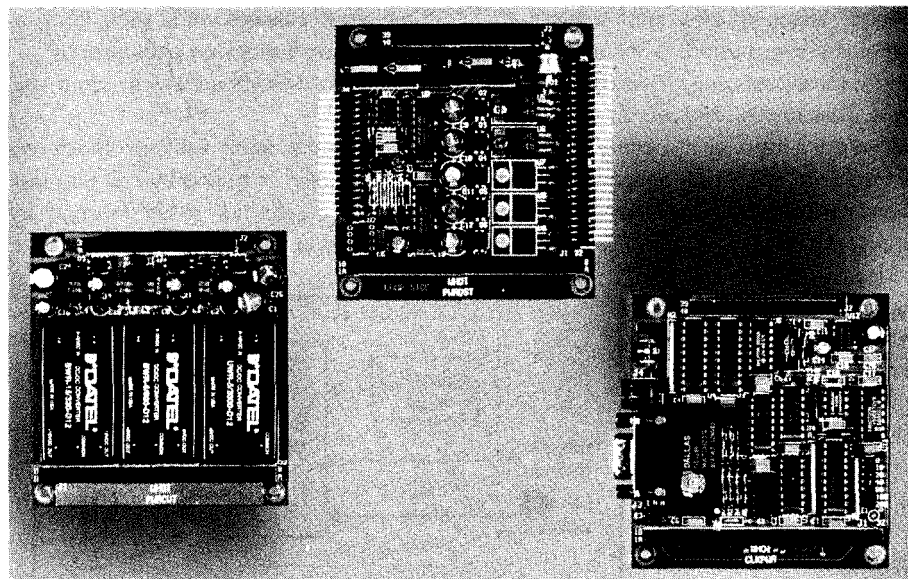
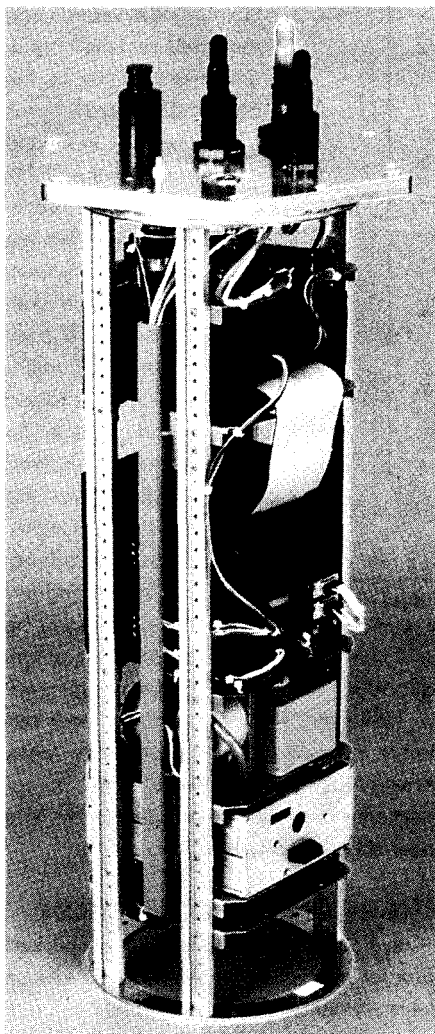


Photo 1: Power control uses a three-board set providing **DC-DC converters**, **linear regulators**, switched control, and distribution through **various connectors**.

backplane. I repopulated the entire board with CMOS (HC, HCT) substitutes for the TTL74xx chips and Harris CMOS substitutes for the 8088.

These substitutions produced an operational PC with an extremely low power drain. I designed a static memory board and low-power peripheral I/O board (serial, parallel, and A/D converter) compatible with the PC bus. A custom BIOS enabled DOS to run on this system.

The result was LOPACS (low-power, acquisition-control system), a hardware- and software-compatible PC that operated at 0.5-W power consumption.

An optical disc drive (WORM) was added to the system which provided 125 MB of storage, an unheard-of amount of data space for that time. Additionally, the disc cartridge could be removed from the sensor system and read on a DOS system with a similar WORM drive, controller, and driver. The file structure on the WORM cartridge was DOS compatible.

A drawback to LOPACS was its standard PC physical structure. The size and shape of the combined PC processor board and passive bus were not easily packaged for deep ocean applications. But, our appetites for better high-performance, stan-

dardized systems (preferably also PC-compatible) had been whetted.

## PC/ 104

PC/104's technology and architecture provided an answer. Its architectural features (deal for industrial applications) make it even more important for ocean-sensor applications.

As PC/104 has matured over the past few years, many exciting and useful functions have been introduced by many manufacturers. Support is available, and PC/104 is here to stay.

With PC/104, the PC's features and ease of use, development, and testing move into an autonomous instrument.

## POWER CONSIDERATIONS

Autonomous oceanographic systems derive power from a variety of battery types. Most systems use stacks of alkaline cells, typically 15 V. Where possible, surface buoys use lead-acid gel cells and solar panels. Autonomous vehicles use lead-acid technology with recharge facilities at home base. It's critical to get the longest acceptable performance from the battery stack without compromising the system's mission.

Even the lowest-power PC/104 processor board requires an energy budget that is larger than we'd like. To use the technology with a limited power budget, special power-control circuits are needed.

I designed a three-board PC/104 stack that provides several switched voltages

Photo 2: The **subsurface electronic unit fits into an 8" pressure cylinder**. A mck assembly attached to the top cover of the pressure cylinder contains the **PC/104** components and various other modules and sensor electronics.

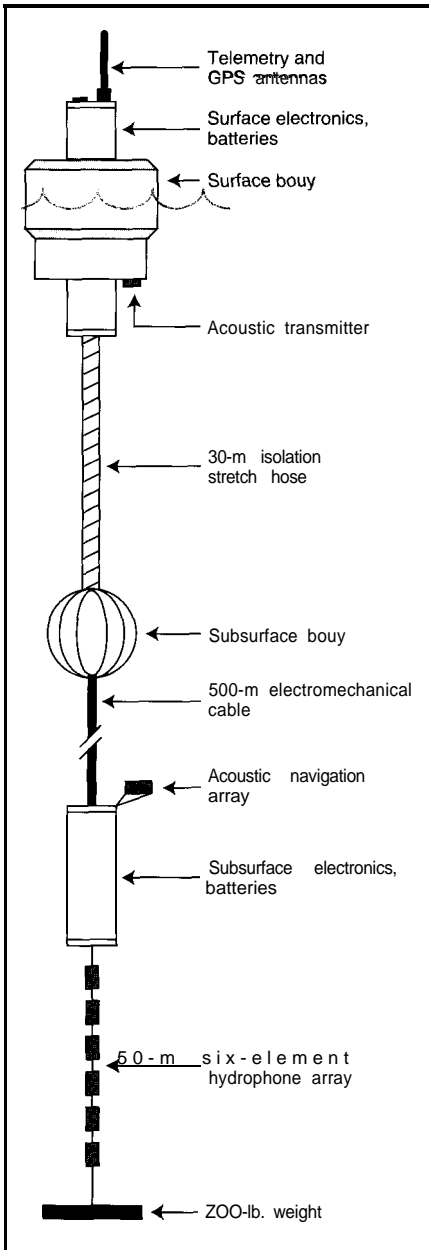


Figure 2: The *space-suspended acoustic receiver* uses a surface buoy with telemetry and GPS capabilities and a subsurface unit that receives and processes acoustic information. Each unit has its own PC/104 stack to control all functionality.

from a single 9-l 8-V battery stack and provides power control for the system itself. Figure 1 shows these boards in a block diagram, while Photo 1 shows you what they really look like.

The CLKPWR board uses the Dallas Semiconductor DS1286, a self-contained real-time clock with alarm and watchdog outputs. The processor can shut itself down. Wakeup is available from three sources: the RTC alarm output, EIA-232 input, or EIA-485 input. Power consumption in the shutdown mode is less than 7.5 mW. Two

8-bit latches provide control for FET switches on the other boards.

The PWRCVT board contains provisions for three DC-DC converters. These can be either 3- or 10-W modules [Datel XWR series or equivalent]. One module provides 5 V to the PC/104 bus. The other modules provide single- or dual-output power for a variety of needs. Onboard filters achieve clean voltages for analog needs. Battery input to these modules is FET switched and controlled from the CLKPWR board.

The PWRDST board provides a series of FET-switched voltages that are powered by direct battery power or standard 3-pin linear regulators. These outputs are also controlled from the CLKPWR latch signals.

The design of any system is usually a compromise between needed processing capabilities and power consumption. For applications where processor horsepower is not critical, there are some excellent low-power processor boards.

One of the recent additions to this group is the CoreModule/PC from Ampro. This board has an average power consumption of less than 0.75 W with no keyboard or serial device connected.

Additional powerconservation is gained from I/O boards with low-power operation. Using the 82C50 UART, I designed a two-channel serial I/O board that consumes less than 100 mW.

One of this board's power-saving tricks involves gated oscillator signals to the UART. The OUT1 signal from each UART gates the oscillator to the UART. Drivers enable and disable the oscillator as needed. When both UARTs are idle, the oscillator itself is disabled. Each of these steps saves only a small amount of power, but the cumulative effect over long periods can be substantial.

Some functions in embedded systems require considerable power but are not needed at all times (e.g., a digital signal processor).

We recently designed a switched-bus extender that allows power-hungry functions to be powered and connected to the PC/104 bus only when needed. The bus extender is addressable and several may coexist in any system.

In the system I'll describe in moment, the DSP used for signal processing probably uses as much power as the other system components combined. By isolating it on a switched bus, we ensure that it is connected

# BIG THINGS COME

## DOMINO

Microcontroller

**DOMINO** Microcontroller

MODEL \_\_\_\_\_ SN \_\_\_\_\_

Starting at  
**\$79**

Micromint's  
**Domino-S2**  
microcontroller is a  
"supercomputer" in less than  
**0.75 cubic inches.** We've packed  
the most essential elements into  
one tiny package. Domino is a  
plug-and-go module, just attach  
+5V and a terminal or network.  
A simple keyed sequence saves  
an autostarting program in  
nonvolatile memory.

SPECIAL FEATURES

- 80C52 with ROM-resident, full floating-point BASIC
- 32K bytes SRAM and 32K bytes EEPROM
- Two PWM outputs, I<sup>2</sup>C bus
- Serial I/O: (up to 19,200 bps) RS-422, RS-485 & RS-232A
- Two interrupts and three timers
- Parallel I/O: 12 bits, 3 shared with ADC and I<sup>2</sup>C
- Power: +5V @ 15 mA;
- Size: 1.75" x 1.062" x 0.4" potted
- A/D converter: 2 channels, 12 bits, 10k samples/sec.
- Connections: via 2-10, 0.1" dual-row header
- -20°C to 75°C operating temperature
- Industrial temperature available

4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(860) 871-6170 • Fax (860) 872-2204

For Borland C/C++, Microsoft C/C++, Borland Pascal

# RTKernel

## Real-Time Multitasking for DOS

RTKernel is a professional, high-performance, real-time multitasking system for **MS-DOS** and **Embedded Systems**. It can use DOS device drivers and BIOS, and runs other DOS applications as a task - even **Windows!**

RTKernel is **loaded with features**: an unlimited number of tasks, excellent performance, a full set of inter-task communication functions (semaphores, mailboxes, synchronous message-passing), real and protected mode support, drivers for up to 38 COM ports and Novell's IPX services, and lots more...

It's **ROMable** and very **compact** (about 16K code, 6K data), making it ideally suited for Embedded Systems.

RTKernel is well-documented and easy to use. All hardware drivers always come with source code; kernel **source code available** at extra charge. No **run-time royalties**.

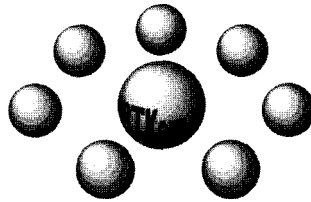
**join thousands of satisfied customers!**

In North America, please contact:

**On Time Marketing**  
88 Christian Avenue Setauket, New York 11733 USA  
Phone (516) 689-6654 Fax (516) 689-1172  
BBS (516) 689-6285 FaxFacts (516) 689-6315 CIS 73313,3177

From other countries, **please contact:**

**On Time Marketing**  
Karolinenstrasse 32 · 20357 Hamburg · GERMANY  
Phone +49-40-437472 · Fax +49-40-435196  
CompuServe 100140,633



Use **RTKernel** for:

- process control
- ▶ data acquisition
- ▶ real-time simulations
- background processing

Libraries: **\$495**  
Source Code: add \$445



**Time**  
**MARKETING**

Professional Programming Tools

**Free demo disk!**  
**Internet:**

Request Info Kit!  
<http://www.on-time.com>  
<ftp://on-time.com>  
[info@on-time.com](mailto:info@on-time.com)

topowerand bussignalonlywhen needed. This saves a large amount of power and extends instrument life significantly while still providing the processing power needed.

All of these power-saving methods reduce overall long-term power consumption to a level consistent with mission constraints. While each step may not seem substantial, they produce significant power savings.

### PACKAGING

The PC/104 architecture is **ideally suited** to packaging in the ocean-systems environment. Most underwater instruments and systems are packaged in pressure bottles. The bottles are typically cylindrical containers fabricated from tubing (aluminum, stainless, titanium) of varying wall thicknesses, depending on depth requirements. Inside diameters vary but typically range from 6" to 8".

The PC/104 form factor with its stacking bus fits easily into these containers. As you can see in Photo 2, the embedded system is often attached to an end cap so that it is removed when the cap is detached. Since the tube is just a cover, assembly is easy. Wiring is simple and convenient because the end cap usually contains connectors for power, signals, and communications.

### MEASURING GLOBAL OCEAN TEMPERATURE

I'd like to describe a **PC/104** application we developed recently at Woods Hole. It is a complex system which records variations in global ocean temperatures. The system was designed to detect temperature variations over extended periods of time (i.e., years) by measuring acoustic travel **time over** long ranges. Here's how **it works**.

At predefined intervals, a low-frequency acoustic energy source transmits a coded tone. Acoustic receivers at various locations record the tone's arrival time. Variations in travel time over long periods indicate variations in average temperature of the intervening water. Autonomous drifting sensors are one type of acoustic receiver.

The drifting receiver, called SSAR (Surface Suspended Acoustic Receiver), uses a surface buoy and a subsurface receiver suspended 500 m below (see Figure 2). The units are electronically connected by a two-wire EIA-485 link that is part of the support cable. Each unit contains an **em-**

C Compilers of choice —  
for the chip of your choice!

- Fast, efficient optimizing compilers
- Chip specific
- Built-in assembler
- Integrated Development Environment
- Linker, librarian

We respond to your  
**C compiler needs.**



Byte Craft Limited  
421 King St. N., Waterloo, Ontario  
CANADA N2J 4E4  
(519) 888-6911  
Fax: (519) 746-6751  
BBS: (519) 888-7626

**MPC** for PIC16/17Cxx

**C38** for MELPS 740

**COP8C** for COP8

**C6808** for MC68HC08

**Z8C** for Z8

**C6805** for MC6805

#209



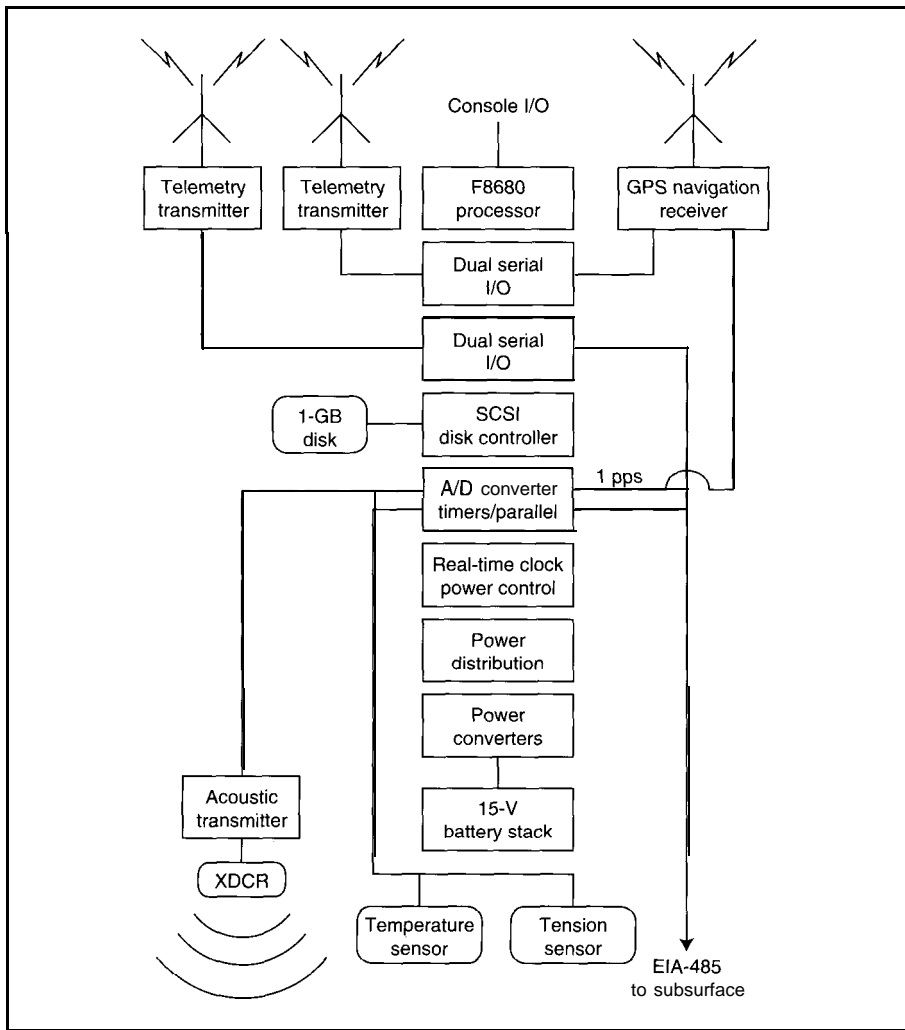


Figure 3: The **SSAR surface-unit PC/104** stack controls redundant telemetry systems, navigation using GPS, acoustic navigation **transmitter**, and other sensors. Prototype systems included large-volume disks to record engineering and test data.

bedded processor which handles its specific tasks. Each unit also has its own battery stack.

Figures 3 and 4 are block diagrams of the surface and subsurface units. They show the large-volume disk storage used in the prototype and test units but not intended for use in the final, expendable configurations.

The surface unit awakens at scheduled intervals. The Global Positioning System (GPS) receiver is activated and an accurate navigation position is derived (post-processing guarantees 1 O-m accuracy). The internal real-time clock is set to the accurate time from the GPS receiver. An accurate 1-Hz signal from the GPS unit synchronizes a local 32-kHz counter to provide very accurate millisecond timing.


When surface system housekeeping is complete, the subsurface system is awakened by sending a single character over

the EIA-485 link. When the subsurface system has completed its boot operation, full communications are established between systems. Accurate time is sent to the subsurface unit and synchronized by sending the GPS 1-Hz signal over the EIA-485 link.


The position of the receiving hydrophone array must be known if acoustic arrival time is calculated precisely. Wind drift of the surface unit, surface and subsurface currents may separate the units. The exact location of the receiving array relative to the surface unit is determined by a short-baseline navigation system combined with tilt sensors and compass.


The acoustic navigation system uses a transmitter at the surface unit triggered by a pulse sent over the EIA-485 link. Signals are received by a 4-channel transducer on the subsurface unit. The DSP is powered on and connected to the PC/104 bus. It deter-

**IN SMALL PACKAGES**

**BLACKJACK**  <sup>TM</sup>

**TELECONTROLLER**



**BLACKJACK**  <sup>TM</sup>

TELECONTROLLER

MODEL \_\_\_\_\_ SN \_\_\_\_\_

**Starting at \$139**

Micromint's **BlackJack-552** is a true "telecontroller" incorporating both microcomputer and FCC-certified Xecom modem in a single package. **BlackJack** comes preprogrammed with firmware utilities which are optimized for assembly language, C, and BASIC programs. Like **Domino**, **BlackJack** is easy to use. Attach power and a terminal, then upload, store, and execute your program.

#### SPECIAL FEATURES

- 80C552 processor with firmware monitor, 14.7456 MHz
- Serial I/O: (up to 38,400 bps); full-duplex TTL, 1<sup>2</sup>C bus
- Connection: 1.2" dual-row 48-pin DIP header
- Parallel I/O: 20 bits, 4 bits shared with RTC
- Two interrupts: three timers, watchdog timer, two PWM outputs
- Hardware real-time clock
- A/D converter; 8 channels, 10 bits, 20k samples/sec.
- 2400 bps data modem (data/fax available)
- Power: +5V @ 55 mA;
- Size: 2.75" x 1.375" x 0.5" potted
- 32K bytes SRAM, 32K bytes ROM and 128K bytes EEPROM
- Industrial temperature available



4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(860) 871-6170 • Fax (860) 872-2204

mines the exact hydrophone array position by processing the acoustic navigation arrivals with array and package tilt information.

The subsurface unit now awaits reception of the scheduled low-frequency (75 Hz), long-duration coded tones. During the reception period, the I-Hz GPS signal is sent over the EIA-485 link to assure millisecond timing accuracy for the received tones.

The received signals are processed for accurate arrival times. Next, the subsurface unit sends array-navigation and tone arrival-time information to the surface unit over the EIA-485 link. The subsurface system then puts itself to sleep.

