

CIRCUIT CELLAR[®] INK[®]

THE COMPUTER APPLICATIONS JOURNAL

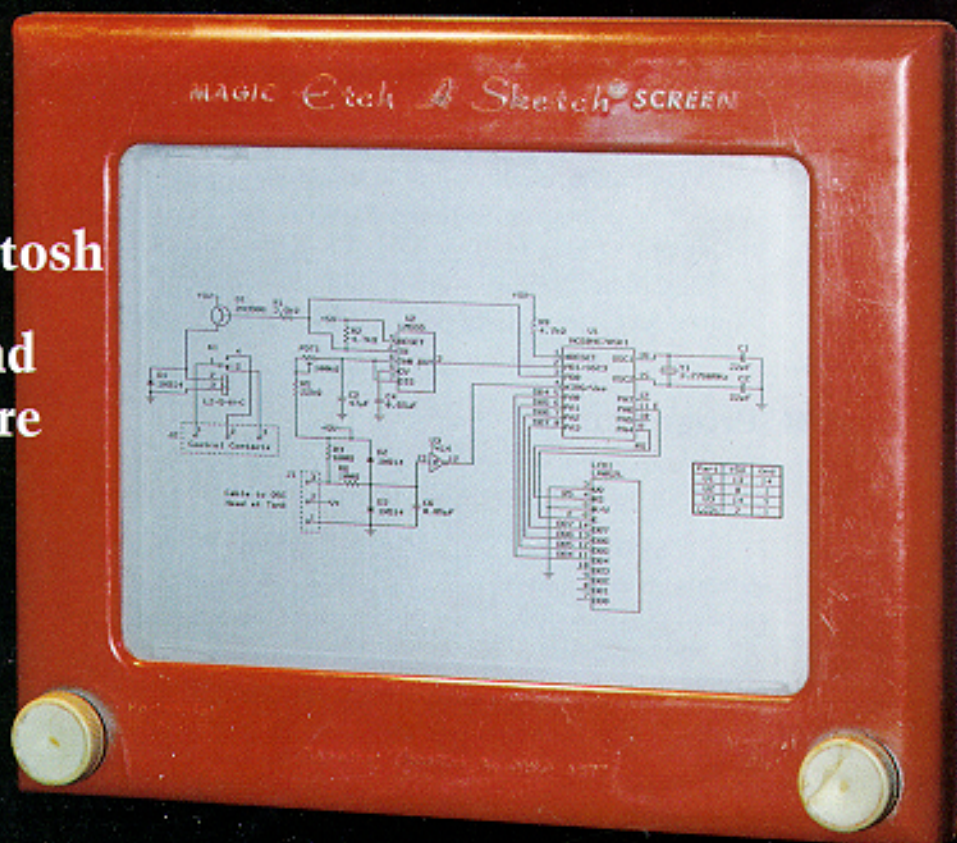
#70 MAY 1996

CROSS-DEVELOPMENT TOOLS

Digital
Storage Scope
Using a Macintosh

PCB Router and
Layout Software
Evaluation

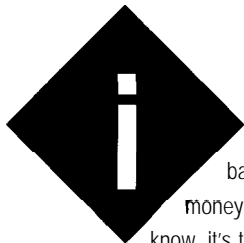
Revealing
the Mysteries
of PC Cache



\$3.95 U.S.
\$4.95 Canada

TASK MANAGER

'Round and 'Round
We Go



was recently at a seminar where we were bandying about the old adage, "You have to spend money to make money," when someone quipped, "You know, it's the circle of life."

My first thought was, "Someone has been watching too much Lion King." Then I realized that perhaps the someone was me. Any of you with small children and a VCR at home likely know what I'm talking about.

When it comes to development tools, I've also come full circle. For years, much of my own development work has been for the **Zilog Z180** processor. Back when I started with that chip, I did all my programming on a **Z180** machine running **CP/M**. Using native assemblers and compilers, I was able to do some testing of the code right on the development machine, while more extensive testing was easily done on the (very similar) target hardware.

When PCs became more prevalent and powerful, I switched to cross-development tools to program the same **Z180**. Gone were the days of running the code on the development system, but there were other advantages to the new platform.

Now that embedded PCs are becoming all the rage, we're back to running native development tools and doing some testing on the development platform.

The embedded PC isn't yet king, so there is certainly still a place for traditional cross-development tools. For example, in our first feature article, Francis Deck describes some simple circuitry for turning a desktop Macintosh into a digital sampling (or storage) oscilloscope. Having a static trace of a one-time dynamic event can be invaluable for tracking down intermittent problems.