The surface unit combines the information from subsurface operations with GPS navigation fixes taken at the start and end of the receiving period. These data are combined with system-performance parameters, battery-condition data, and error information.

Formatted information frames suitable for telemetry are produced using the low-bandwidth ARGOS satellite system. The frames are loaded into autonomous telemetry transmitters, which also contain small embedded systems.

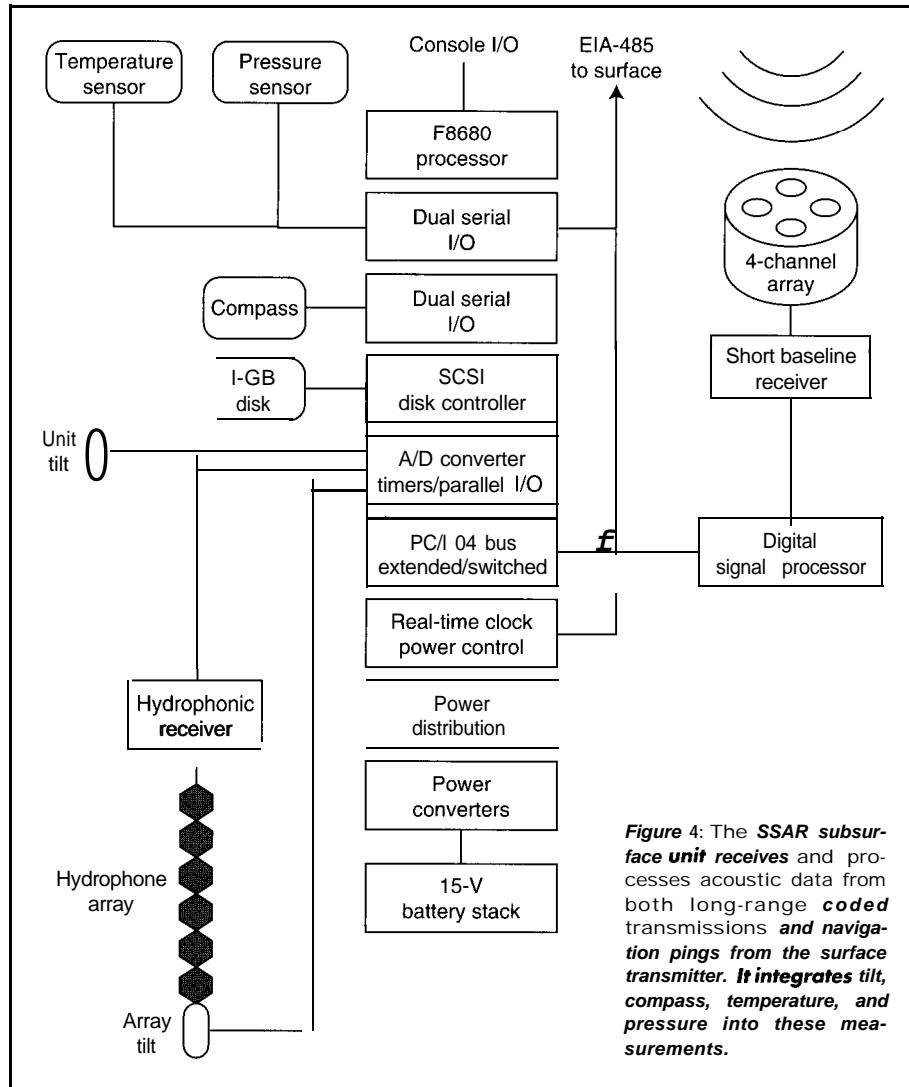
The surface system then calculates the next operational time, sets the clock for wakeup, and goes to sleep. Meanwhile, the intelligent telemetry transmitters continue to send information to shore using rotating buffers and multiplatform IDs.

During development, surface and subsurface units were equipped with Ethernet boards and connected to a server using PC-NFS. This procedure enabled several engineers in different locations to develop and test code in a group environment.

Each engineer had a desktop PC networked to the server. Executables loaded directly from the server into the benchtop prototype units for testing. Such development features are possible because of the PC/104 architecture. Productivity increased significantly over prior embedded architecture environments.

## CONCLUSIONS

The SSAR system described above would not have been possible five years ago. PC/104 technology meets the various complex operational characteristics of this system.



**Figure 4:** The SSAR subsurface unit receives and processes acoustic data from both long-range coded transmissions and navigation pings from the surface transmitter. It integrates tilt, compass, temperature, and pressure into these measurements.

In more recent oceanographic systems, PC/104 permits in situ fast processing, real-time strategic sampling, real-time vehicle control, and many other great things PC systems do on dry land.

The PC/104 architecture has made it possible to package modular, complex, and versatile systems within standard oceanographic instrument housings (6-8" pressure cylinders).

low-power processors, peripheral-board functions, and modern batteries produce systems with capabilities and durations that meet modern measurement needs.

Compatible mass-storage devices provide the space needed for extended sensor deployments—a recent non-PC/104 system deployed more than 14 GB of SCSI disks in an acoustic receiving array!

Add to all this a mature operating system (DOS) and compatibility with desktop development tools and post-experiment processing, and what is the result?

We can build oceanographic sensor and control systems that expand productivity and capabilities, leading us to a better knowledge and understanding of the oceans. PCQ.EPC

Special thanks to the Department of Defense Advanced Research Projects Agency (ARPA) for providing funding for development of the SSAR system.

Ken Prada is a principal engineer in the Applied Ocean Physics and Engineering department at the Woods Hole Oceanographic Institution. He manages the instrument and systems development laboratory. Ken may be reached at (508) 289-2711 or at [kprada@whoi.edu](mailto:kprada@whoi.edu).

## IRS

4 16 Very Useful

417 Moderately Useful

418 Not Useful

Applied PCs

Russ Reiss

# Embedded PC Buses and CPU Boards

*This month, Russ compares various bus options for embedded PCs. ISA, EISA, VLB, and PCI buses mix with newer standards such as PC/104 and IndustryPack. Tips for choosing the most suitable bus round things off.*

The main advantage to using embedded PCs over other solutions is clear-similarity to desktop PCs. You get:

- user and designer familiarity
- wide availability of hardware, software, and interfaces
- rapid development cycles
- future expandability and supportability

These characteristics are all important elements for achieving short time to market, user acceptability, and a successful product!

Building on my last column (*INK 62*), I'll spend some time on embedded PC standards. I'll look at the most popular bus options and processor boards, keeping in mind what these options mean in terms of system design size, weight, packaging density and flexibility, and cost. You'll soon see there are many approaches and that no one option is right for all situations.

## WHICH BUS TO RIDE?

The choice of which embedded PC to use boils down to selecting a bus standard for your system.

Bus **primarily dictates** form factor. It also determines overall size, packaging density, robustness of mounting, input/output connections, cooling, ease of board replacement, expansion options (and how they combine), system speed, and cost.

Busess represent an evolution. Most buses began rather simply, often as 8-bit versions only. As CPU technology evolved, bus designs have been forced to adapt to keep pace. Now, nearly all buses support 16-bit pathways. Some buses support 32-bit pathways for special ports transfer or for convenient expansion modules.

Let's look at the evolution of modern embedded-system bus options we have in building an embedded PC system.

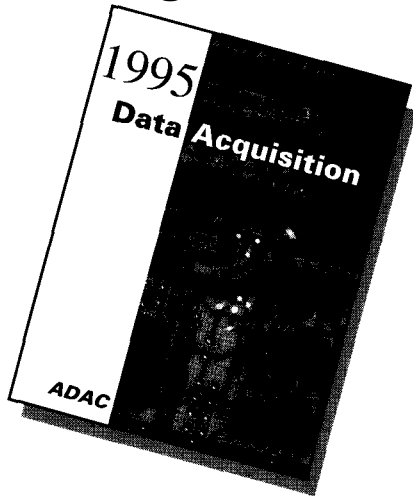
## EARLY BUS ROOTS

Before the IBM PC, its clones, and embedded compatibles, t computers. In particular, desktop personal computers, typically S-100-based and other proprietary formats, usually ran the CP/M operating system.

In those days, commercially available embedded systems more often than not used the ubiquitous Zilog Z80 or Motorola 6800 CPUs or their improved versions. Embedded systems were offered by a host of vendors, the most popular system designs based on Multibus, STD bus, and VME bus. There was no explosion in personal computing and no real software standard for embedded designs.

With the coming of the PC, all that changed. Personal computers standardized on the 8088 and 8086 chips (leading later to the '286, 386, & 486 and Pentium), and PC-DOS or MS-DOS became the de facto operating system. Old Z80 systems

# FREE Data Acquisition Catalog



**PC and VME data  
acquisition catalog  
from the inventors of  
plug-in data acquisition.  
Featuring new low-cost  
A/D boards optimized  
for Windows,  
DSP Data Acquisition,  
and the latest  
Windows software.  
Plus, informative  
technical tips and  
application notes.  
Call for your free copy  
1-800-648-6589**

## ADAC

American Data Acquisition Corporation  
70 Tower Office Park, Woburn, MA 01801  
phone 617-935-3200 fax 617-938-6553  
info@adac.com

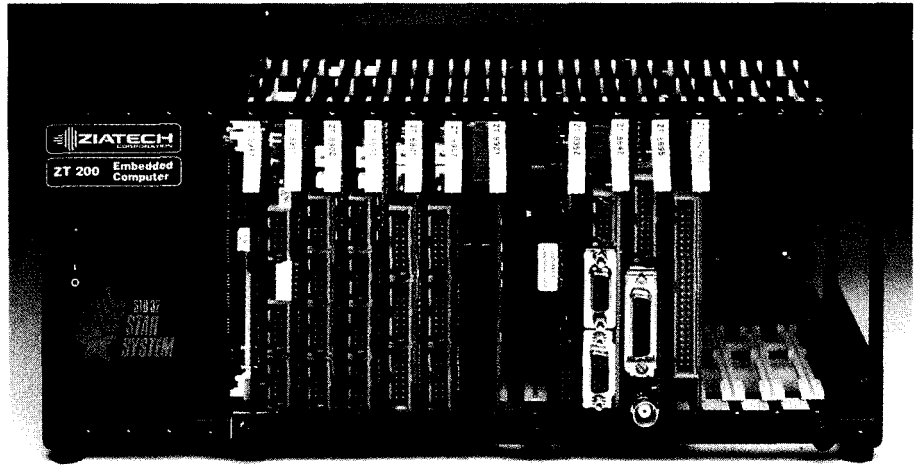


Photo 1: A compact and powerful embedded system can be built around **STD** cards, cages, power supplies, and drives as **this 32-bit** system from **Ziatech**.

were redesigned for PC-compatible hardware and software. S-100 designs, popular then in personal computing and used in some early embedded products, died quickly.

In the embedded world, the few who attempted to keep the older standards alive by making them PC compatible didn't have enough momentum to capture significant market share. VME, often based on Motorola CPUs, has persisted, but remains outside the PC-based sphere. Multibus fared somewhat better, but it too has become an insignificant player now. Of the older embedded-system standards, only STD bus adapted well and is still a viable contender.

Why? While much of the change is due to nontechnical marketing factors, form-factor plays a significant role. STD bus uses a relatively smaller board footprint (4.5" x 6.5") than all the others, as shown in Figure 1. It also is more robust in terms of mounting and cooling, and is supported on at least three sides—two sides by card guides and one by the bus connector itself. These features permit rugged and compact implementations, which are hallmarks of embedded systems. Because board size and card cages are compatible with the dimensions of 3.5" disk drives and switching power supplies, it makes for a neat total package.

The open-market philosophy of STD bus has also contributed. With numerous ex-



Photo 2: Conventional ISA/AT passive backplanes such as those from Microbus provide economical system solutions in a variety of sizes.

pansion boards, card cages, and enclosures to choose from, you can configure an embedded STD system to meet a variety of needs.

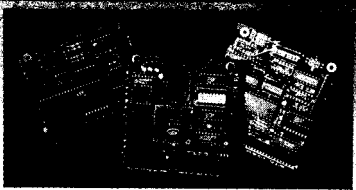
A typical STD bus package might look like that from Ziatech shown in Photo 1. Notice how neatly the drives and AC power supply fit within the card cage. By interleaving the required new signals between the traces for the old ones as is done with the EISA bus, this particular system uses the latest 32-bit version of STD bus.

This approach permits older 8- and 16-bit boards to be mixed with newer 32-bit units. As an example of the processing power available on a single STD board, Ziatech's ZT8905 offers a 133-MHz Pentium processor, onboard PCI video pathway, and up to 48 MB of DRAM.

contained only the computer and its support circuitry, many newer ones incorporate serial and parallel ports, floppy- and hard-drive interfaces, and video controllers. For some applications, this is all that's needed.

Another concern, though, is operating temperature. A conventional PC motherboard is intended for a rather benign environment. Often, embedded systems do not have the luxury of operating in a home or office.

As a case in point, an in-lobby ATM machine design I oversaw fit these constraints admirably. A minimum of computing power was required—one floppy drive, a receipt printer with standard LPT-port interface, a monochrome CRT monitor and a small keypad for user interaction, and a



### PC/104 Serial I/O

- RS-232 and RS-422/485
- New MIDI Version Available
- ASYNC or SYNC
- \* Single & Multi-Port Modules
- In Stock, Call
- Made in USA
- Free Technical Support from the Leader in PC Communications



**SEALEVEL™**  
COMMUNICATIONS & I/O

P.O. Box 830 • Liberty, SC 29657  
**(803) 843-4343**

#212

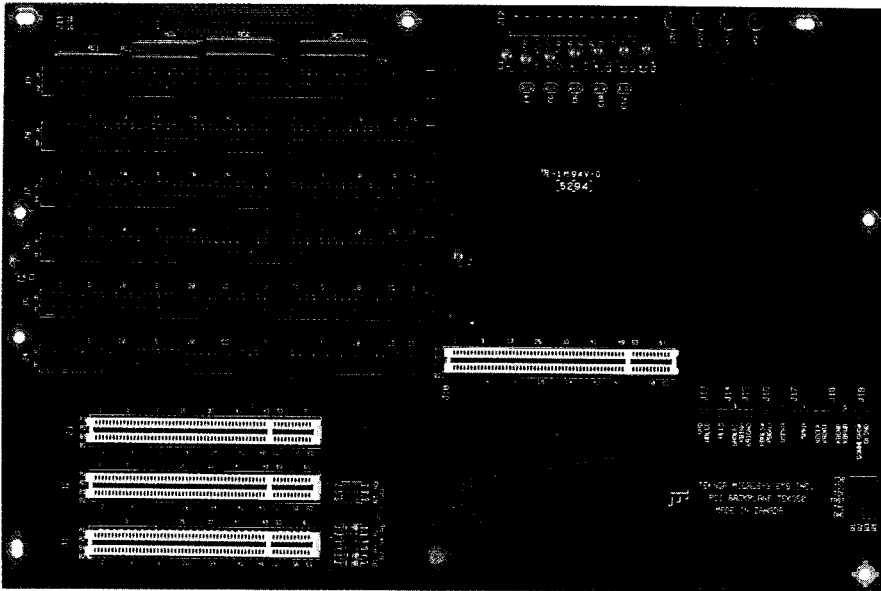


Photo 3: Teknor's **PCI-950** advanced passive backplane board with ISA and PCI sockets provides a powerful yet compact (13" x 8.7") and flexible approach to system construction.

## ISA, EISA, VLB, AND PCI

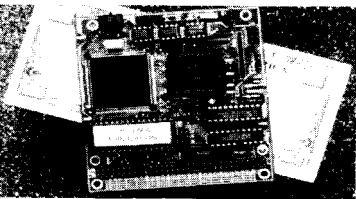
An even more direct approach to embedded-systems design is to simply embed a desktop PC motherboard in a target system. The main attraction to this approach stems from its very low cost and ready availability. Certainly, the result is PC-compatible, for it truly is a PC minus the case and desktop trappings. But, such an elementary approach is not without its shortcomings.

A PC-motherboard solution is not bad (provided the form-factor doesn't kill you) if everything you need to implement your embedded system can be found on the motherboard. While early motherboards

solenoid for accepting and locking the deposit envelope. A small custom board, which didn't even have to plug into the PC bus, supported these extra peripherals easily. The signals to and from this board connected to one of the motherboard's two serial ports serving as discrete I/O bits. Cost was of primary concern, and the stationary, vault-like enclosure in the bank presented a nearly ideal operating environment.

Space was of no concern—the cavernous ATM enclosure had plenty of room. A single 24-V power supply was needed for the receipt printer, so a simple, linear 5-V regulator on the custom interface board

### Power Miser Powerhouse!



- 5V at 100mA operating • SPI/2C Port (great for battery powered applications)
- PC/104 XT Engine
- Flash ROM Module
- Watchdog
- DOS in ROM
- 512K or 2MB DRAM
- E<sup>2</sup>PROM

**Only \$100/quantity 1000**

**vesta technology, inc.**  
303-422-8088 • FAX: 303-422-9800

#213

## AMX

The Real-Time Multitasking Kernel

680x0, 683xx 80x86/88 real mode  
80386 protected mode i960<sup>+</sup> family  
R3000, LR330x0 Z80, HD64180

**NEW** • DOS Compatible File System  
• TCP/IP • 29K<sup>+</sup> support

- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- InSight™ Debug Tool
- Comprehensive documentation
- No royalties, source code included

For a **free** Demo Disk and product description,  
Phone: (604) 734-2796  
Fax: (604) 734-8114



**KADAK Products Ltd.**  
206 1847 West Broadway  
Vancouver, BC, Canada V6J1Y5

#214

**Figure 1: STD bus uses a relatively small board footprint. Reliable mounting is achieved by supporting the board on three edges. The 8; 16; and 32-bit boards are all compatible due to interleaved edge fingers.**

provided the power for the rest of the system.

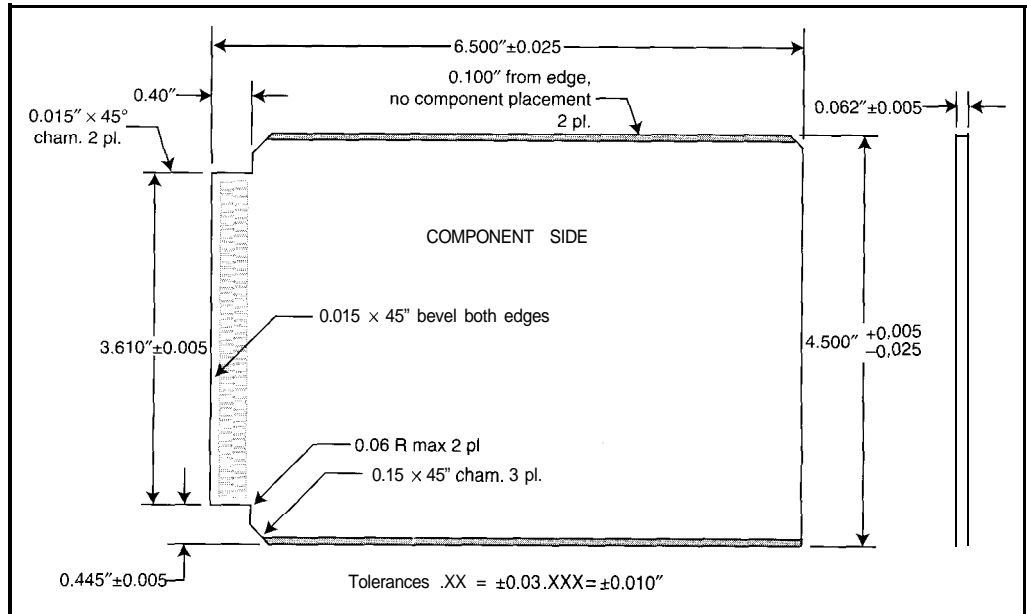
The design went together in no time flat (not counting the design and bending of sheetmetal for the cabinet), and the software effort was a lot like doing any other desktop PC application. The resulting design met all the objectives perfectly! However, had the system operated outdoors, needed many interfaces, or required compactness, life would not have been so simple.

PC motherboards are not designed for extremes of temperature or humidity and are notorious for poor mounting rigidity. If at all possible, it is best to avoid them entirely! When they are needed, you can improve mounting integrity as well as space requirements by using a right-angle riser board.

These nifty adapters keep system height to a minimum by mounting expansion boards in the same plane as the motherboard. They also permit improved support from the rear panel, with perhaps a bracket to support the edge opposite the bus.

But, watch out for variations due to PC and AT size standards and physical mounting tolerances. Cooling efficiency is also improved when all the boards are in the same plane. A single fan and judicious location of an intake filter provides a good, clean flow of air over all components.

With modern PC motherboards sporting VESA Local Bus (VLB) or PCI interfaces, a powerful and cost-effective solution can be achieved provided the



mechanical constraints can be tolerated. But, remember one other caveat: motherboard designs change rapidly! What is available from a given vendor today may not be available tomorrow. This shortcoming can wreak havoc with product longevity. Due to mechanical variations in size, mounting holes, connector location, and available peripheral controllers onboard, switching one board for another later on can be cumbersome and costly.

Nonstandard product is a particular concern with offshore offerings. Because you buy the system in the U.S. does not

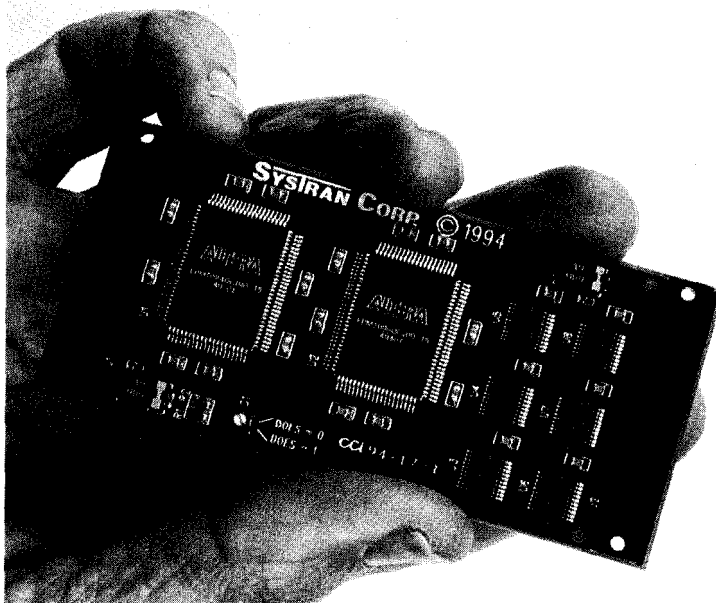
assure that is where it was built. Support in such cases is often minimal, irksome, or nonexistent. American Predator addressed these concerns head on. Their LPX and NSC line of '386 and '486 motherboards are made in the U.S. and are guaranteed to be available for at least two years. They are the only manufacturer I know of to make such a claim.

#### PASSIVE BACKPLANES

Many problems-mounting, ruggedness, reliability, packaging density, and cooling-associated with conventional motherboards may be overcome by eliminating the motherboard with its onboard expansion sockets. Just replace it with a passive backplane and collection of processor and interface boards.

This approach is the same as that used for STD-bus designs. But here, a conventional ISA, EISA, VLB, or PCI connector (or a combination) is employed. The key is that the backplane simply provides a means of interconnection. The processor resides in a plug-in board just like the interface boards. This approach offers many advantages.

To start with, one is no longer locked into a particu-



**Photo 4: Industry pack modules stack onto processor, controller, and system boards for compact expansion of I/O. These A/D and D/A converters are data-acquisition modules from Systran Corp.**

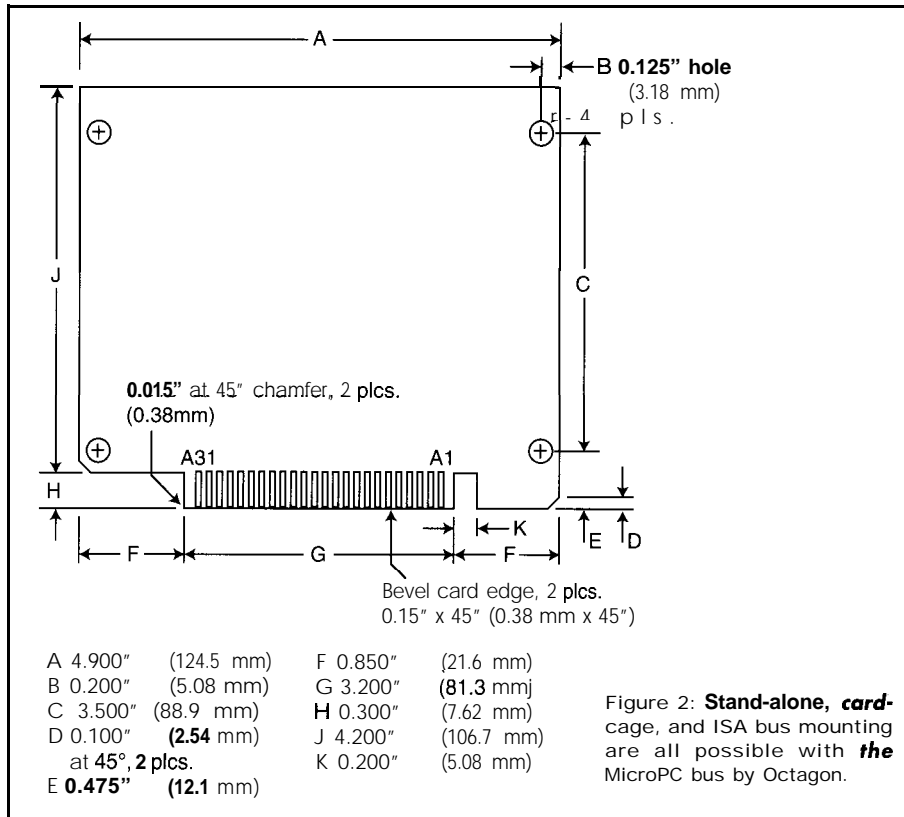


Figure 2: **Stand-alone, card-cage, and ISA bus mounting** are all possible with **the MicroPC bus** by Octagon.

lar motherboard or processor, so there is no danger of obsolescence. Should a more powerful (or less expensive) processor board become available, you can unplug one and exchange it for another. This exchange can even be done in the field if required.

With PC passive backplanes, though, it's important to design for sufficient card length right from the start. PC cards vary tremendously in length. Many passive backplanes **simply contain** ISA/PC or ISA/AT sockets, such as those shown in Photo 2 from Microbus. However, newer ones also support the emerging VESA, PCI, or VLB standards and connectors as well for enhanced system performance. The Teknor PCI-950 (shown in Photo 3) is one example of these.

Vendors, such as Octagon Systems, have defined their own board standard. This standardization ensures **that all** boards mechanically fit in the system. As shown in Figure 2, the MicroPC board has a compact (4.5" x 4.9") footprint. The original design uses a conventional 8-bit ISA/PC connector. Their boards are specified for the wide temperature operation (typically -40°C to +85°C) required in **many embedded** applications. These boards can mount in three different manners:

- plug into a conventional ISA passive backplane
- plug into a MicroPC rack (similar to an STD cage)
- operate stand-alone supported by their four convenient mounting holes.

Obviously, the first approach is the most flexible since other conventional ISA/PC-bus and **nonMicroPC** boards may be intermixed in the system. However, this solution often is not mechanically robust, and is limited to 8-bit ISA/PC bus cards. The second approach works well, provided that all the interface boards needed in the system are MicroPC compatible. Although similar to the approach of conventional PC motherboards, the third configuration is much more compact.

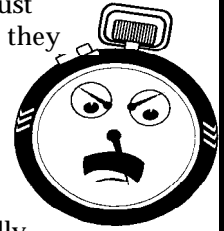
Octagon also specifies a more advanced **16-bit** connector **pinout** which supports all the ISA/AT signals. It uses a 72-pin **high-density** (interleaved) connector so that both **8- and 16-bit** boards may be mixed within the same system. I'm hopeful the MicroPC approach will catch on, for it offers many flexible packaging options.

### SHRINKING THE SIZE

While I will not say a lot about PC/I 04 boards (this technology is covered in

## Time is not on your side!

The customer just called to say they need the embedded controller prototype 2 weeks sooner.



There was hardly any time for development *before*. How can you possibly get all the Software

## But we are!

Our quick solution Single Board Computers will help you deliver fast. And we have the tools to support you - C compilers, debugger ROMs and Link-Locate Utilities. Custom work is our specialty. Check out some of our offerings on the Internet or call for brochures on these products:

## 80C188 and 8051 Products

**188 SBC** - use your Borland

or MS C/C++ compiler to develop and debug code.

A/D, D/A, Opto-rack I/F, LCD, Keyboard, PC/I 04, RTC and so much more.

**New! 188STD** - STD bus card with PCMCIA,

IEEE-488, 2 serial I/Os, more.

**552SBC** - an 8051 derivative with A/D, PWM, 40 I/O bits, 3 - 2321485, RTC, Watchdog.

**8031SBC** - we have a family of 8051 based single board computers, with serial ports, relays, opto-isolators, etc.

**We specialize in custom work too; as few as 25 units. CALL NOW!**

**HTE** HiTech Equipment Corp.  
9400 Activity Road  
San Diego, CA 92126  
[Fax: (619) 530-1458]

Since **1983**

**(619) 566-1892**



E-mail: [info@hte.com](mailto:info@hte.com) - Ftp: <ftp://www.hte.com>  
Web: <http://www.hte.com>

"PC/104 Quarter"), they should at least be mentioned for completeness. These miniature (3.6" x 3.8") boards, with the footprint shown in Figure 3, represent an excellent candidate for embedded PC systems. With the advent of the open-architecture PC/104 standard, a host of manufacturers now offer CPU boards as well as innumerable expansion boards and mounting accessories.

PC/104 uses one or two connectors for interconnection—64 pins for the basic 8-bit PC-bus equivalent and an additional 40-pin connector for the 16-bit AT-bus extensions. Signals are essentially identical to those found on a conventional PC, except for having lower drive capability and a unique interrupt-sharing provision.

These connectors are also stackable (pin and socket type), and permit mounting in either a stack or planar configuration. Either way, they offer:

- dense packaging
- sufficient rigidity for high-shock environments
- wide operating-temperature specs
- wide availability from over 100 manufacturers
- good cooling capability because all boards are in the same plane.

Due to their wide acceptance, PC/104 expansion sockets are often found on non-

PC/104 motherboards. This is a convenient means of expanding system capabilities either in the initial design stage or later on when needs change.

### ANOTHER OPTION

While not a processor bus standard, **IndustryPack** is another emerging bus standard intended for expansion modules only. These tiny (1.8" x 3.9") modules have connectors on each end which stack on top of a processor or interface board, all in the same plane. Their typical high-density SMT design, stackability, and location of the dual connectors makes for a compact, rugged, and convenient way to expand system capabilities.

Most **IndustryPack** modules are for A/D and D/A converters and specialized interfaces. Photo 4 shows some of Systran's data-acquisition **IndustryPack** modules.

### NONBUS BOARDS

Not all embedded-PC processor boards comply with a bus standard. Some stand-alone boards exist in whatever dimensions the manufacturer thought appropriate. These boards are most suitable when all the interface capabilities needed are contained on the one system (processor and interfaces) board.

Although these boards vary greatly in capabilities and size, they often contain one or two PC/104 or **IndustryPack** sockets (or both) to support features not built into

the basic system board. A spare PC/104 socket is also a great hedge against changing future needs.

Photo 5 shows just one of many **nonbus** system boards. This '486SLC embedded PC from Micro/Sys comes complete with **onboard** VGA graphics controller, two serial ports, parallel port, floppy- and (IDE) hard-disk controllers, and conventional PC keyboard port. Sockets support:

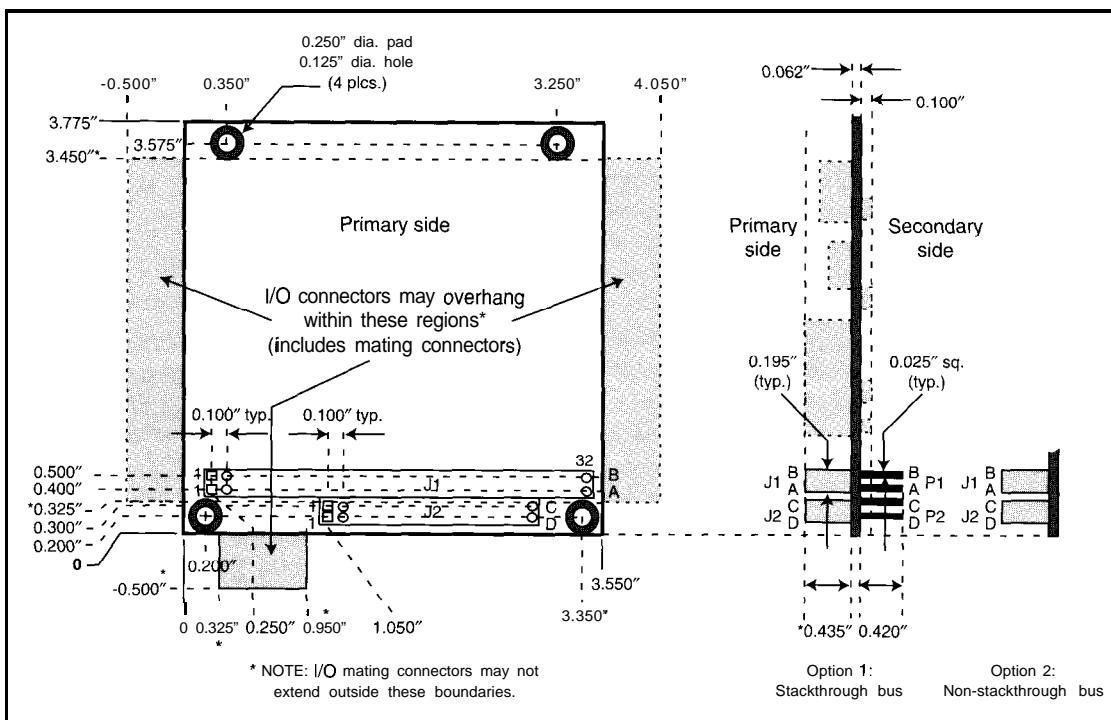
- up to 8 MB of RAM
- 1.8 MB of EPROM, flash (including on-board programming circuitry), or battery-backed SRAM
- an **80387SX** coprocessor
- one PC/104 socket

Conventional PC-compatible timers, DMA, interrupt support, and BIOS area enable the board to operate like a conventional desktop PC with all its software.

### FINDING THE FOREST

With so many options, just how do you select the right bus for your embedded system?

First of all, remember there's probably no right bus! Much depends on the space you have available for housing your complete system. Certain approaches can be ruled out on size alone. When footprint area and tiny total volume are the prime driving forces, a stack of PC/104 modules often proves to be the best approach.





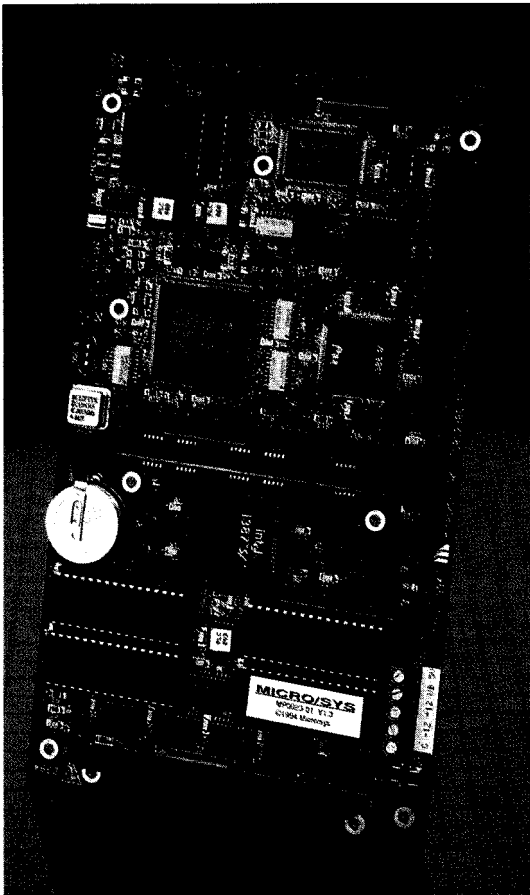


Photo 5: **Micro/Sys's nonbus** SBC2486 system board comes complete with RAM, EPROM, flash, peripheral controllers, and optional VGA and disk controllers. Note the PCI 04 expansion socket.

If you need the capability of expansion, rack mounting, disk drives, and power supplies in one enclosure, an STD card cage, MicroPC boards, or an ISA approach with passive backplane seems ideal. Don't forget that many processor boards offer expansion through piggybacking PC/104 or **IndustryPack** modules, and this hybrid approach might make great sense.

To meet overall system requirements, the next most critical factor is usually availability of suitable interface boards. It is best to make an exhaustive list of every interface needed, not only for the initial design, but also in planning for future expansion. A whole design can fall apart or get very messy if even one interface board is not available for the bus you have chosen. When the system requirements are well known, relatively static, and quite simple, **one of the nonbus, all-in-one boards is often an expedient solution.**

Once you have decided on a bus (or **nonbus**) standard, you must then select a processor. Unfortunately, old 8088, 8086, and even 80286 designs are nearly obso-

lete and offered by precious few vendors, though they often fill the bill nicely.

For true PC compatibility, the 80386SX is the low-end processor of choice. Intel's '386EX version is a particularly attractive chip if you are rolling your own or find it incorporated into an existing board. Available in 16-, 20-, and 25-MHz versions, it includes three serial ports, three timers, up to 64 MB of addressing space, two DMA channels, watchdog timer, 8259A interrupt controller, DRAM refresh logic, and eight chip-select lines in 5-, 3.3-, and 3-V configurations. For a **one-chip** PC, this is an ideal choice.

Equally available are boards using variations of the '486 CPU. These boards may be SX (no math coprocessor) or any of the host of DX versions in different speed ranges. Prices have plummeted on these, as the Pentium chip has rapidly gained prominence in the PC marketplace.

As yet, there are few Pentium embedded processor boards available (the Ziatech ZT8905 mentioned earlier is a notable exception), and I suspect their entry will be slow. Most embedded applications simply don't warrant the higher processing power, complexity, and cost associated with them. When supercomputing power is required, there are many RISC chips (such as the i960, ARM, and **PowerPC**) available to meet the need.

Regardless of which bus or processor chip you select, you can be assured of continued growth and vitality of product offerings, both in hardware and software. If you select carefully, design defensively, and keep an eye on your product's future needs and evolution, you'll reap the rewards of this powerful and flexible approach to embedded-system design.

## COMING ATTRACTIONS

Next column, we'll explore the myriad options available to you for packaging your embedded design. From open-frame mounting to card cages to fully packaged enclosures, both custom and off-the-shelf, you'll see there is always a suitable home for your embedded system. **APC.EPC**

Russ Reiss holds a Ph. D. in **EE/CS** and has been *active in electronics for over 25 years as industry consultant, designer, college professor, entrepreneur, and company president.* He may be reached at [russ.reiss@circellar.com](mailto:russ.reiss@circellar.com) or 70054.1663@compuserve.com.

## SOURCES

STD-Bus package-ZT8905  
Ziatech Corp.  
1050 Southwood Dr.  
San Luis Obispo, CA 93401  
(805) 541-0488  
Fax: (805) 541-5088  
BBS: (805) 54 1-82 18

LPX and NSC line of '386/'486 motherboards

American Predator Corp.  
c/o Global American, Inc.  
17 Hampshire Dr.  
Hudson, NH 03005  
(603) 886.3900  
Fax: (603) 886-4545

Passive Backplanes  
Microbus  
10849 Kinghurst, Ste. 105  
Houston, TX 77099  
(713) 568.4744  
Fax: (715) 568-4604

## MicroPC board

Octagon Systems  
65 10 West 9 1 st Ave.  
Westminster, CO 80030  
(303) 430-1 500  
Fox: (303) 426-8 126

## PCI-950

Teknor Microsystems, Inc.  
616 Cure Boivin  
Boisbriand, PQ  
Canada J7G 2A7  
(5 14) 437-5682  
Fox: (5 14) 437.8053

## IndustryPack

Systran Corp.  
4126 Linden Ave.  
Dayton, OH 45432.3068  
(513) 252.5601  
Fax: (5 13) 258.2729  
[info@systran.com](mailto:info@systran.com)

## '486SLC embedded PC Micro/Sys

3447 Ocean View Blvd.  
Glendale, CA 9 1208  
(8 18) 2444600  
Fax: (818) 244.4246

## PC/104 technology

Ampro Computers, Inc.  
990 Almanor Ave.  
Sunnyvale, CA 94086  
(408) 522-2 100  
Fax: (408) 522.3678

## IRS

419 Very Useful  
420 Moderately Useful  
421 Not Useful

## DEPARTMENTS

74 Firmware Furnace

84 From the Bench

92 Silicon Update

99 ConnectTime

## FIRMWARE FURNACE

Ed Nisley

# Journey to the Protected Land: Behind the Interrupt Curtain



Wrapping  
up this leg  
of the  
journey,

Ed checks out how  
interrupts are handled  
while in 32-bit Virtual-86  
mode. The necessary  
overhead may surprise  
you.



With only a few  
notable exceptions,  
all current CPUs

support interrupts in one  
form or another. CISC or not, there  
just aren't many alternatives that pro-  
vide rapid response to unpredictable  
events. Although we may quibble over  
just how rapid the response may be  
and whether the gain justifies the  
complexity, the machinery sits there  
waiting for us to get on with the job.

Interrupts require immediate at-  
tention, which is why designers build  
a hardwired reflex right into the CPU.  
Intel 80x86 CPUs running in Virtual-  
86 mode behave somewhat differently,  
balancing the need for speed against  
the strictures of protected mode. The  
fact that Pentium chips support hot-  
wired V86 interrupts tells you what's  
valued more in today's market!

Last month, we looked at V86  
interrupts from the 16-bit side. Now,  
we can pull back the curtain and ex-  
amine the 32-bit machinery that  
makes it all possible. Even if you're a  
diehard real-mode fan, you'll learn a  
few things about how interrupts work  
and why protected mode is so pro-  
tected.

### SWAPPING AND STUFFING STACKS

There are two parts to the V86  
monitor code behind each interrupt.