Next, William Rogers surveys some of the popular PC board layout and routing packages on the market, and presents his test results and opinions. For such a well-defined task, many of the tools still aren't up to the challenge,

Following up on some of his past articles, Professor Mike Smith next describes his latest adventure in trying to legally procure cross-development tools for his classroom on a university budget. It turns out to have a happy ending.

Our last two feature articles are conclusions to articles that started last month. The first looks at some software tools for the **MicroSPARCIIe** processor, while the second finishes up our treatment of the **CAN** and **J1850** intravehicle protocols.

In our columns, Ed continues exploring memory cache, Jeff looks at several popular prototyping methods, and Tom finishes up with the latest contender on the **PID-Pong** challenge.

editor@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

MANAGING EDITOR
Janice Marinelli

CIRCULATION MANAGER
Rose Mansella

TECHNICAL EDITOR
Elizabeth Laurençot

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 0696-6965) 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 698, Holmes, PA 19043.9613 or call (600) 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) 875-2199.

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES




NORTHEAST & MID-ATLANTIC	SOUTHEAST	MIDWEST	WEST COAST
Barbara Best (908) 741.7744 Fax: (908) 741-6823	Christa Collins (305) 966-3939 Fax: (305) 985-8457	Nanette Traetow (706) 357-0010 Fax: (706) 357-0452	Barbara Jones & Shelley Rainey (714) 540-3554 Fax: (714) 540-7103

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

All programs and schematics in *Circuit Cellar INK®* have been carefully reviewed to ensure their performance 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 © 1996 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 4** A Simple DSO Circuit for the Macintosh
by *Francis Deck*
- 2 0** Autorouter and Board Layout Software Tool Analysis
by *William Rogers*
- 3 0** The Evaluation Board Saga Continues
Low-Cost Educational Tools
by *Mike Smith*
- 3 6** The Embedded Sun
Part 2: Exploiting the Microkernel
by *Doron Hendel & Rowena Turner*
- 4 2** Vehicular Control Multiplexing with CAN and J1850
Part 2: Application to Motorola Embedded Controllers
by *Willard Dickerson*
- 5 2**  Firmware Furnace
80x86 Performance
Probing the Cache
Ed Nisley
- 6 2**  From the Bench
Handcrafting Design Ideas
Jeff Bachiochi
- 6 8**  Silicon Update
Fuzzy PID-Pong
The Final Chapter
Tom Cantrell

INSIDE ISSUE 70

- 2** Task Manager
Ken Davidson
- 'Round and 'Round
We Go
- 6** Reader I/O
Letters to the Editor
- 8** New Product News
edited by Harv Weiner

- 81** **ConnectTime**
Excerpts from
the Circuit Cellar BBS
conducted by
Ken Davidson
- 96** **Priority Interrupt**
An Inventing Experience
- 65** **Advertiser's Index**

READER I/O

DEVOTED FAN NEEDS SUPPORT

I just received my April *INK*. Much to my surprise on page 19, I saw a schematic nearly identical to the Caller ID kit we've been selling since June 1995 and that was featured in *Nuts and Volts* November 1995.

Although Richard Newman claims "there is not a single inexpensive Caller ID interface for the PC," our unit sells for \$39 and includes all the parts, a double-sided PCB, software for DOS and Windows, and source code in Visual Basic and C. I'm not sure how such an oversight could have taken place, particularly since we always submit our new product releases to your magazine.

Have a look at our web page (<http://www.itutech.com>). There, you'll see some of the products relating to Caller ID and PIC programming that you have missed informing your readers about.

Please keep the sharp technical focus you have had over the years in *INK*. Don't change it into a boring, fluff-oriented publication like so many of the trade magazines in this field.

Chris B. Sakkas
chris@itutech.com

Of the hundreds of newproduct news releases INK receives per month, only 9-10 of them are selected for publication. I'm sure you understand that there's no way to track such a high volume of releases without incurring significant costs.-Editor

YOU'RE RIGHT-THERE IS A BETTER WAY

Steve's "The Old Curmudgeon" (*INK 68*) unveils the surface of perhaps the biggest revolution in humanity's development.

Electronic computers are the best tools ever developed for data handling, math computations, graphic rendering, and communications. Such devices are not static or fully developed, but instead are being improved at an exponential rate.

Naturally, we have to be conscious of the effect on both sides of our social environment. Scientific and technical development provides the means for humanity to undertake space-age projects, modern aircraft, weather forecasting, genetics, medical and biological research, and so on. It brings the big dollars that keep society and development going.

On the other side, too many software products have been forcefully and artificially rushed into the market. To make "full use" of these commodities, consumers make prompt acquisitions of newly developed equipment for trivial tasks. The consumer gets locked in the bigger, better trend.

Undoubtedly, we have marvelous equipment, but we've only come half way. Our technical society still has a lot of brainwork to do to develop software products that solve many practical problems not yet faced. To take advantage of the data-handling capabilities available, they may need to request equipment configurations based on their specific problems.