Listing 1—When an IRQ 7 occurs in V86 mode, the CPU vectors through a 32-bit interrupt gate to the stub routine, then to the main handler. The code modifies the contents of both stacks to simulate an interrupt aimed at the 16-bit code. The register structure appears in Listing 3.

```

LABEL      V86IRQ07Stub
PUSHA                                ; save bystanders
MOV        AL, 7
JMP        V86IRQxxHandler

<<< Stubs for IRQ 0-6 and 8-15 omitted >>>

PROC       V86IRQxxHandler

MOV        EBP, ESP                    ; aim at stack structure
MOV        EBX, GDT_DATA                ; aim seg regs at kernel
MOV        DS, EBX
MOV        EBX, GDT_CONST                ; and kernel constants
MOV        FS, EBX

CallSys    CGT_V86_REMOVEGATES          ; remove our V86 handlers

redirect the interrupt to the V86 task

MOVZX     EAX, AL                        ; convert index to dword
LEA       ESI, [HWIntGates + EAX*SIZE HWGATEMAP]
MOVZX     EAX, [(HWGATEMAP PTR FS:ESI).V86IntNum]

SHL       EAX, 2                          ; get V86 vector offset
CallSys    CGT_MEM_PEEKREAL, 0, EAX; get V86 vector in EAX

SUB       [INT_PTR.01dESP], 2              ; push interrupted state
CallSys    CGT_MEM_POKEREAL, \
           [INT_PTR.01dSS], [INT_PTR.01dESP], \
           [INT_PTR.01dEFLAGS], 2

SUB       [INT_PTR.01dESP], 2
CallSys    CGT_MEM_POKEREAL, \
           [INT_PTR.01dSS], [INT_PTR.01dESP], \
           [INT_PTR.01dCS], 2

SUB       [INT_PTR.01dESP], 2
CallSys    CGT_MEM_POKEREAL, \
           [INT_PTR.01dSS], [INT_PTR.01dESP], \
           [INT_PTR.01dEIP], 2

MOVZX     EDX, AX                        ; aim ret addr at
SHR       EAX, 16                          ; V86 handler
MOV       [INT_PTR.01dCS], EAX
MOV       [INT_PTR.01dEIP], EDX
AND       [INT_PTR.01dEFLAGS], NOT MASK EF_IF

reinstall our V86 handlers and return to the V86-mode handler

CallSys    CGT_V86_INSTALLGATES

POPA                                ; restore bystanders
IRET                                       ; execute the handler

ENDP       V86IRQxxHandler

```

The first part, shown in Listing 1, gets control when the CPU responds to the interrupt signal and finds itself in V86 mode. The second part, shown in Listing 2, starts when the concluding **I RET**

instruction in the 16-bit handler triggers a GPF. We'll dissect each chunk in turn.

The pure 32-bit PM handlers we used in *INK 57, 58, and 59* don't suffer

from this division. Only when you must activate a 16-bit interrupt handler while the CPU is in V86 mode does this trickery come into effect. Unfortunately, that situation isn't nearly as rare as you'd hope. Every protected-mode OS runs into precisely this situation when the subject of DOS programs comes up!

Last month, you saw how FFTS matches the printer port's IRQ 7 signal with the PM interrupt gate at Int 57. Each of the hardware interrupts activates a stub routine similar to the first few lines in Listing 1 that save the CPU registers and load AL with the IRQ number. Remember that an Intel 80x86 handler has no way to identify the interrupt that invoked it, which means that number must appear somewhere in the source code.

At the start of the V86 IRQxx Handler routine, SS:ESP points to the structure shown in Listing 3. The CPU automatically stacks the 16-bit V86 segment registers (padded with two high-order bytes), ESP, all 32 bits of EFLAGS, and EIP before entering Listing 1 in 32-bit protected mode. Except for CS and SS, the segment registers contain binary zeros to prevent protection exceptions in the interrupt handler.

The first few lines copy ESP into EBX to get easy access to the stacked values, then aim two segment registers at the FFTS kernel's variables and constants. A more complex handler would certainly use local variables on the stack and require more setup, but this suffices our purposes.

I remove the V86 GPF handlers before starting the stack manipulations and reinstall them just before returning to the V86 code. This step eliminates the problem of PM code bugs invoking the V86 error handler, a situation fraught with peril. You can combine both PM and V86 error functions into a single routine and eliminate this hassle. I wrote two separate handlers, so you can discard the V86 one if you're running pure PM code in your box.

The contents of AL, indexed into the table in Listing 3, *INK 64*, extracts the V86-mode interrupt number. Knowing that number, we can locate

the V86 task's interrupt vector and extract the real-mode handler's address.

Essay question: what's the C syntax for the five lines starting with **MOVZ X** in Listing 1? Extra credit: if you do it in one line, can anyone else decipher it? Bonus points: can you!

The left-hand side of Figure 2 in **INK 64's** column showed the contents of the Ring-0 and Ring-3 stacks just after the IRQ 7. Before the V86 monitor activates the 16-bit handler, it must transfer the address of the interrupted instruction to the Ring-3 stack and put the handler's address in the Ring-0 stack. The right-hand side of that figure shows the desired result.

The monitor code can only access the Ring-3 stack as data, which means it cannot use **PUSH** and **POP**. The next few lines in Listing 1 fetch values from the Ring-0 stack and push them on the Ring-3 stack using the `MemPokeReal` routine. I'll admit that TASM's extended **CALL** instruction syntax makes this slightly impenetrable. You should examine the assembler's output listing to see the tonnage of code created by each of these "instructions."

I've fought through webs of obscure constants and magic numbers in similar code from other operating systems. If speed isn't everything, try the method I used here, even if it takes more effort to set up the stack structures and figure out the **CALL** parameter notation. Once you see how it works, you can then tweak it for higher performance.

After preparing the Ring-3 stack, the monitor stuffs the 16-bit interrupt handler's address into the CS:IP values on the Ring-0 stack. It changes only the low word of EIP, secure in the knowledge that the high word must be zero. This assumption is not valid for arbitrary V86-mode, but it suffices for now.

The last few instructions reinstall the V86 GPF handler gate, restore the V86-mode registers saved by the entry stub, and execute an **I RET**. Even if entering an interrupt handler with an **I RET** seems peculiar, that's the way it works in protected mode. Remember that **I RET** can perform a task switch, flip through an interrupt gate, or re-

**Listing 2**—When the 16-bit interrupt handler attempts to execute an **IRET**, the CPU generates a GPF. This chunk of the V86 monitor verifies that the instruction was an **IRET** and then rearranges both stacks to allow a return to the 16-bit instruction interrupted by the external signal. The stack structure is similar to Listing 2 with an error code between **EAX** and **EIP**.

```

CMP    AL,0CFh           ; CF = IRET
JNE    @@NotIRET

CallSys CGT_MEM_PEEKREAL,[EC_PTR.01dSS],[EC_PTR.01dESP]
MOVZX  EAX,AX
MOV    [EC_PTR.01dEIP],EAX
ADD    [EC_PTR.01dESP],2

CallSys CGT_MEM_PEEKREAL,[EC_PTR.01dSS],[EC_PTR.01dESP]
MOVZX  EAX,AX
MOV    [EC_PTR.01dCS],EAX
ADD    [EC_PTR.01dESP],2

CallSys CGT_MEM_PEEKREAL,[EC_PTR.01dSS],[EC_PTR.01dESP]
MOV    [WORD PTR EC_PTR.01dEFLAGS],AX ; low word only!
ADD    [EC_PTR.01dESP],2

CallSys CGT_V86_INSTALLGATES ; restore our handlers

CLI                                ; turn interrupts off again
POPA                                ; restore bystanders
ADD    ESP,4                       ; step over Ring-0 error code

IRET                                ; return to V86 code

```

turn from an interrupt, depending on the circumstances. Talk about operator overloading!

The CPU is still in protected mode when it recovers the CS:EIP

values and EFLAGS from the Ring-0 stack. The VM bit in EFLAGS is set, telling the CPU to return from 32-bit PM to V86 mode. It restores the segment registers to their address-bit

**Listing 3**—This structure shows the Ring-0 stack layout used by the V86 monitor's hardware interrupt handler. The handler's **PUSHA** saves the CPU registers starting with **EAX** and ending with **EDI**. The CPU stores the other registers automatically while passing through the interrupt gate.

```

STRUC INT_STACK
01dEDI DD ? ; last of PUSHAs regs
01dESI DD ?
01dEBP DD ?
01dESP2 DD ? ; points to 01dEIP
01dEBX DD ?
01dEDX DD ?
01dECX DD ?
01dEAX DD ? ; first of PUSHAs regs
01dEIP DD ?
01dCS DD ?
01dEFLAGS DD ? ; should have VM & RF set
01dESP DD ?
01dSS DD ?
01dES DD ?
01dOS DD ?
01dFS DD ?
01dGS DD ?
ENDS INT_STACK

INT_PTR EQU <<(INT_STACK PTR EBP)>>

```

values, loads SS:ESP from the stack, and enters V86 mode again. It fetches the first instruction of the 16-bit interrupt handler and, as far as that code can tell, the IRQ 7 triggered Int OF just as in real mode.

We, of course, know better.

## UNWINDING THE STACKS

The V86-mode interrupt handler appeared in Listing 2 of *INK* 64. It does everything you'd expect a real-mode interrupt handler to do. When it's done, it attempts to execute an IRET instruction.

In real mode, that instruction simply pops CS:IP (not EIP) from the stack and returns control to the interrupted instruction. In V86 mode, however, IRET is a privileged instruction that causes an immediate GPF. The CPU bails out of V86 mode and, once again, enters the 32-bit PM V86 monitor.

In *INK* 63, you saw how the GPF handler got control after an Int 20. The process is identical for an IRET, except that the monitor must now decipher two possible causes for the GPF. The code in Listing 3, *INK* 63, shows the mechanics of retrieving the op-code. This month's Listing 4 shows the test for an IRET.

Figure 1 shows the stack layout just after the CPU encounters the IRET. Once again, the Ring-0 stack holds the V86-mode segment registers, flags, and so forth. The CPU also pushes an error code after the registers. Even though the error code is always zero, the GPF handler must discard it before attempting to return through the stack.

The contents of the Ring-3 stack should look familiar: it's the same three registers we placed there just after the hardware interrupt. As before,

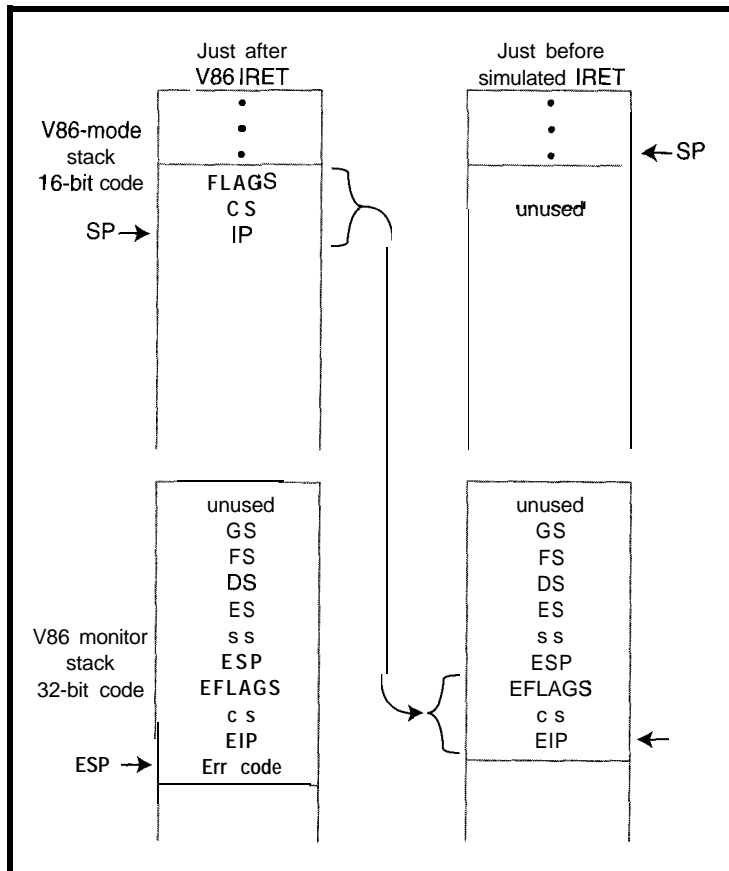


Figure 1 V86-mode IRET, the CPU automatically switches stacks and invokes the V86 monitor program. The monitor copies the address from the Ring-3 stack into the Ring-0 stack and returns to the interrupted instruction just as the CPU would in real mode. Unlike real mode, however, a bogus address or invalid stack quickly leads to a protection exception.

the monitor simulates the CPU's real-mode actions by transferring CS:IP and FLAGS to the Ring-0 stack and deleting them from the Ring-3 stack. The right side of Figure 1 shows the two stacks just before the CPU executes the final IRET in the GPF handler.

At first glance, you might think we could simply leave the 16-bit return address on the Ring-0 stack, while the V86 interrupt handler executes, and skip all the stack shuffling. Unfortunately, consider what happens as the CPU switches from the V86-mode stack to the Ring-0 stack. It simply reads the SS:ESP fields from the task's TSS, loads them into the corresponding CPU registers, and begins pushing segment registers.

Thus, any values left on the Ring-0 stack while the V86 code executes get clobbered the next time the CPU switches stacks. In effect, even if you don't clean up the stack before returning, the CPU does it for you

before entering your code.

In any event, the GPF handler and stack shuffling impose about 20  $\mu$ s of delay from IRET at the end of the 16-bit handler to the beginning of the interrupted 16-bit instruction. Even though that's significantly less than the 50- $\mu$ s latency on the front end, each hardware interrupt in V86 mode drags about 70  $\mu$ s of overhead with it. In our case, a trivial 7- $\mu$ s handler actually takes ten times that long from start to finish.

Now you know why DOS communications programs sometimes lose characters when they're running in Virtual-86 DOS boxes. It's not their fault, they're pedaling as fast as they can!

## NITS AND GRITS

With the details of V86 interrupts well in hand, let's look at the

larger implications.

Interrupts remain disabled from the time the CPU begins executing the 32-bit handler until it returns to the original V86-mode instruction. While Photo 1 shows a 50- $\mu$ s latency, you must also realize that no other interrupts can occur while the V86 monitor is in control. Obviously, you may enable interrupts at any point, but you must decide how to handle nested interrupts, multiple V86 interrupts, and so forth.

All of the code you've seen so far assumes that the V86 interrupt occurs while the V86 task is executing. What happens if a PM task is running when an interrupt intended for the V86 task arrives? Think about it before you answer!

It turns out that Bad Things Happen To Good Code. What you want to happen goes like this: the V86 monitor should detect that the interrupt occurred with a 32-bit PM task active,

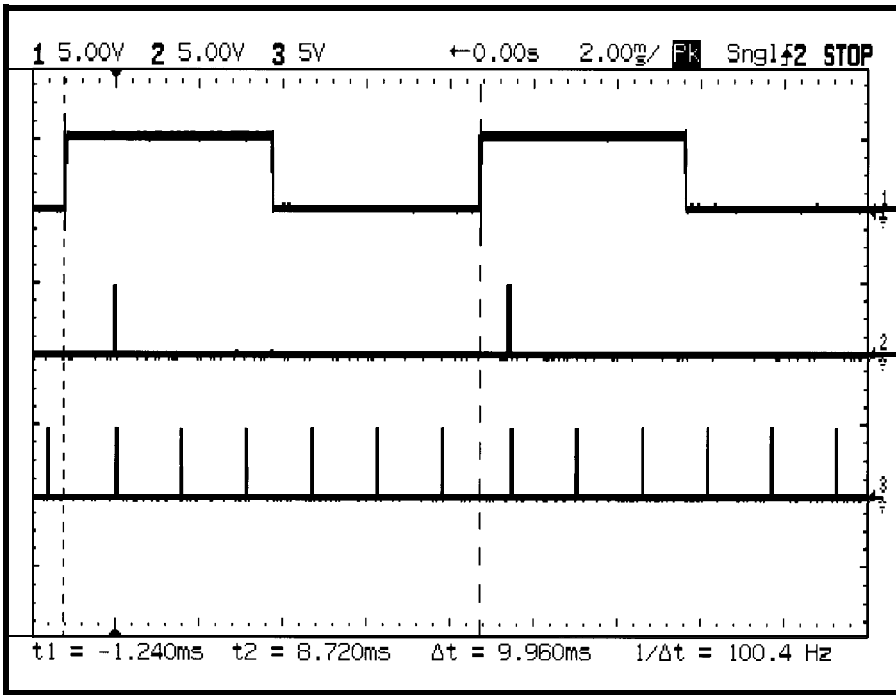


Photo 1—The V86 task has no trouble keeping up with relatively fast interrupts! Each rising edge triggers an interrupt pulse shown in Trace 2. The V86 task produces the blips in Trace 3, and the interrupt handler can run only when the task is active.

invoke the task dispatcher, switch to the V86 task, simulate a V86-mode interrupt, execute the 16-bit handler, then unwind things back to the original PM task.

What actually happens is that the monitor detects a V86 interrupt in protected mode, displays an error message, and locks up the machine. There is a simple motivation: FFTS depends

on cooperative multitasking. We simply don't have the code that fires up a V86 task on the fly for reasons you can easily imagine after you begin sketching out what must be accomplished.

Cooperative multitasking imposes what seems to be a severe limitation on the V86 task. It must enable any interrupts it expects to use when it begins executing and disable them before it returns control to the FFTS task dispatcher. In effect, V86-mode external interrupts can be active only when their task is running.

Listing 4 shows how it's done with IRQ 7 from the parallel port—you can easily extend the idea to other sources. The key point is that you must disable the interrupt either on

**STOP  
LOOK  
LISTEN**

Odds are that some time during the day you will stop for a traffic signal, look at a message display or listen to a recorded announcement controlled by a Micromint RTC180. We've shipped thousands of RTC180s to OEMs. Check out why they chose the RTC180 by calling us for a data sheet and price list now.



**CALL 1-800-635-3355**



**MICROMINT, INC.**

4 Park Street, Vernon, CT 06066  
(203) 871-6170 • Fax (203) 872-2204

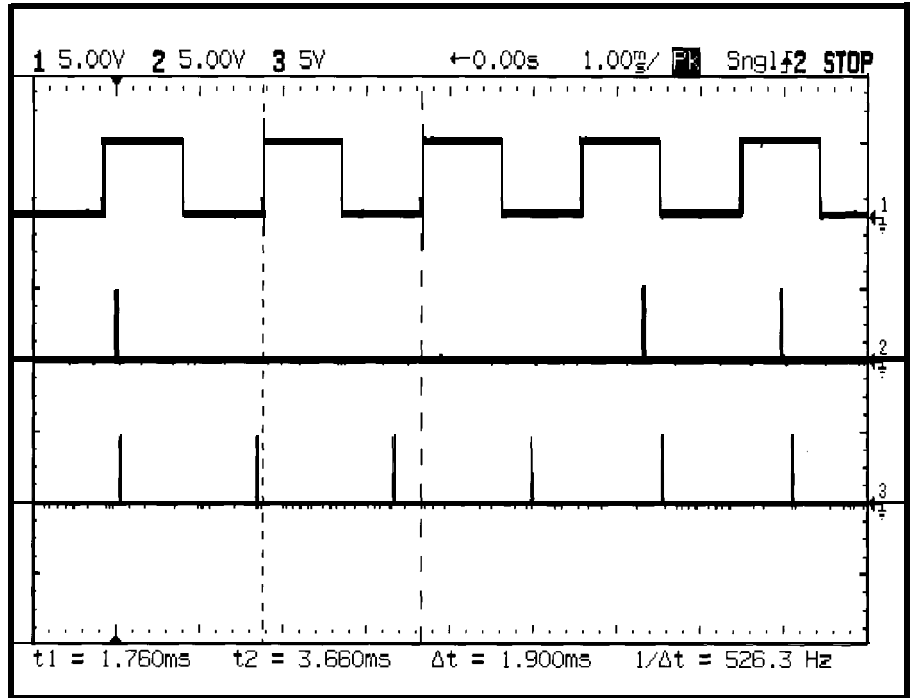
in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194 • Distributor Inquiries Welcome

Photo 2—The V86 task misses interrupts that occur when it's not running. The second and third pulses in Trace 1 occur entirely between the V86 task activations shown in Trace 3 and, thus, do not trigger interrupts. The 8259 interrupt controller does not remember interrupt inputs that become inactive before the CPU acknowledges them.

the card or at the 8259 interrupt controller before returning to the FFTS kernel. There is an obvious security hole in any system that, like FFTS, allows V86 tasks unlimited access to the 8259!

You can use the I/O Permission Bitmap in the task's TSS to restrict access to key I/O ports. The GPF handler can detect a read or write of the 8259's Interrupt Mask Register port, then verify that only the proper bits are modified. As always, there is an obvious tradeoff between speed and security.

Photo 1 clearly shows the delay between the rising edge of the interrupt source and the 16-bit interrupt handler. The maximum interrupt la-



tency is equal to the longest time between V86 task executions, which depends on what the other tasks are doing throughout our round-robin dispatching loop.

In this case, the kernel, three PM taskettes, and the V86 task perform about 3200 task switches per second. The V86 task gains control roughly 650 times per second, leading to a

**NEW! NEW! NEW! NEW! NEW! NEW!**

# LynX-10™

## Serial RS-232 to X10 (TW 523) Co-processor

- Fully Bi-Directional
- Reliable X10 Communication
- No Polling Required
- "Polite" Access to Powerline
- Collision Detection/Auto Retransmission
- Enhanced X10 Command Functionality
- Smooth Dim and Bright Commands
- 50Hz/60Hz Autoswitching
- Many More Features

Available as Chip; Developer's Kit; Fully Assembled Board; Boxed, Plug-in Package

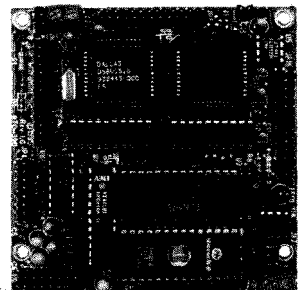


P.O. Box 950940, Lake Mary, FL 32795  
407-323-4467 Fax 407-324-1291  
BBS 407-322-1429

# BLAZING SPEED RTC320 AND SMART TOO!

One of Micromint's hottest-selling products for the past five years has been the RTC31/52 stackable controller. It has been a leading price/performance choice among our customers. With our new RTC320 board, we have expanded the value of that relationship even more.

Occupying the same small 3.5"x3.5" RTC footprint and using S-V-only power, the RTC320 uses the new Dallas Semiconductor 80C320, which is 8031 code compatible and 3-S times faster. At 33 MHz, the RTC320 is an S-MIPS controller! Along with the new powerful processor, the RTC320 board accommodates up to 192 KB of memory, two serial ports (RS-232 and RS-485), 24 bits of TTL parallel I/O, and a 2-channel, 12-bit ADC. The RTC320 puts some real firepower under the abundant variety of RTC I/O expansion boards. Plugging in your favorite ICE or EPROM emulator is the easiest way to develop code. For the diehards who like to twiddle the bits directly, we have a ROM monitor specifically designed for the Dallas '320.



**RTC320-1 (22 MHz)**  
**\$239 \$179/Qty.100**

(Call for pricing on 33 MHz)



4 PARK STREET • VERNON, CT 06066 • (860) 871-6170 • FAX (860) 872-2204

maximum latency of about 1.5 ms. Yes, that's milliseconds, not microseconds. The IRQ signal must remain high until the CPU acknowledges it and, thus, a square-wave input cannot exceed half that frequency: about 325 Hz.

When interrupts arrive faster than that, the V86 task simply can't keep up. The 525-Hz signal in Photo 2 results in a few missing interrupts as the input rises and falls while the task is inactive. Raising the duty cycle helps this situation, with the upper limit being a low-going blip. With 99.9% duty cycle, you can run at just under the maximum task activation rate.

However, the FFTS taskettes present an ideal situation: they are all trivial and well-behaved. In actual practice, a cooperative multitasking system depends on each task to limit its own execution time. Suppose your system has one task that can run for,

say, 100 ms once in a while. That single task limits the interrupt rate to a mere 10 Hz, even if the average rate could be 500 Hz.

Lest you think these problems are unique to protected mode, they're not. You'll find the same situations cropping up in real-mode programming, albeit with different timings. You can try to hide, but the system still won't run!

On the brighter side, you can hot-wire critical PM interrupt directly to the corresponding V86 interrupt, eliminating all of the table lookups and stack shuffling. I suspect you can get the overhead down to a few tens of microseconds with a lot of effort. Por-ing over the OS/2 and Windows DOS-box routines would be interesting, wouldn't it!

Although we won't get into it here, the problems become more complex with multiple V86 boxes. The key

CPL	Current Privilege Level
DPL	Descriptor Privilege Level
EOI	End Of Interrupt (command)
FDB	Firmware Development Board
FFTS	Firmware Furnace Task Switcher
GDT	Global Descriptor Table
GDTR	GDT Register
GPF	General Protection Fault
IBF	Input Buffer Full
IDT	Interrupt Descriptor Table
IDTR	Interrupt Descriptor Table Register
IF	Interrupt Flag
IOPL	I/O Privilege Level
LDT	Local Descriptor Table
LDTR	LDT Register
NT	Nested Task
OBF	Output Buffer Full
P bit	Present bit (in a PM descriptor)
RF	Resume Flag
RPL	Requestor Privilege Level
TF	Trap Flag
TR	Task Register
TSS	Task State Segment
VM	Virtual Machine (in EFLAGS)

**Listing 4—The V86 task is an endless loop punctuated by hardware interrupts and task switches to the 32-bit tasks. Printer port interrupts must not occur when this task is not executing because the FFTS kernel does not support preemptive multitasking.**

```

@@Again:
MOV  DX,SYNC_ADDR      ; set up for scope blips
IN   AL,DX              ; set trace blip
OR   AL,20h
OUT  DX,AL

MOV  [ES:DI],CH        ; pop char into video buffer
INC  CX                ; and tick the counter

MOV  AX,[IntCounter]   ; show normal interrupts
MOV  [ES:DI+2],AL

MOV  AX,[UnExIntCtr]   ; show unexpected interrupts
MOV  [ES:DI+4],AL

MOV  DX,SYNC_ADDR      ; set up for scope blips
IN   AL,DX              ; clear trace blip
AND  AL,NOT 20h
OUT  DX,AL

IN   AL,I8259A+1       ; disable IRQ
OR   AL,INTMASK        ; 1 = mask interrupt
OUT  I8259A+1,AL

INT  20h                ; crash into V86 monitor...

IN   AL,I8259A+1       ; enable IRQ
AND  AL,NOT INTMASK    ; 0 = enable interrupt
OUT  I8259A+1,AL       ; shazam!

JMP  @Again            ; repeat forever

```

issue is deciding how to handle multiple requests for the same interrupt.

For example, suppose two different V86 boxes attempt to enable the same interrupt. What should the V86 monitor do? Or should that situation be defined out of existence when the scheduler creates the tasks?

Ah, engineering tradeoffs... .

## RELEASE NOTES

Take a look at last month's BBS code in light of what we've seen now. It ought to make a bit more sense, particularly when you hitch up a signal generator and start poking around inside the handlers. Give it a try!

Next month, we'll return from the Protected Land to check on some interesting projects. Fear not, though, as this series continues in a few months with a V86 BIOS Box. □

*Ed Nisley (KE4ZNU), as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at ed.nisley@circellar.com or 74065.1363@compuserve.com.*

## IRS

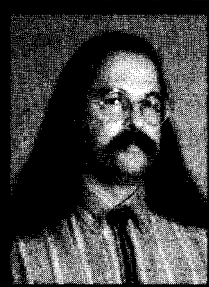
422 Very Useful  
423 Moderately Useful  
424 Not Useful



# FROM THE BENCH

Jeff Bachiochi

## Carrier Current Modem Part 2: Alternative Control



This  
month  
Jeff  
shows us

how to shrink a computer into one of the power-line modem interfaces to control appliances. With this method, you gain X-10 power-line advantages and closed-loop confidence.



here's a crispness in the air. I need gloves for my morning commute. With

no fairing on my motorcycle, my fingers are the first to be nipped by autumn's icy breeze.

Green and yellow leaves are falling from lack of rain. The yellows fill in the median's stripes, cautioning drivers against crossing over the slippery boundary.

Although the fall colors are disheartening, the smell of fresh grapes and other fall fragrances compensates. These sensations must be captured quickly during my commute's few allotted minutes of freedom. If well seized, they can be savored all day, counteracting normal daily stress.

They're there, they're free, take notice, and use them.

Last month, I showed how PCs can be tied together, networked if you will, using an ST7537 power-line modem. No biggie, you say, networks have been around since the sharing of resources was found to be profitable.

And you're right. However, networking without having to run special cable, be it coax, twisted pair, or fiber optic, is elusive. New homes often come with most rooms prewired for telephone and AC power. But, internal or external modems rarely can use the

phone lines for intrahome communications.

That's where the AC power lines come in. They're there, they're free, take notice, and use them.

While hazardous potentials exist at every outlet, AC power is our way of life. For X-10, it's big business—a business that has run open loop long enough. You know open loop. It's like a telephone conversation with an answering machine. You never know if your message gets through.

Perhaps, while sitting in your living room reading the paper (or your favorite magazine), it begins to get dark. You may ask someone walking by the wall switch to turn on the overhead light. Or, you may tap the X-10 transmitter located on the end table next to your easy chair.

Conceivably, your home-control system would have analyzed the light level and automatically sent an X-10 command. Without realizing it, you close the loop as you expect the light to turn on. If the command gets lost, you make the appropriate adjustment to turn it on again. Without this feedback, you can't be assured the command has been received.

Perhaps you only have a single PC in your house. You don't need a network. But, hold that thought.. .

By shrinking a computer down into one of the power-line modem interfaces, you can control appliances, just like X-10, but with closed-loop confidence. This means you not only acknowledge commands, but return status information to the transmitter.

### START SIMPLE

Refer to Figure 2 of *INK 64*. If we replace the MAX232 with a small processor, we have the components necessary for local control. X-10 offers triac (solid-state) control of up to 300 W and mechanical-relay control of up to 500 W. Either of these items can be added [see Figure 1 and Photo 1).

Different outputs are provided—one for the on/off signals used with either a triac or a mechanical relay and the other for PWM used with a triac to dim lights. The PWM signal must be in sync with the 60-Hz line frequency to retain a constant output level.

Since uncovering this 1200-bps power-line modem, I've discovered SGS has replaced the ST7537 with a 2400-bps version. The newer chip is a direct replacement with twice the throughput. Things just keep getting better.

The protocol I chose for the '7537 is a simple one: just seven bytes. The transmission packet contains a sync, preamble, to-address, from-address, function, value, and checksum.

Prior to transmitting any packet on the line, carrier detect is checked to certify the medium is free from other packet traffic. The originator starts by sending the sync byte, which initiates the transmitter's carrier. By the time the carrier is detected by all the listeners on the line, it is well into the byte.

Since the data is all 1s, each of the micros listening to the line can easily find the stop bit and prepare to receive the preamble and subsequent bytes. If

the preamble isn't recognized correctly, the packet is rejected. If the preamble is correct, the micro looks for its own address. If the to-address byte doesn't match, the whole transmission is likewise rejected.

The from-address byte tells the micro where the packet originated. This information determines the legality of the following function and value byte. Not all functions require a value. However, to keep the packet length consistent, a dummy value must be used.

The packet ends with a checksum byte. The sum of all six data bytes should equal zero or it can be assumed to be corrupt.

### SIMPLE FUNCTIONS

To keep things simple, let's use three basic functions. Table 1 summarizes the functions.

Mechanical relay control is designated by the name AM (appliance

module). It has two values: 0 (off) or any nonzero number 1-255 (on).

Triac control is designated as LM (lamp module) and is similar to AM, except its value can be anywhere from 0% (full off) to 100% (full on). For a lamp that is on, this value controls the delay between each power-line zero crossing and when the triac is turned on.

To close the loop and give the originator confidence the task has been accomplished, the target module responds to each command with an acknowledge packet. This packet swaps to and from addresses and adds 128 to the function byte, providing the originator with a duplicate copy (slightly rearranged) of its original packet (see Table 2).

### MECHANICAL OR SOLID-STATE RELAYS

Solid-state relays are becoming a popular replacement for mechanical

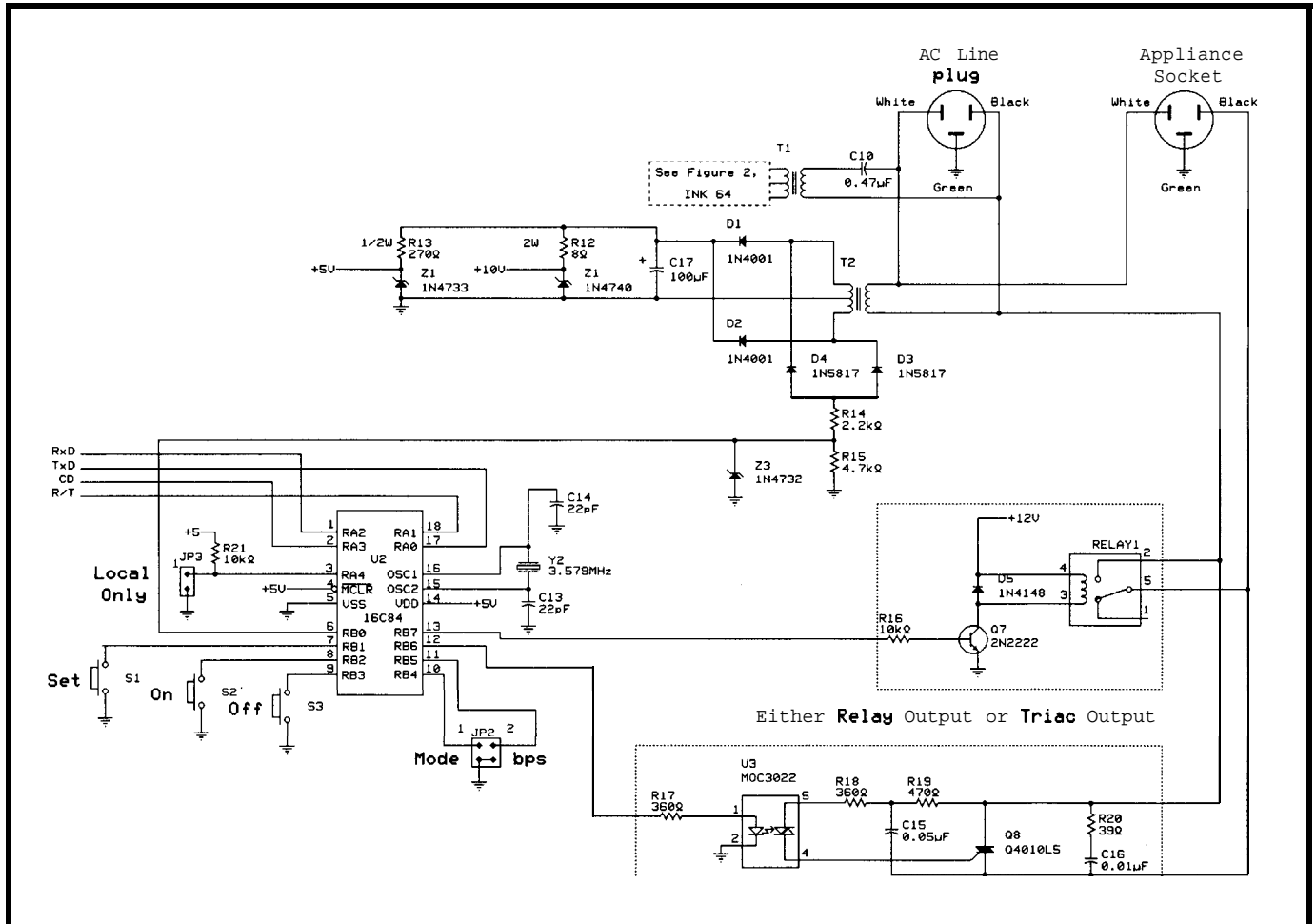


Figure 1—Based on last month's schematic, the RS-232 interface is traded in for either relay or SSR power-control circuitry

relays in many applications. Sized correctly, a solid-state relay operates indefinitely. Mechanical relays not only have mechanical cycle limits, but also contact degradation limits.

Since solid-state relays can directly replace mechanical relays, you may choose to use either one in conjunction with the micro's relay-output circuitry.

The only thing you have to bear in mind is that most solid-state relays come with built-in zero-crossing circuitry, which makes them unusable as dimming controls. You must use special units called *random turn-on modules* instead.

The drive circuitry for both mechanical and solid-state relays consists of a drive transistor which can sink either kind of relay to ground. Commands to control the AM are simply On or Off. Off uses a value of zero, while On may use any nonzero value.

## TRIAC CONTROL

A triac can be used as a solid-state relay. Once turned on through a gate input, it remains on until the line voltage returns to zero (at each zero crossing). Most solid-state switches incorporate a zero-crossing detector which causes the device to turn on only at zero crossings (while no current is flowing).

Although this is great for switching things on and off, it doesn't allow for intermediate settings. When selecting a dimming device for this project, use only random turn-on devices. LM commands use a value of 0 to 100. This value pertains to light level—0% represents full off while 100% is full on.

At the 60-Hz line frequency, each half cycle takes just 8.33 ms. If each half cycle is divided into 100 equal parts, each part would be 83  $\mu$ s in length. If we wait 50 x 83  $\mu$ s or 4.15 ms after each zero crossing before turning on the triac, it would be on for the second half of each half cycle.

Unfortunately, this step does not result in half the light output. Because the

Name	Function byte	Value byte
Unit On	0	NOT 0
Unit Off	0	0
Lamp On	1	1-100%
Lamp Off	1	0
Query	64 + function	value
Ack	128 + function	value

Table 1—In addition to simple on and off commands, the module also supports querying and acknowledgment.

percentage of light output does not correspond linearly to a percentage of a cycle's on time, we have to fudge the timing.

A 100-entry lookup table passes the appropriate delay-on time which corresponds to the selected percentage of light output. So, when you ask for 50%, you get 50% of the light's full output.

The table's adjusted delay-on timings were compiled by experimentation. I started with a table filled with linear delay-on times and took light-output readings using a photographer's exposure meter in a darkened room. To produce the desired light output, each table entry's delay-on time was adjusted to coincide with the appropriate delay.

For the whole shebang to work properly, the delay-on timing must be referenced to each zero crossing. Therefore, the micro needs a nice way to detect every zero crossing. Most methods involve edge triggering. The 60-Hz crossings occur every 8.33 ms; one positive-going edge is followed by a negative-going edge.

Since most interrupts are rising- or falling-edge triggered, you're only going to detect every other edge. You're forced to calculate when the second crossing should occur because you

must use delay-on from that edge as well.

By tapping the full-wave-rectified signal with Schottky diodes, you get double the edges and do not have to resort to estimating the second edge. In addition, the reduced voltage from the transformer's secondary is less likely to cause harm to the micro.

Microchip suggests that limiting the input current to 5  $\mu$ A is enough protection due to internal diodes to  $V_{DD}$  and  $V_{SS}$ . But, I put a zener on the full-wave signal just to be safe. The micro's external interrupt input is set for rising-edge trigger, which can automatically interrupt the program flow whenever a zero crossing is detected.

During the external interrupt's routine, the triac's gate is turned off and the timer is enabled and preset with the on-delay time picked up from the lookup table. This routine requires about 17 instruction cycles. Should lookup take place during serial reception, it delays the bit samples of 1 byte by only 5%

Once the timer overflows, a second interrupt is generated. The timer's interrupt routine turns the triac on and disables itself. This routine requires about 14 instruction cycles. Should the interrupt take place during serial reception, it delays the bit samples of 1 byte by only 4%.

A 7-character packet at 2400 bps requires about two 60-Hz cycles. Any individual character may be affected by up to two interrupts, one from the zero crossing and one from the timer overflow. The delays introduced are not cumulative between bytes. The serial routines can be interrupted by the triac-control interrupts without causing its own reception errors.

Those of you who follow this column know I like to use a 4-MHz crystal clock so execution cycles are easy to count (1  $\mu$ s per cycle). I'm bending this rule a bit for this project because with a timer prescale of 32 (32  $\mu$ s

		Sync	Pre	To	From	Func	Value	Chksum
a)	Sent:	FF	AA	25	6B	01	32	50
	Received:	...	AA	25	6B	01	32	50
b)	Sent:	FF	AA	6B	25	01	32	00
	Received:	FF	AA	6B	25	01	32	00

Table 2—Example command (a) and acknowledgement (b) packets are passed between modules with addresses of 6B and 25. The sync byte is used by the receiver to lock onto the transmitter, so is lost at the receiver.

per tic), the timer's maximum count (256) adds up to 8192  $\mu$ s.

Since one half of a 60-Hz cycle takes 8333  $\mu$ s, the timer doesn't give us time until the next zero crossing. I could use a prescale of 64, but the resolution doubles (64  $\mu$ s per bit].

Instead, I chose to slow the instruction cycle down 10% by using a 3.579.MHz crystal. This increases the resolution to 36  $\mu$ s per tic and timer maximum to 9156  $\mu$ s.

## LOCAL CONTROL

It's nice to have appliances automated. However, there are times when you need to control them manually. Internally, two input bits can be configured using pin jumpers. A mode input determines which module you wish it to act like: an appliance module (relay) or a lamp module (triac). The bps input sets the module's packet data rate as either 1200 or 2400 bps.

In addition to the internal configuration inputs, there are three push-button inputs which are mounted on the enclosure and are accessible to the user. Each input is labeled with its function (On, Off, and Set).

The Set button has multiple functions. When Set is pushed, the module turns on and off, indicating you have entered the Set Address mode. Set This Module's Address mode is found by pushing the On button. To reach Set The Remote Module's Address mode, push the Off button. The module turns on and off again, signifying it understands.

The module address is the address others use to send packet commands to this module. The remote address is the module address you wish this module to contact whenever a button

is pushed (you can use this mode to remotely control another module). The actual address is configured by listening to the line.

Unlike packet transmissions, the address transmission is a continual stream of the address character you need to set the module. When 16 of the same characters have been received consecutively by the module, it saves this as data its address in internal EEPROM and turns the lamp on and off, thereby signifying completion. It then goes back to monitoring the line.