What we really need is resource optimization. Keeping in mind costs, volume production, balance, standardization, and so on, we need to design equipment appropriately balanced for home, commercial, technical, and artistic applications.

I'd also like to take this opportunity to thank you for your great *Circuit Cellar INK*. You provide the technical electronics and computer community with a lot of practical knowledge.

Joaquin M. Gonzalez
Mexico City, Mexico

Contacting Circuit Cellar

We at Circuit Cellar *INK* encourage communication between our readers and our staff, so we 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) 8752199.

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. Send new subscription orders, renewals, and address changes to 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.

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

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

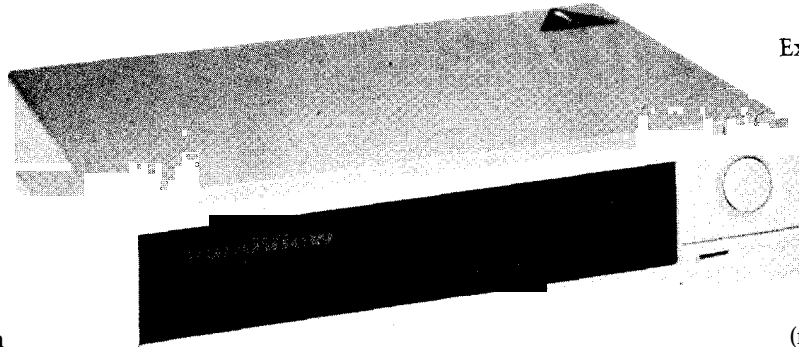
NEW PRODUCT NEWS

Edited by Harv Weiner

MULTIPLE VIDEO INTERFACE

Belkin introduces **ExpandView**, a video broadcast box that enables users to send picture-perfect video output to as many as eight separate monitors. Most video switch boxes and Y-cables degrade the quality of computer video signals, causing flicker and color shifts when connected to multiple monitors. ExpandView's advanced gain controls automatically amplify weak signals, eliminating flicker and color variations, even when broadcast to as many as eight monitors simultaneously.

ExpandView's signal processing enables monitors to be placed at distances up to 200' from the source computer. In addition, multiple units may be cascaded so images can be displayed on a virtually unlimited number of monitors.



ExpandView is available in two models, supporting either four or eight monitors, using high-density DA-151/O ports.

At a suggested list price of \$159

(four monitors) and

\$239 (eight monitors), the device is

compatible with all standard VGA and SVGA analog RGB video cards. Users connect both the computer and monitors to ExpandView with standard monitor extension cables (not included). The device comes with its own power supply and carries a one-year warranty.

Belkin Components

1303 Walnut Park Way • Compton, CA 90220
(310) 898-1100 • Fax: (310) 898-1111

#500

AC/DC VOLTAGE REFERENCE

Designed around a Thaler Corporation (Tucson, AZ) VRE305 +5-V DC reference IC, the TDL304 and 304X Voltage Reference provide outputs of ± 10 VDC, a

10-V p-p square wave at 1000 Hz, and a 10-V RMS 1000-Hz sine wave, all to 0.1% accuracy after a 30-minute warm-up. Long-term stability is within 1 mV per 1000 hours. The 1000-Hz

frequency is crystal controlled to $\pm 0.01\%$.

The Model 304 accommodates an internal voltage divider such as an ESI model DP211 (not supplied) as shown in the accompanying

photo. The Model 304X is designed to use an external voltage divider such as a General Radio Co. Model 1454-A or 1455-A (not supplied).

The unit is housed in an aluminum cabinet that measures 6" x 7.5" x 4". The front panel is screened in black epoxy ink. Input power is 115 VAC, 50-60 Hz at 10 W.

The units are available as a kit (voltage divider not included) for \$149.95. An assembled-and-tested unit sells for \$189.95 (without voltage divider).

TDL Electronics, Inc.
5260 Cochise Trail
Las Cruces, NM 88012
(505) 382-8175
Fax: (505) 382-8810 #501



NEW PRODUCT NEWS

VIDEO LINE DOUBLER

Genesis Microchip introduces **DICE**, a Video Line Doubler (VLD) family of chips. DICE, a contraction of deinterlacing, is a line-doubling technology that converts interlaced video for display on noninterlaced systems. Available in both 8- and 10-bit versions (**gmVLD8** and **gmVLD10**), DICE is a single-chip, cost-effective solution for performing high-quality temporal and vertical video and image processing.

Unlike other deinterlacing methods, which merge video fields or double interlaced odd and even lines, the Genesis chip uniquely blends vertical and temporal filtering to remove the visual artifacts commonly associated with inferior deinterlacing techniques. This blend enhances video quality and increases the video's effective vertical and temporal resolution.