If the module is defined as a lamp module, then the Set button has an additional function. If the Set and On

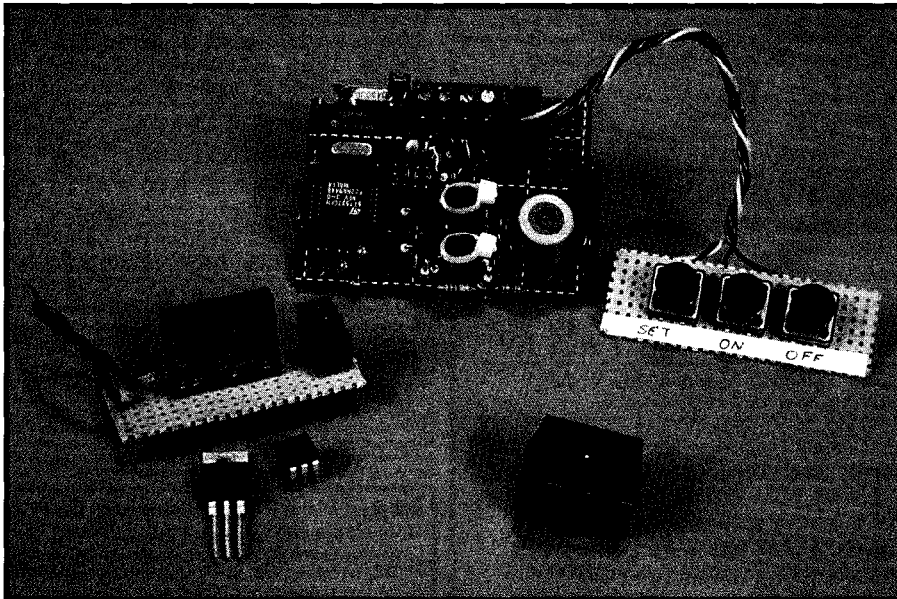


Photo 1--This month's circuitry mimics the X-10's appliance and lamp modules with the added bonus of providing status query.

buttons are pressed down together, the triac's on-delay is decreased, brightening the lamp. If the Set and Off buttons are pressed, the on-delay increases, dimming the lamp. The final level is held in memory to be used whenever the triac is commanded on, retaining its last set value.

## PROGRAM FLOW

After initialization has set up the micro's registers (including port I/O and data rate), the external interrupt is enabled which catches zero crossings and takes care of the triac control in the background. The foreground task scans for local button presses.

Since I do not want things to happen in microseconds, delay loops slow reactions to reasonable user-interface times (i.e., hundreds of milliseconds).

But, I don't want to miss a single packet character transmitted over the line. So, I test for carrier detect whenever I am waiting in a loop. This way I can enter the serial-reception routine as soon as carrier is detected and be ready to receive the packet preamble {AAh}.

As shown earlier, the packet is fixed in length, and a checksum verifies its authenticity. Each packet sent actually contains seven characters. However, since the first character {FFh} is used by the receiver to establish a carrier

detect, it is mid-character before the micro is engaged.

The serial routine looks for a stop bit as a sync to the packet and in turn only receives the following six characters. This packet, including the leading (assumed) FFh, is in memory and dissected to establish its authenticity.

Once it's established as a good packet des-

tined for this address, the command character is interrogated. If the command is an acknowledgment, an internal acknowledge flag is cleared, and the serial routine is exited.

If the command is a query, a response is created (using the original packet) by swapping the address bytes, setting bit 7 of the command (acknowledgment), and plugging the appropriate data into the value byte. The checksum is then recalculated, and the packet is sent off as an acknowledgment.

Otherwise, the command is acted on by setting or clearing an output bit in the micro or changing the delay-on

time of the PWM background task. Finally, an acknowledgment packet is formulated and transmitted.

When not receiving a packet, the main loop continues counting the number of times the button inputs are consistently the same. After a hundred consistent tests, the program branches to a table in which the button pattern becomes the offset into the table.

At the appropriate offset, a second branch directs program flow to the routine pertaining to each key-press pattern. Each routine performs a different function (e.g., turn on the relay, dim the triac, set the module's address, etc.).

Each local push button not only controls the local function (presuming the hardware is there to support it), but also serves as a remote controller by transmitting its function as a command to its remote address. This function helps a master track the on-line modules and their status, or any module you wish to operate remotely, without extra wiring.

## THE END OR JUST THE BEGINNING?

Many functions have been designed into this one device to help spark your imagination about how to use it. The major point here is the acknowledgment of commands and the ability to query modules for present status.

I consider this feature to be a major step in creating a power-line control system which competes with the security and flexibility of other closed-loop systems. Custom circuitry and especially custom packaging are never inexpensive. It is only through high volumes that these kinds of controls become available to home owner at reasonable costs.

X-10 has shown us this is possible, and I thank them for it. The CEBus committee and Echelon with their LONWORKS are attempting to lead us into the future. My plea to them is: Don't complicate protocols to the point where no one can afford to use them. After all, we are the bottom line. □

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

## CONTACTS

ST7537 and ST7537HS1  
SGS-Thomson  
55 Old Bedford Rd.  
Lincoln, MA 01773  
(617) 259-0300  
Fax: (617) 259-4421

PIC16C84  
Microchip Technology, Inc.  
2355 W. Chandler Blvd.  
Chandler, AZ 85224-6199  
(602) 786-7200  
Fax: (602) 899-9210

## IRS

425 Very Useful  
426 Moderately Useful  
427 Not Useful

# BDT™ BASIC DEVELOPERS TOOL

```

F1=Help  F2=F10  F3=Read  F4=Save  F5=Load  F6=Print
Line 128 Col 1 Cursor Indent  W=WARNING BDT
'If not H/V or 45 deg. line, use the Bresenham algorithm
'First, determine the lines major and minor axis
'and assign the appropriate values ex:dmaj=dx or dmaj=dy'

IF absdx>absdy THEN BEGIN 'x is major axis'
  dmaj=dx 'major axis distance'
  dotmaj=x:dotmin=y 'major/minor axis dot address'
  absdmaj=absdx:absdmin=absdy 'abs values of major/minor axis distance'
  dirmaj=dirx:dirmin=diry 'major/minor axis direction flags'
END
ELSE BEGIN 'y is major axis'
  dmaj=dy
  dotmaj=y:dotmin=x
  absdmaj=absdy:absdmin=absdx
  dirmaj=diry:dirmin=dirx
END

'Finally, the Bresenham algorithm draws the line'

c=(2*absdmin)-absdmaj 'initialize error accumulator'
FOR i=0 TO dmaj STEP dirmaj 'step in the major axis'
  IF absdx>absdy THEN BEGIN 'if x is major axis'

```

## ATTENTION BASIC-52, BASIC-180 and BASIC-11 DEVELOPERS!

Finally, an advanced development environment for BASIC single-board computers. BDT combines all the tools you need including **Editor**, **Reprocessor**, **Debugger**, and **Terminal** emulator in a powerful, fast, easy-to-use and totally integrated package.

- ✓ **Editor**
  - Configurable keystrokes and colors
  - Memory-resident text (FAST!)
  - Block move/copy/delete/read/write
  - Find& replace
  - Auto-indent
- ✓ **Reprocessor**
  - Structured programs: DO/UNTIL, WHILE/WEND, BEGIN/END
  - No line numbers
  - Up to twenty character variables and label names
  - Subroutine LOCAL variables
  - Five types of comments (including multi-line) stripped during download
- ✓ **Debugger**
  - Up to 1000 BREAK/PASSpoints
  - Execution PROFILE
  - Up to forty WATCH variables
    - Integer variables WATCHable as DEC/HEX/BIN
    - All, or partial, array, WATCH
- ✓ **Terminal**
  - Editor, file, and compile buffer download to SBC
  - Program capture from SBC
- ✓ **Host PC requirements**
  - 512K
  - One disk drive, one serial port
  - Mono, C/E/V/GA
  - DOS 3.x-6.x

Individual versions are available for BASIC-52(BDT52), BASIC-180(BDT180), and BASIC-11(BDT11). Works with SBCs, from Micromint, Iota Systems, Photonics, Blue Earth Research and others.

**SPECIAL OFFER: now only \$199! ORDER TODAY!**

**MICRO FUTURE** 5 10) 657-0264 • Fax: (5 10) 657-544 1  
40944 Cascade Place  
Fremont, CA 94539



# PC Times in Silicon Valley

## SILICON UPDATE

Tom Cantrell



As I write this, Windows 95 has hit the streets amidst great fanfare. As a Mac user, I'm somewhat bemused by the idea of people lining up at midnight as though the OS might work better if it's fresh. At least the early birds had a hope of getting through on the help line before it jammed up.

Along with all the Windows 95 hoopla, visits to a couple of my favorite summer shows—Hot Chips and SVPC (Silicon Valley PC)—only reinforce the fact that the PC is hot.

### SVPC

Over the years, conferences and trade shows have exhibited a “nichification” trend, targeting ever-narrower audiences. SVPC is certainly a good example of this. It now targets the chosen few who design PC hardware and write the low-level system software [e.g., BIOS, I/O drivers, firmware, etc.).

However, even if you aren't a PC designer, you might find SVPC interesting on a couple of fronts. First of all, you find what the PC suppliers plan to do for (or to) you with next year's models. Secondly, anyone designing electronic gear needs to monitor what's happening in the PC world since it affects everything from batteries to DRAMs.

Consider the problems associated with designing the power subsystem for any portable gizmo. What I'd like to see is an embedded UPS which combines power supply and battery charger in a single unit (i.e., it powers the system and charges batteries) at the same time. The “EUPS” should handle multiple batteries simultaneously with automatic switching between draw and charge as required (i.e., nonstop battery switching).

What the heck, let's up the ante with mix-and-match capability for the popular battery types (NiCd, NiMH, lithium, zinc, lead acid, and rechargeable alkaline, etc.). The unit should feature a wide input range (e.g., 3-18 VDC) and multiple precision-regulated outputs. Oh yeah, mustn't forget the accurate (not guesstimated) Gas Gauge and Party's Over outputs.

As shown in Figure 1, the Intel-developed System Management Bus (SMBus) along with compatible smart batteries from Duracell and power management chips from Maxim, Linear Technology, Benchmark, and Microchip go a long way toward making my tall order a reality.



Back on the conference circuit, Tom gives

us the latest on Hot Chips and Silicon Valley PC. News on buses, DRAM, and SCSI gives way to the latest “limit” in silicon, which seems to have more to do with wallets than walls.

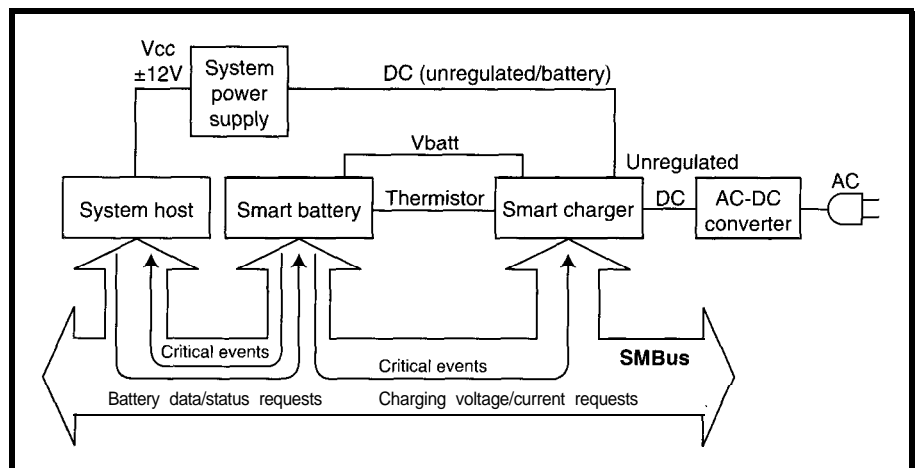


Figure 1—Though SMBus (System Management Bus) was spawned in PCs, any application can fake advantage of its power-management capabilities.

Thankfully, SMBus is a simple two-wire clock and data interface based on ACCESS.bus (see *INK* 28 "The Ultimate Desk ACCESSory?"), itself based on the ubiquitous I<sup>2</sup>C bus. I'm discouraged by the proliferation of yet another I<sup>2</sup>C variant, but according to Intel and the parties involved, they'll be migrating toward each other.

For instance, the next version of ACCESS.bus (V3.0) includes SMBus as a protocol option. Despite the gotchas, it's certainly better than dealing with something completely new or overly complicated, and it's quite possible to take advantage of existing I<sup>2</sup>C knowhow and chips.

## ROCK 'EM, SOCK 'EM DRAMS

Thanks largely to the antics of Microsoft, DRAM biz is crazy. This trend shouldn't be a surprise since Windows 95 upgrades collectively demand a terabyte or so of DRAM.

I'm pleased to report my previous prognostics about DRAMs are proving accurate. Sure, it's only about a year since I made my predictions (*INK* 55), but that's pretty good in this fast-moving era when pundits' pontifications are routinely punctured in short order (remember PDAs?).

The gist of the earlier article was that so-called Fast-Page Mode (FPM) has emerged as the conventional DRAM fast-access mode of choice, dispensing with other contenders (i.e., nybble and static column modes).

Poised for imminent success is an FPM variant called Extended Data Out (EDO). EDO involves minor changes in the role of \*CAS. In an FPM DRAM, ● CAS both latches the column address (the leading edge) and turns off the output driver (the trailing edge). For EDODRAMs, \*CAS's role is limited to the former (i.e., data out remains valid when ● CAS goes high-hence, the "extended" moniker). Instead, the output is turned off with the rising edge of 'RAS.

The result of this seemingly minor tweak is a rather surprising 30–50%

	SDRAM	BEDO
No. of Banks	1 or 2	1
Read Latency	1, 2, or 3	2
Burst Length	2, 4, 8, 512	4
Burst Sequence	Linear, Interleave	Linear, Interleave
Programmable by:	WCBR	WCBR
Burst Advance	CLK	CAS
RAS Control	Pulsed	Level*
Byte Control	New DQM	CAS*
x32 Module	None	72 pin*
x64 Module	168 or 200 pin	168 pin
Supply Voltage (VCC)	3.3 v	3.3 V (5 V tolerant)
Relative Die Size	1.03-1 .05	1.0*
Defined by:	Committee	PC architects, chip set designers, and Micron

\* Same as fast page and EDO

Table 1—BEDO (Burst Extended Data Out) DRAMs adopt the on-chip burst counter; programmable burst sequence, and pipelining of SDRAMs (synchronous DRAMs). However, they largely retain traditional DRAM control signals and pinout.

claimed increase in bandwidth. Freeing the trailing edge of \*CAS from its previously critical timing role streamlines the entire access. The "theoretically" small advantage of EDO over FPM is amplified when the difficulties of generating theoretically perfect FPM ● CAS timing are considered.

Along with the technical whizzos, EDO's success is assured by marketing and production realities that tripped up many specialty memory hopefuls. In particular, the changes are so minor that most major DRAM suppliers are implementing EDO as a bond-out option on their standard FPM DRAMs. Not only in principle, but likely in practice, EDO will match FPM prices. You don't have to be Milton Friedman to figure that something for nothing is a pretty good deal.

Despite the improvements, EDO DRAMs won't satisfy insatiable bandwidth demand for long. Waiting in the wings beyond 50 MHz are synchronous DRAMs (SDRAMs), expected to take buses from 66 to 100 MHz and beyond. As fully described in *INK* 64, SDRAMs rely on a pipelined, multi-bank architecture in which the control signals and data are sampled and driven with a high-speed clock.

Technically, everyone seems to agree that SDRAMs are ultimately the right way to do DRAMs [i.e., bandwidth per die area is intrinsically superior to async DRAMs]. Also, SDRAMs have JEDEC blessing. As far as I can tell, most every major DRAM supplier is planning to make SDRAMs.

What's new since *INK* 55 is the emergence of another alternative: Burst EDO (BEDO), which is positioned to fill a number of gaps between today's FPM or EDO and tomorrow's SDRAMs.

One issue—and the DRAM folks aren't alone in facing it—is the forced march to 3.3 V. For a long time, this has been something you could worry about tomorrow. But, tomorrow is finally here. The SDRAM folks simply decided that 3.3 V is where it's at and declared victory. Along the

same onward-and-upward lines, SDRAMs are targeted for at least 16-Mb density and a x 8 pinout.

The power and pinout differences, sluggish 4–16-Mb crossover, and incompatibility with current SIMM technology all conspire to make SDRAMs a reach in the short term.

Stepping into the breach, at the prodding of Micron Technology, BEDO retrofits a couple of the best SDRAM features while retaining the basics (i.e., asynchronous design and pinout) of the traditional FPM and EDO DRAMs. Furthermore, BEDO finesses the 5- or 3.3-V issue by cleverly specifying 3.3-v power with 5-V tolerant I/O.

One BEDO addition is a built-in burst counter so only the initial 'CAS in a burst needs a column address, marking the return to nybble mode. Though it's still async, BEDO pipelines addressing and data transfer. Together, these SDRAM features boost claimed burst bandwidth another notch to roughly two times FPM DRAMs.

Table 1 (taken from a proceedings paper by Bob Fusco of Micron Technology) sums up the proponents' view that, even conceding the merits of SDRAM, there's not only room, but a need, for BEDO in the interim.

The outlook for BEDO is tough to call. It sounds pretty good, but the claimed performance advantages need to be verified in a specific design. The practicality quotient is high, but announced support for BEDO isn't as strong as either EDO or SDRAM.

	EIDE	Ultra SCSI	1394	SSA	FC-AL
Speed (MHz)	4.2 5.5 a.3	20	100 200 400	200 400	266 531 1062
Transceiver Technology	1.0 $\mu$ CMOS	0.6 $\mu$ CMOS	0.6 $\mu$ CMOS	0.5 $\mu$ CMOS	GaAs BiCMOS
Number of Devices	2	7/1.5 m 413.0 m	63	127	127
Distance between	0.5 m	3 m (SE) 25 m (D)	4.5 m 4.5 m	20 m 660 m	10-90 m 10 km
Max Distance	0.5 m	3 m (SE) 25 m (D)	72 m 72 m	2.5 km	62.5 km

Table 2—The chart highlights the main features of a variety of disk interfaces. EIDE and Ultra SCSI are parallel interfaces while the others are serial, so the former's speed numbers compare more favorably, assuming the metric of interest is bandwidth rather than just clock rate.

Though there may be a few gems (RAMBus for performance at any price applications or the so-called WRAM, which reportedly does well in graphics boards), it seems like most of the other contenders (call them A-ZDRAMs) are falling by the wayside.

#### IDE SAY IT'S A SCAM

My old PC has an ST506 disk. So, I'm totally unknowledgable (impartial)

in my view on PC disk interfaces.

And, my view is quite simply "What the !@#\$ is going on?" It looks as though all the confusion mongers that were setting up camp in the DRAM market packed up and moved over into disk interfaces.

I mean, what's a disk-interface-challenged guy like me to make of something like Figure 2? The road map [from a presentation by Dr. Robert

Selinger of Adaptec) looks more like a lo-disk pileup and reminds me of how much I like my Mac.

Of course, Mac fans shouldn't gloat, since whoever is in charge of these things isn't content to leave well enough alone on the SCSI front either. There's talk of Ultra SCSI, SCSI-3, and multiple flavors of serial SCSI, including IEEE P1394 and SSA [Serial Storage Architecture]. My vote for the dubious acronym of the year goes to SCAM, which stands for SCSI Configured AutoMagically and purports to add automatic ID and hot-plugging. Looks to me like it'll be magic if you can get a disk-any disk-to work.

But wait, there's more. Looming on the horizon is the long-awaited Fibre Channel that has, as pending standards are prone to do, evolved from a simple high-speed point-to-point link into a steroid-dripping brute. Judging by Table 2, which compares the major interfaces, Fibre Channel can certainly claim bragging rights.

Sorry I can't really divine which way the wind is blowing on this disk

# ALL ELECTRONICS CORP.

QUALITY PARTS . DISCOUNT PRICES . HUGE SELECTION

## C-30 AUDIO CASSETTE



Brand-new, good-quality 30 minute (15 min. per side) audio cassettes. Excellent for demos or instruction tapes.

Manufactured without record-prevent knockout tabs, so the indentations must be taped over before use.

**CAT # C-30 3 for \$1.00**

100 for \$30.00 • 500 for \$125.00

ORDER TOLL FREE

**1-800-826-5432**

CHARGE ORDERS to Visa, Mastercard or Discover

TERMS: NO MINIMUM ORDER. Shipping and handling for the 48 continental U.S.A. \$5.00 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include local state sales tax. Quantities Limited NO COO. Prices subject to change without notice.

CALL, WRITE, FAX or E-MAIL for a FREE 64 Page CATALOG Outside the U.S.A. send \$2.00 postage.

MAIL ORDERS TO:  
ALL ELECTRONICS CORPORATION  
P.O. Box 567  
Van Nuys, CA 91408  
FAX (818)781-2653  
E-Mail - allcorp@allcorp.com

## COMMUNICATION HEADSET w/ ANTENNA

Headband with electret microphone, dynamic earphone and antenna. Mouthpiece position and headband are adjustable. Cable is 3' long with 6 wire termination. These are new units that appear to have been removed from walkie-talkie radios.



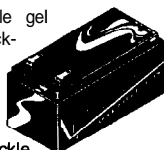
**\$5.00 each**

CAT# PHN-8

10 for \$45.00

## RECHARGEABLE GEL CELL 12 VOLTS 3 A/H

Maintenance free, rechargeable gel cell battery. An excellent back-up power source for alarms, communications equipment, lighting or computers. Can be used in any position and



**\$15.00 each**

can be trickle charged for long periods of time. 5.3" X 2.63" X 2.39"

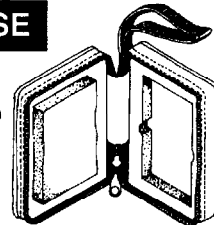
CAT# S12-3

case of 10 - \$120.00

Visit Our World Wide Web Site...  
<http://www.allcorp.com/allcorp/>

## CANVAS CASE

Good-looking, well constructed 7.5" X 6" X 1.95" canvas carrying case with zipper on three sides. Webbed nylon hand strap and ample foam padding inside. Probably designed for transporting memory storage media. The black foam padding inside can easily be reconfigured to accommodate photograph audio or video components. The removal of one lar chunk of foam gives you a well padded interior volume of 5.9" x 4.3" x 0.75". All black with red and white embroidered "K-stor" (tm) logo on front.



CAT# CSE-3 **\$3.75 each** 10 for \$35.00

## 3 WIRE POWER CORD

Black, 5'3" power cord with standard 3 prong grounded plug one end, stripped and tinned wires other end Flat SPT-1 type insulation 0.3" X 0.11" thickness. 105 degree C. CSA listed

CAT# LCAC-34

**\$1.25 each**

10 for \$10.00



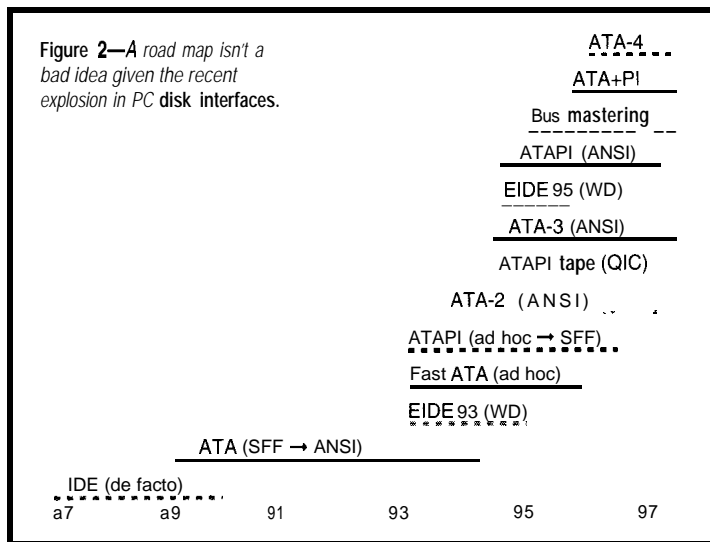
stuff. My only recommendation is that if your job has anything to do with disk drives, pay close attention lest you end up riding the wrong horse.

### HOT CHIPS VII

If SVPC is where PC designers go, Hot Chips is the place for an even more elite bunch—the gurus who design the CPU chips themselves.

Held at Stanford University, the conference traditionally has an academic flavor. Notably, it's been the forum for the RISC revolution that's kept computer architects busy for the last decade or so.

However, over time, the show has taken on a more commercial air with more and more of the presenters affiliated with companies, not universities. The reasons for this shift are becoming apparent and have big implications for the entire IC business.



The issue was made clear in a presentation by Silicon Valley's Grand Old Man himself, founder and president of Intel, Gordon Moore. A soft-spoken, intellectual fellow (always left the rough stuff to Andy "What Pentium Bug?" Grove), he conveyed the message gently but authoritatively in a presentation entitled "Nanometers To Gigabucks" (see Figure 3).

The good news is the long-feared wall (i.e., physical limits to continued improvements in IC density) is still over the horizon. The indisputable fact that there has got to be a wall is discounted by its repeated failure to make an appearance. A notable escape hatch is ever-lower operating voltages, reinforcing the fact that the good-old 5-V era is fading fast.

Thus, for the short term anyway, it's a mistake to focus on what can

be done when in fact the issue is what can be paid for.

Unfortunately, development costs reflect the incredible complexity of the latest CPUs and are easily more than 100 times (and heading toward 1000 times) what they were in the good old 8-bit days. Of course, rather well-heeled Intel can afford two separate development teams (and all their su-

# WE'RE NOW

## 80C52-BASIC CHIPS IN VOLUME

Micromint's 80C52-BASIC chip is an upgraded replacement for the venerable Intel 8052AH-BASIC chip. Ours is designed for industrial use and operates over the entire industrial temperature range (-40°C to +85°C). Available in 40-pin DIP or PLCC.

**80C52-BASIC chip \$19.00**

**OEM 100 qty. \$12.00**

**BASIC-52 prog. manual \$15.00**

# MICROMINT, INC.

Call (860) 871-6170 or 1-800-635-3355 to order  
MICROMINT, INC. 4 PARK STREET, VERNON, CT 06066

## This Parts List Software is for Engineers / Designers

Not MRP: P&V actually assists development!

Windows™ program helps you stay organized

Keep track of:

- Drawings & Specs
- Suppliers & quotes
- Product material costs
- Engineering stock

# Parts & Vendors™

...for independent and departmental projects. Use alone or in a workgroup.

**\$99 + shpg** **Call 800.280.5176**

**Trilogy DESIGN™** voice. 916.273.1985  
fax. 916.477.9106  
P.O. Box 2270, Grass Valley, CA 95945

Requires 486, 8meg min. RAM, Windows™

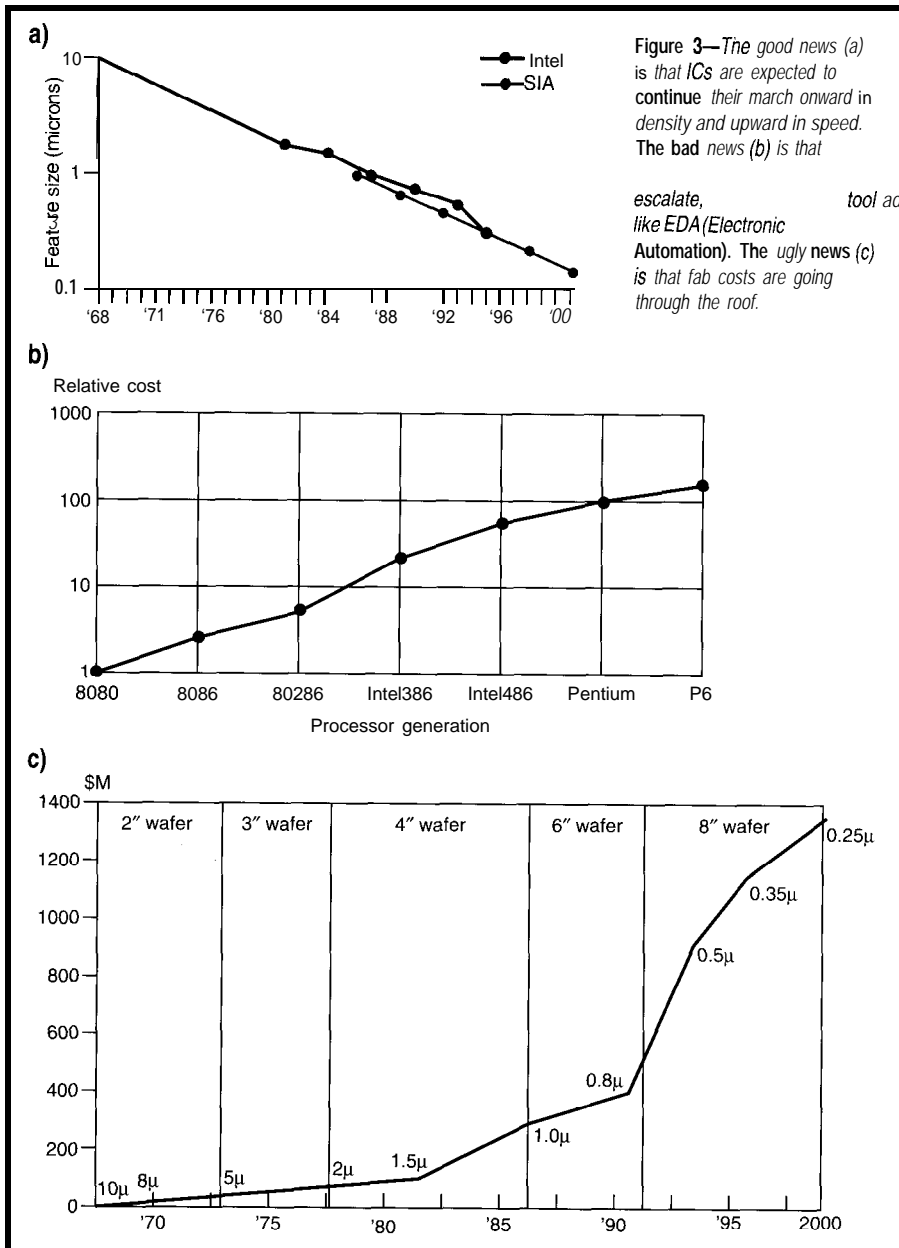


Figure 3—The good news (a) is that ICs are expected to continue their march onward in density and upward in speed. The bad news (b) is that

escalate, like EDA (Electronic Automation). The ugly news (c) is that fab costs are going through the roof.

Most of the past and present stuff is well known. You remember how the 4004 and 8008 weren't intended to be microprocessors but rather expeditious ways to deliver dedicated calculator and terminal chips. It's less well-known that the '86 was an emergency-stopgap measure to cover delays in the ill-fated iAPX432. The 8088 (8-bit bus) was considered an afterthought until a rather contrived "16-bit performance, 8-bit price" Intel marketing campaign managed to sell a few customers. One was IBM, and the rest is history.

Interesting to be sure, but there's no doubt everyone was most anxious to hear about the future—as in P6—part of the pitch, and we weren't disappointed.

The P6, pictured in Figure 4, actually consists of two chips [the CPU and a Level-2 cache] packaged in a 387-pin PGA. Some might call it an MCM (Multichip Module), but it's actually a "no-substrate two-die dual-cavity PGA."

The technical aspects of bundling the L2 cache pale in comparison to the business impact. Looks like those who've been living off the external cache SRAM biz may as well schedule an appointment with Dr. Jack.

The RISC versus CISC debate (with CISC being a pseudonym for 'x86) has raged for years. The good news for the RISC fans is they can now declare victory. The bad news is the 'x86 is now a RISC, which just happens to be able to digest 'x86 binaries.

Let's see how it works with the caveat that, since each little box on the block diagram contains several million transistors [i.e., 5.5 M for the CPU, 16 M for the L2 cache], I have to keep the discussion at 30,000 feet.

The 'x86 codes wind their way into the CPU through the caches. Unlike other beyond-Pentium chips (such as the AMD K5), the code in the L1 cache is still good old 'x86 format.

Next, the In-Order Front End grabs big chunks of a dozen or so 'x86 instructions and translates them into so-called U O Ps, which are simply RISC instruction sequences that duplicate the CISC instructions compound functions. For instance, an 'x86 instruction that compares a register with memory

perexpensive EDA gear) operating in parallel, interleaving even and odd processor numbers.

Even presuming you can design a multimillion transistor chip, the really tough question is who, with a state-of-the-art fab now topping \$1B, can afford to build it?

In other words, the problem isn't the wall, but the wallet. Those who can't ante up won't be dealt in. It looks like the winning strategy in the IC arms race is going to be to spend 'em into the stone age.

### 'x66 GETS RESPECT

Traditionally, the academic types have pooh-poohed the 'x86 as another

example of crass commercialism gone awry. But now, the monetary frenzy surrounding the PC, 'x86, Windows hegemony can't be ignored. Heck, all their students have to get jobs someday, don't they! So, the agenda included quite a few 'x86 presentations.

One particularly interesting one was "'x86 Generations: Past, Present, and Future" by John Wharton of Applications Research, a self-described "inveterate Intel watcher," who manages to walk that fine line between black-listed gadfly and cozy insider. He's also a funny guy. So, if he tells you the one about how "Urn" is Navajo for '86 (as in Pentium), don't bite.

turns into two UO Ps, a load, and a compare. This wouldn't be a very good deal, except for the fact the P6 can issue up to six UO Ps in parallel.

Given the historically meager register set, it's quite likely a chunk of 'x86 instructions contains some that step on the same register. Consider a sequence that loads a register, outputs it to an I/O device, and then loads it again. Normally, the second load, not to mention any subsequent instructions that depend on it, can't be issued until the I/O completes.

Rather than bring everything to a screeching halt, the P6 front end employs a technique called *register renaming* that maps a single logical (i.e., architecture visible) register to multiple hidden physical registers. In the above example, the second load pro-

ceeds with the result placed in a re-named register. Once I/O is complete, the renamed register is simply re-mapped to the real register.

As the name implies, the front end's stream of UO Ps retains existing program order, which is a marked

contrast to the recipient, (i.e., Dataflow Execution Engine).

The dataflow concept isn't really new or hard to understand. Any CPU running for elected office must simply remember "It's the data, stupid."

Imagine a computer with infinite execution units, unlimited cache, perfect branch prediction, and so on. The immutable barrier to performance that remains is data dependency. Consider the simple program:

```
A = B + C
D = A + 1
```

No matter how much hardware you throw at it, the second instruction can't be issued until the first one completes. Thus, the ultimate speed limit for any program can be represented by a data-dependency graph.

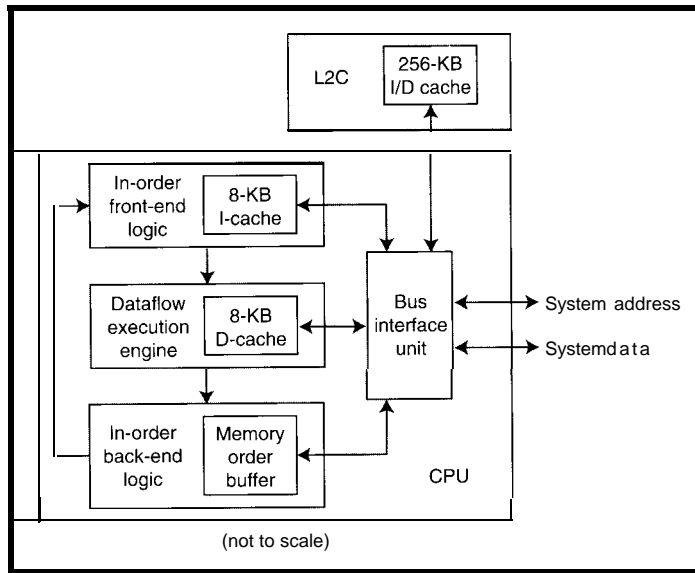


Figure 4—It's not your father's CISC. Dataflow architecture and 20+ million transistors give the P6 RISC-like punch.

# PTAC™

With Micromint's PTAC, each person, cow, shipping container, or vehicle can have a unique coded transmission which keeps track of its location.

PTAC has two function-button codes, an intermittent tracking locator signal, and a low-battery code which are user-selectable among 65,535 combinations. The intermittent tracking signal is programmable from 0.25 sec. to 6 min. and runs for months on a battery.

We offer a sample 300-MHz keychain transmitter with two function buttons. It has factory-preset codes, a 12-second tracking signal, and a 25-30-foot range. We also offer a 300.MHz receiver with signal level indicator that automatically recognizes all transmitters within range and queues the codes until polled via RS-232 or RS-485 network. Hundreds of receivers can be used together (via repeaters) to make a vast tracking system.

- ☆ Inventory Management
- ☆ Automatic Tracking
- ☆ Personal Identification
- ☆ Access Security



PTAC keychain \$49.00  
 PT-REC1 300-MHz network/serial receiver \$249.00  
 PTAC Evaluation System: comes with four PTAC keychains, two receivers, power supplies, and demo PC software \$595.00  
 RS-485 repeater w/power supply call for pricing

CALL  
 1-800  
 635-  
 3355  
 TO  
 ORDER



**MICROMINT, INC.**

4 Park Street • Vernon, CT 06066 • (860) 871-6170 • Fax (860) 872-2204

## CADPAC

PCS AND SCHEMATIC TOOLS  
2 for the price of 1

PCB II & SUPERSKETCH FOR NEW LOW LOW PRICE

only \$159 US

\*\* features comparable to packages costing thousands!  
 \*\* must be tried to be believed  
 \*\*\*easiest product to use for designing PCBs'  
 \*\* customers call it "the 8th wonder of the world!"

!!REDUCED!!

R4 SYSTEMS INC.

NEWMARKET ONTARIO  
 UNIT 20, 3771, SUITE 11 S-332  
 905 898-0665 F.A.X. 905 898-0683

FREE DEMO  
 WRITE OR CALL  
 TODAY

Email [r4system@astral.magic.ca](mailto:r4system@astral.magic.ca)

The execution engine features so-called *dynamic execution*, 'which is a new buzzword that encompasses every other buzzword and then some: super-scalar (peak 5 instructions per clock), superpipelined (a whopping 17 stages between entrance and exit), speculative and out-of-order execution, nonblocking caches [i.e., hit proceeds, though previous miss is pending), bypassing, streaming, and so forth. Though none of these are new concepts, I've never seen them used so aggressively, even in the gigantic mainframes of yore.

The **U O Ps** streaming from the front end are shoved into a 20-entry "reservation station." From there, the execution unit uses heuristic [i.e., voodoo) techniques to schedule instructions for the half-dozen or so functional units. The goal is to execute any instruction that can be (i.e., data is available) so that in turn you enable the execution of instructions dependent on that result. In dataflow-speak, the P6 execution unit tries to flatten the data-dependency graph.

With execution proceeding in parallel, speculatively, and out of order, what's going on inside the chip barely resembles what the programmer had in mind. Fast interrupt response is good, but executing the handler before the interrupt occurs is going too far! It's the back end's responsibility to put everything in proper order (i.e., the same order as a classic CISC 'x86), quit speculating, and irrevocably commit to a visible state CPU.

## THE END OF ARCHITECTURE?

Intel's got to be given a lot of credit for stretching the ancient 'x86 architecture so far. Sure, they've got a lot of money, but it's still an achievement akin to winning the Indy 500 in a hopped-up milk truck.

However, some argue the payoff for the incredible complexity seen in the latest generation of CPUs remains to be proven. I tend to be among the skeptics who wonders if everybody wouldn't be better spending their money for a faster disk.

Less flippantly, the key is understanding that the performance is a

Model	SPECint92@MHz	SPECint/MHz
UltraSPARC	240 @ 167	1.4
R10000	300+ @ 200	1.5+
SPARC 64	256 @ 154	1.7
620	225 @ 133	1.7
PA8000	360 @ 200	1.8

Table 3—One measure of architectural merit is SPECint92/MHz, and the latest machines (including the P6) all achieve about 1.5. Whether the number will continue to improve and at what cost, is the question.

byproduct of architecture and implementation (i.e., clock rate), and there's a tradeoff.

It's easy to compare clock rates, but what about architectures? One popular technique uses SPECint92/MHz as a metric, thus also appropriately measuring the effectiveness of the particular machine's C compiler.

As shown in Table 3 (taken from a presentation by Andrew Essen and Stephen Goldstein of HaL Computer Systems), most of the latest CPUs, like the P6, deliver about 1.5 SPECint92/MHz. By comparison, the original 5-MHz 8088 PC achieved a meager 0.2.

Fifteen years of computer architects' blood, sweat, and tears boils down to about an eight times performance improvement. However, as of the last few generations of chips, it's become very difficult to increase the number. At this point, it takes hundreds of thousands of transistors to get even a few percent of architectural speedup.

By contrast, over the same time frame, the process and circuit folks delivered a whopping 30 times (i.e., 5-150+ MHz) and, as per the discussion about the wall, still have legs (though shod with ever-more-expensive shoes).

All this adds up to the conclusion that processor architecture is running out of gas. Certainly, there'll be more attempts to brute force the issue. Attention will focus on the compiler as well (the premise of so-called VLIW—Very Long Instruction Word—which will probably be called SuperWide once marketing gets hold of it).

Nevertheless, getting more juice out of a single processor is getting hard enough that commercial necessity is shoving multiprocessing out of the cloistered academic closet. The expectation (hope?) is that two Px on one die

can pack more punch than a single Px + 1 of the same size. It's really the old SMP (symmetric multiprocessing) technique shrunk to chip level. The idea of SMP-on-a-chip has already achieved buzzword (multiscalar) status.

One thing that is very clear from experience with box-level SMPs is you've got to have a good operating system that can nimbly weave all the separate threads into useful work

I'll bring this article full circle by pointing out that Windows NT has some pretty neat SMP features built-in. Slap the Mac (oops, I mean Windows 95) interface on that sucker, and you've got an OS any multiscalar could learn to love. Windows 96 anyone? ☹

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached at (510) 657-0264, by fax at (510) 657-5441, or at tom.cantrell@circellar.com.

## CONTACTS

Hot Chips  
IEEE  
701 Welch Rd., Ste. 2205  
Palo Alto, CA 94304  
(415) 941-6699

SVPC  
Systech Research  
1625 The Alameda, Ste. 207  
San Jose, CA 95126  
(408) 293-8383

Applications Research  
P.O. Box 60231  
Palo, Alto, CA 94306  
(415) 856-8051

SMBus  
Intel  
(800) 253-3696  
Fax: (916) 356-6100  
BBS: (916) 356-3600

428 Very Useful  
429 Moderately Useful  
430 Not Useful

# CONNECTIME

I am building a new home and am employing radiant slab heat throughout. I want to monitor the actual temperature of the cement slab subfloor in various zones in the house. I am using the DS1820 from Dallas Semiconductor, which does the A/D conversion on board and sends the digital result over a common data line.

Much of the floor in the house is tiled with Mexican Saltillo tile and in those locations I plan to place the IC directly on the slab between two adjacent tiles, in the grout joint. It will be covered with almost an inch of mortar. The cable emerges from the mortar about one foot later and tunnels into the wall and then up into a junction box.

In carpeted areas I plan to simply rout out a channel in the concrete near the wall and place the IC and cable into the groove.

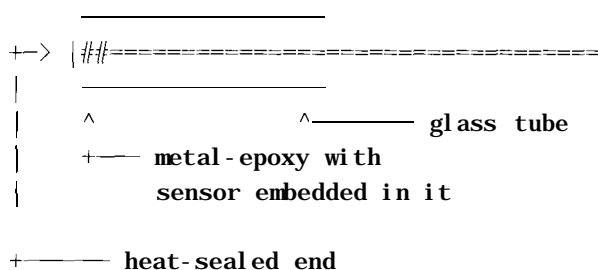
Should I use a potting compound? What is available that conducts temperature well but is also a good insulator? What about the stuff electricians use to seal spliced wires which are to be buried?

Thanks in advance for any feedback from users with experience with potting compounds or whatever.

**Msg#: 4837**

**From: Ken Simmons To: Paul Glasser**

I'd embedding that Dallas chip in metal epoxy (i.e., J.B. Weld or similar) in a sealed-end glass or metal tube like this:



Make sure you heat-shrink the wires and solder junctions and encapsulate the leads in waterproof silicone caulking to prevent the metal epoxy from shorting out the Dallas device's wires and solder joints. The probe doesn't have to be very long (3 inches?). Just make sure you completely seal both ends so there's no chance of moisture infiltration.

Glass will probably be easier to work with, just because you can see everything, however metal (e.g., copper or thin-gauge steel) might be cheaper as well as stand up to the stresses of drying mortar (i.e., shrinkage) better.

Either way, make sure your cable is Teflon-jacketed or similar waterproof-material coated!

**Msg#: 4842**

**From: Paul Glasser To: Ken Simmons**

Thank you Ken for your great idea for embedding a temperature sensor in mortar. The idea of using a pipe for protection against movement during mortar curing and the use of metal weld for a good moisture barrier that also conducts heat well are good ones. There should be no moisture present after curing (since the floor will be heated), but just in case I will probably substitute electrical splice compound for the metal weld and enclose the whole affair in a stainless steel pipe instead of copper.

**Msg#: 4865**

**From: Ken Simmons To: Paul Glasser**

I'm pleased my little idea had merit for you.

As to lack of moisture: don't count on it! Edsel Murphy can always find ways of introducing unwanted moisture wherever moisture isn't supposed to be (e.g., broken hose, outside/ground seepage, etc.).

Your decision to use stainless instead of copper is an excellent one, IMHO. I merely suggested copper because it's (generally) cheaper and definitely easier to work with.

**Msg#: 4824**

**From: Matthew Levine To: Paul Glasser**

Here's a little trick I used for protecting small circuit assemblies for use in model airplanes and model cars. I dipped the assembly in lacquer and let it dry.

I repeated this process four times, ending up with a small package sealed right up to and including the first inch of cable. I don't know what the thermal properties are of the stuff, but you may want to try a cabled thermistor in a few different commonly available liquids-that-harden and see what happens.

**Msg#: 5390**

**From: Paul Glasser To: Matthew Levine**

Thanks Matt for your suggestion for using lacquer as a potting compound. Novel idea! I'll remember it for future projects. I've decided to go with a stainless steel pipe and the goo electricians use to seal underground splices, which should all work to protect from the effects of mortar movement as it cures.

---

## Hydrophone design

**Msg#: 5482**

**From: Barry Klein To: All Users**

Anyone into building hydrophones? I was wondering how to best go about making a mic for one and also if noise-

# CONNECT TIME

conducted by Ken Davidson

The Circuit Cellar BBS  
300/1200/2400/9600/14.4k bps  
24 hours/7 days a week  
(860) 871-1 988-Four incoming lines  
Internet E-mail: [sysop@circellar.com](mailto:sysop@circellar.com)

*Response to our new Web page has been tremendous. Keep the feedback coming. If you haven't seen it yet, connect to <http://www.circellar.com/> and check it out.*

*This month's message threads cover quite the diversity of topics. First, we look at a simple circuit for generating 8-bit random numbers. Next, we consider the issues surrounding embedding a temperature sensor in mortar. Finally, we look at puffing together a homebrew hydrophone and talk about some noise-reduction ideas.*

## Random generator circuit

**Msg#:13873**

**From: Mike Smith To: All Users**

I am looking for a circuit that will, at powerup, generate a 4-8-bit or higher random digital word, preferably in CMOS. Thank you.

**Msg#:13957**

**From: Russ Reiss To: Mike Smith**

One common approach to random number generators is to simply have a counter, which usually runs at a rather fast rate, and to freeze the count on some (statistically) independent event.

In your case, perhaps you could use an 8-bit counter (CMOS, RC clock adequate, etc.) and a flip-flop that freezes the count and enables the output. The FF could perhaps be triggered by a zero-crossing of the AC line (if one is present). Since the time the circuit starts and the time of occurrence of the zero crossing are statistically independent events (in most normal hardware configurations), the number frozen in the counter will be random. Ideally, you'd like to guarantee that you pass through a bunch of full counts within the maximum time a zero-crossing could be delayed (half cycle at 60 Hz, or around 8 ms).

I think you get the idea; the circuit would be rather simple. A ubiquitous PIC chip would do nicely, and might cost no more than the collection of small-scale parts using other approaches. (I know; someone's gonna say, "Gosh, can't you build ANYTHING without a PIC in it?" Why bother, when that single chip works so well in so many ways!) But there are many different implementations possible depending on your needs.

**Msg#:14972**

**From: Ken Simmons To: Mike Smith**

Have you thought of a pair of simple 7490 counters feeding a latch? Unless I'm mistaken, when they're powered up, they'll have a "random" 4bit number on their outputs.

**Msg#:21006**

**From: Pellervo Kaskinen To: Ken Simmons**

Don't be fooled to expect that the randomness mentioned for the 7490 (or other counters for that matter) is random for any particular unit. It is intended by the manufacturers as a warning that the next unit or the next production batch (or...) does not wake up in the same state as the one you based your design on.

Given that the power on of most commercial power supplies is also somewhat statistically related to the power line waveform, a true randomness may be difficult to achieve in larger counts. But for the relatively short number range of 0 to 255 decimal, it might just be random enough for the intended practical applications, especially with high oscillator frequencies. On the other hand, what is the behavior of the oscillator on power up?

If the oscillator is slow starting, and the first sample is taken early in the process, then the result may be far from meeting the criteria for randomness.- Again, this may or may not be a problem in the practical implementation.

By the way, 7490 is not the low-power (CMOS) type Mike was looking for. But there are plenty of choices for those. The first counter chip coming to mind is the 4040, a 12-bit counter. An RC oscillator could be built with a single or dual NAND or a couple of stages from a hex inverter. The two-stage design is the better one, with the single-stage design generally requiring a Schmitt trigger version such as 4093.

---

## Potting compound

**Msg#: 4822**

**From: Paul Glasser To: All Users**

I am soliciting suggestions on how to protect an IC and connecting three-wire 22-gauge cable which will be embedded in mortar.

# CONNECTIME

canceling techniques could be used to eliminate local boat-motor noise.

**Msg#: 5673**

**From: Russ Reiss To: Barry Klein**

Noise canceling, as I suspect you are thinking-with another pickup of the noise source for cancellation in a summing amplifier-can be tricky! Not only do you need the right gain, and a noise that is very similar to what the main mic is picking up, but it also must be in phase with the noise picked up by the mic. You might find a lucky combination, but it probably will take lots of experimentation; not so easily done under a boat!

Often, though, the hydrophone is dropped on a long line (sometimes with an amp at the mic to overcome losses and noise pickup on the long cable) far away from the noise source. All depends on what you're listening for.

**Msg#: 5615**

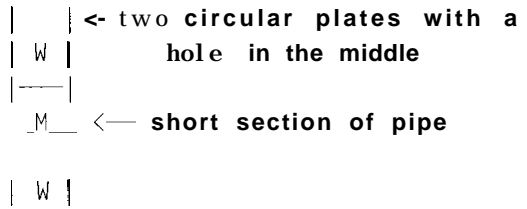
**From: James Meyer To: Barry Klein**

The microphone for a hydrophone system is simply called a hydrophone.

A hydrophone can be built just like you would build an ordinary microphone, except because of the pressure of the water, it must be built quite a bit stronger.

If you can find a one-pound spool of magnet wire of 32-38 gauge and a magnet, you can build a workable hydrophone at home. The magnet can be salvaged from an old speaker. One of the old-style cylindrical alnico magnets would be best. About an inch in diameter and an inch thick is ideal, but feel free to improvise.

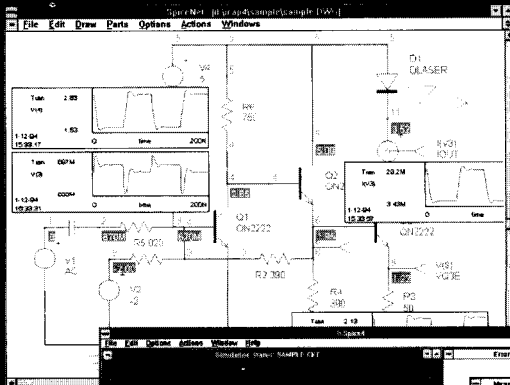
Make a spool from some scrap plastic sheets and a section of PVC water pipe like this:



Cross section

# Interactive

Experience The First and Only Interactive SPICE 3 Simulator



**ICAP/4, Intusoft's Virtual Circuit Design Lab, lets you:**

- Analyze and Simulate all types of designs
- System, Board, and IC level
- Electrical / Mechanical / Physical Power, ASIC, RF, Mixed Mode
- User Friendly and Affordable
- IsSPICE4, Interactive Mixed Mode Simulator
- AC, DC, transient, distortion, Monte Carlo, noise, optimization, and Fourier analyses.
- Works with all popular schematic entry systems!
- Completely Integrated System with Schematic Entry, Simulation, & Extensive Model Libraries

Windows  
NT - DOS  
Power  
Mac

Call or Write for your **FREE Working SPICE Simulation Kit.**



Ftp Site: <ftp://ftp.iee.ufrgs.br>  
 74774.2023@compuserve.com  
 P.O. Box 710 San Pedro, CA 90733-0710  
 Tel. 31 0-833-071 0, Fax 31 0-833-9658

# SPICE

# CONNECT TIME

The dimensions depend on the size of the magnet.

The pipe section should be just big enough so that the magnet fits inside where the "M" is with the ends of the magnet flush with the outside faces of the end plates. The "W" indicates where the wire will go.

Use the wire to wind about 5,000 turns, more or less, on the cemented-together spool. Stick the magnet in the center of the spool. Solder the ends of the magnet wire to a long piece of coax cable and use some tape to secure the coax to the magnet wire coil. RG-174 is small, easy to handle, and cheap.

Find some sponge rubber somewhere. Cut up an old wet-suit or a mouse pad and make two circles the same size as the end pieces of your spool.

Flatten a "tin" (steel) can to get enough material to make two circles the same size as the rubber circles. Then assemble the hydrophone. Put the rubber circles on both ends of the spool. Put the steel circles on top of the rubber circles. Use some silicone rubber caulking to build up a layer over the whole assembly to make it waterproof.

The steel diaphragms will vibrate when sounds cause increased and decreased pressure in the water. Because they are steel (iron), the magnetic field from the magnet will get stronger or weaker and that will couple into the coil of wire and generate a voltage.

I doubt that you will be able to make a "noise-canceling" hydrophone in the same way that the same thing is done for something like a microphone for aircraft pilots to use. That type of noise canceling picks up the closest sound (the pilot) and reduces sound from farther away (the engines). What you are looking for in noise cancellation with hydrophones is exactly the opposite.

(By the way, I used to ship out aboard the research vessel "Eastward" for Duke University.)

**Msg#: 5651**

**From: Barry Klein To: James Meyer**

Thanks for the suggestion for a design. So then an electret-based mic would not be as good? Also, would adaptive filtering like that which is demonstrated in the Analog Devices EZLAB 2101 DSP kit (to demonstrate suggested use in phone applications) work to eliminate the motor noise? Would there be some way to achieve directionality in a mic design so maybe that approach could work, or else maybe in combination with a more local omnidirectional mic for noise cancellation?

**Msg#: 5823**

**From: James Meyer To: Barry Klein**

Maybe even better. It's just that you can't usually build an electret with stuff found around the average home. I could do it, but then, mine \*isn't\* the average home. 8-)

Many of the hydrophones used for survey work are of the piezoelectric type. Some are tuned to very narrow bandwidths, and some broadband. piezo transducers are available, but you have to know where to look.

Funny you should mention the AD kit. After a long search, I finally got one of their "EZKIT-LITE" evaluation setups. I haven't had time to do more than make sure it worked, though.

If there is enough difference in the frequency spectrum of the signal you'd like to listen to and the noise that you'd like \*not\* to listen to, then yes, filtering will help.

I suspect, though, that the signal and noise are going to be all the same as far as any filter is concerned.

I suspect that concentrating on directionality is the best way to gain some control over noise problems. Any technique that works to make an ordinary microphone directional will also have a similar technique that will work for a hydrophone. After all, the only thing that's different is the medium that you're working with. Air, in the case of ordinary mics, and water, in the case of hydrophones.

I could be more helpful if I knew what you would like to listen to.

**Msg#: 6127**

**From: Barry Klein To: James Meyer**

I was talking with a sales guy at REI (an outdoor sports equipment retailer) who happens to also be a researcher at the Dana Point Oceanographic Institute nearby. They would like to use a hydrophone to listen to whales and dolphins on their excursion trips for the public that they take out of Dana Point. Trouble is that the boat motor noise overshadows any "natural" sounds.

At REI they were selling a hydrophone for \$250 and I got to thinking about using an electret for one and then wondered about solutions for the motor noise problem.

One thing I thought of today is using two in a binaural setup. At least you would then be able to mentally concentrate on whales or whatever and the motor noise could be kind of ignored. Also the "head" could be manipulated to focus in the direction you want.

I spoke with a guy today who is experimenting with spheres with two electrets mounted on them (in "ear" positions) for use to record concerts and ocean waves. He says it works great! So why not have a bunch of them in a switchable array or something that maybe would switch in response to your own head movements or something? Kind of like audio VR...

**Msg#: 6149**

**From: Bob Paddock To: Barry Klein**

Seems the wheel is being reinvented here. Check out the references from a related project I've been working on:



# CONNECTIME

"Role of the Pinna [External Ear/Ear Lobe] in Localization: Theoretical and Physiological Consequences" by Dwight Wayne Batteau. Ciba Foundation Symposium on Hearing Mechanisms in Vertebrates, 1968. pp. 234-239 (Edited by A.V.S. deReuck and Julie Knight. Published by J. & A. Churchill Ltd., 104 Gloucester Place, London, W.I.)

Proceedings of Royal Society, Biology, 1967, 158, 158-180 by Dwight Wayne Batteau.

"Study Molecular Sensation" by Dwight Wayne Batteau and W. M. Hemmes (1966). First Semiannual Report for U.S. Navy Office of Naval Research, Contract 4863(00). NTIS order number: AD-635955 \$17.50 in paper.

"Dwight Wayne Batteau's work On The Significance of Energy Level Transitions in Nerve Function" by Dwight Wayne Batteau and T. B. Eyrick. (1967) Interim Technical Report for U.S. Navy Office of Naval Research, Contract 4863(00). NTIS order number: AD-670614.

"Theories of Sonar Systems and Their Application to Biological Organisms", D.W. Batteau Department of Mechanical Engineering, Tufts University, Medford, MA, Sept. 1966.

"The Neurophysiology of Spatially Oriented Behavior" edited by Sanford J. Freedman. 1968 Dorcy Press IL. Chapter 7 "Listening with the Naked Ear" by Dwight Wayne Batteau.

Phone conversations with Lloyd Mac Gregory Trefethen, Professor of Mechanical Engineering, Tufts University Anderson Hall. He was instrumental in locating some of Batteau's research papers (Dwight Wayne Batteau is now deceased [Died of heart attack while swimming with the Dolphins he was researching]).

"Experiments In Hearing" by Georg von Békésy; translated and edited by E.G. Wever. McGraw-Hill Book Company, Inc. 1960.

National Technical Information Service (NTIS)  
5285 Port Royal Rd.  
Springfield, VA 22161  
(703) 487-4650 Order Menu  
(703) 487-4780 Title Search

We invite you to call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (860) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps.

## ARTICLE SOFTWARE

Software for the articles in this and past issues of *Circuit Cellar INK* may be downloaded from the Circuit Cellar BBS free of charge. It is also available on the Internet at <ftp://ftp.circellar.com/pub/circellar/>. Web users should point their browser at <http://www.circellar.com/>. For those with just E-mail access, send a message to [info@circellar.com](mailto:info@circellar.com) to find out how to request files through E-mail.

For those unable to download files, the software is also available on one 360 KB IBM PC-format disk for only \$12. To order Software on Disk, send check or money order to: Circuit Cellar INK, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (860) 875-2199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

431 Very Useful      432 Moderately Useful      433 Not Useful

STATEMENT REQUIRED BY THE ACT OF AUGUST 12, 1970, TITLE 39, UNITED STATES CODE SHOWING THE OWNERSHIP, MANAGEMENT AND CIRCULATION OF CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL. published monthly at 4 Park Street, Vernon, CT 06066. Annual subscription price is \$21.95. The names and addresses of the Publisher, Editorial Director, and Editor-in-Chief are: Publisher, Daniel Rodrigues, 4 Park Street, Vernon, CT 06066; Editorial Director, Steven Ciarcia, 4 Park Street, Vernon, CT 06066; Editor-in-Chief, Kenneth Davidson, 4 Park Street, Vernon, CT 06066. The owner is: Circuit Cellar, Inc., Vernon, CT 06066. The names and addresses of stockholders holding one percent or more of the total amount of stock are: Steven Ciarcia, 4 Park Street, Vernon, CT 06066. The average number of copies of each issue during the preceding twelve months are: A) Total number of copies printed (net press run) 35,354; B) Paid Circulation (1) Sales through dealers and carriers, street vendors and counter sales : 4,896, (2) Mail subscriptions: 22,966; C) Total paid circulation: 27,862; D) Free distribution by mail (samples, complimentary and other free): 4,068; E) Free distribution outside the mail (carrier, or other means): 83; F) Total free distribution: 4,151; G) Total Distribution: 32,013; H) Copies not distributed: (I) Office use leftover, unaccounted, spoiled after printing: 527; (2) Returns from News Agents: 2,814; I) Total: 35,354. Percent paid and/or requested circulation: 87.0%. Actual number of copies of the single issue published nearest to filing date are: (November 1995, Issue #64) A) Total number of copies printed (net press run) 33,500; B) Paid Circulation (1) Sales through dealers and carriers, street vendors and counter sales : 4,583. (2) Mail subscriptions: 22,174; C) Total paid circulation: 26,757; D) Free distribution by mail (samples, complimentary and other free): 3,179; E) Free distribution outside the mail (carrier, or other means): 0; F) Total free distribution: 3,179; G) Total Distribution: 29,936; H) Copies not distributed: (1) Office use leftover, unaccounted, spoiled after printing: 300; (2) Returns from News Agents: 3,264; I) Total: 32,500. Percent paid and/or requested circulation: 89.4%. I certify that the statements made by me above are correct and complete. Daniel J. Rodrigues, Publisher.

# STEVE'S OWN INK

The Powers that Be



f

or centuries, nations have been beating their plowshares into swords and vice versa. Witness the change in the Soviet Union. Once the world's largest nation, it now stands as a conglomeration of independent countries. No longer can it command the authority and fear of the cold-war years.

And Canada, now the world's largest nation and the U.S.'s largest trading partner, is having problems. While I write this editorial, the residents of Quebec are going to the polls to decide whether or not they wish to remain Canadian citizens. It appears that even first-world nations cannot escape the massive political and economic upheavals that characterize the third world.

The world of business is not so very different. Yesterday's king of the heap is not necessarily tomorrow's Witness the rise and fall of companies such as Wang, Burroughs, Data General, Prime. And, then there are the companies like IBM and Digital Equipment that rose, dwindled, and are currently attempting to make a comeback.

And, although one would think right now that the eventual fate of the embedded PC is highly determined by the success of Intel and Microsoft, pause a moment. The 'x86 chip is here, manufactured not only by no-name fly-by-nighters, but also by Advanced Micro Devices, Cyrix, and National. Would the demise of Intel, something hard to imagine at this point, stop 'x86 production? I doubt it very much.

The software end isn't much different. Microsoft's DOS is most certainly king in the embedded PC world. However, the craving for fancy graphical interfaces is resulting in many variations of stripped-down, Windows or Windows-like operating systems. No doubt, the need for preemptive multitasking and real-time operations in the embedded world will continue the drive for better, perhaps non-Microsoft, operating systems.

At present, the embedded PC market looks good. Companies needing to solve more complex problems and shorter development time are flocking to the 'x86 architecture. Specialization is occurring at every level. PC boards are specializing in GPS, analog-to-digital and digital-to-analog conversion, frame grabbing, and the list goes on. Companies simply do not have the time to provide these sophisticated solutions in-house. Specific solutions are becoming a mixture of out-sourced and off-the-shelf solutions. The real art is knowing how to mix and mesh specific features for a client.

And, that's where we come in with Circuit Cellar *INK's Embedded PC*. We aim to keep you in touch with the embedded PC industry's pulse. We want to introduce you to new hardware and software and then show you the magic mixes that enable you to solve those difficult client problems. We want to provide you contacts through editorial, advertisements, references, and sources.

How committed are we to helping you and your company? Our initial commitment to quarterly inserts in 1995 has already evolved to a commitment to bimonthly inserts in 1996. We continue to support our core **commitments**, but with sufficient enthusiasm from readers, who knows how far we can expand.

Tomorrow's king of the embedded PC heap-who knows? But, the fight over whether it's here to stay or not seems over. The embedded PC is here to stay-at least for a while.



steve.ciarcia@circellar.com