Equipment benefitting from this technology includes large-screen televisions, video walls, projection systems, video-in-a-window workstations, as well as a growing number of consumer-level products, such as home-theater screens.

Volume pricing for the gmVLD8 is \$30 and \$55 for the gmVLD10.



Genesis Microchip, Inc.

200 Town Centre Blvd., Ste. 400 • Markham, ON • Canada L3R8G5 • (905) 470-2742 • Fax: (905) 470-2447

#502

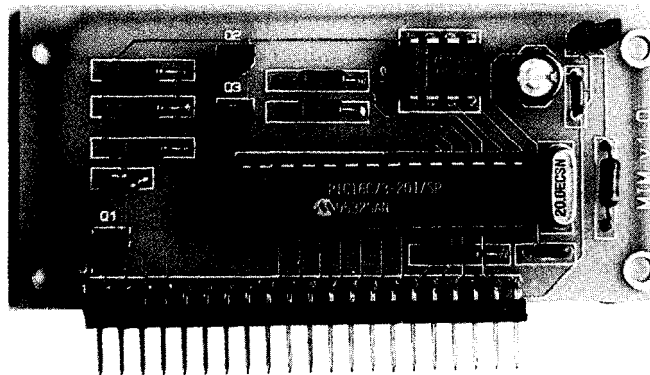
PACKET-RADIO TELEMETRY UNIT

The **MIM** module is a miniature APRS-compatible packet-radio telemetry unit. It sends APRS GPS position reports, analog and binary telemetry data in packet form (APRS compatible), a beacon text message, and CW Morse ID when attached to a suitable FM transmitter. Each format can be sent at user-selected time intervals or bypassed. A single digipeater path can be designated for packet transmissions.

The unit accepts GPS data direct from GPS receivers in NMEA 0183 (RMS) format for position reporting and can telemeter five A/D converter inputs (8-bit resolution and 8 bits of parallel digital data). Through PC-configurable software, the user customizes call signs and other packet transmission parameters by temporarily attaching the module to a PC serial port.

An onboard EEPROM stores configuration information indefinitely. Configuration software can be rerun as needed to change settings for different applications. All connections are through a 21-pin SIP connector.

An onboard voltage regulator accepts power from an unregulated 8-40-VDC source, and the board may also be powered from a regulated 5-VDC source. The module requires only 15 mA operating current, making it ideal



for low-power applications. It produces standard tones and transmitter keying signals that can be connected directly to most transmitters and handi-talkie equipment.

The MIM module measures 1.5" x 3.25" with a 21-pin, single row, right-angle header along one length. MIM modules sell for \$89.95 plus shipping and handling.

Clement Engineering, Inc.

P.O. Box 1086 • Severna Park, MD 21146

(410) 268-6736 • Fax: (410) 268-4612

wiclement@aol.com

#503

NEW PRODUCT NEWS

LOW-COST IN-CIRCUIT EMULATOR

Zilog announces an in-circuit emulator for users of the Z 180 architecture that was designed for Z 180-based design development and debugging. The **Z8S180** provides serial communications for applications such as communication ports, modems, printers, WAN cards, PC I/O cards, and wireless LANs.

The Z8S180 emulator combines the benefits of an evaluation board and many high-end emulation capabilities. It does not require a target board for initial debugging and development. The software provided includes a debug monitor that enables users to download programs and run with or without breakpoints. For larger applications and advanced development, it can plug directly into the Z8S180 socket of the user's target board via an emulation adapter pod.

For real-time, in-circuit emulation, the Z8S 180 emulator plugs directly into the microprocessor socket of

your design. Application-specific system-level debugging enables faster design implementation because there is no need to write special software. The emulator features an interactive real-time environment for emulation and debugging.

The evaluation board contains an integrated microprocessor, a monitor program, and onboard memory. A suite of software support tools includes an assembler/linker, a C compiler, and a simulator/debugger to generate and debug code. Software can be assembled or compiled on a PC host.

The Z8S180 sells for under \$200.

Zilog, Inc.

210 East Hacienda Ave. • Campbell, CA 95008-6600

(408) 370-8000 • Fax: (408) 370-8056

<http://www.zilog.com/>

#504

INFRARED TRANSCEIVER MODULE

The TFDSG000 is a multimode integrated infrared transceiver module for data communication systems. It supports all IrDA (Infrared Data Association) speeds up to 4 Mbps, HP-SIR, and Sharp ASK modes.

The TFDS6000 contains the optical components and mixed-signal

control chip necessary for an optimized, high-speed IrDA implementation.

Transmit and receive functions are accomplished by using proven emitter and detector technology.

By integrating the preamplifier of the receiver and the driver stage of the transmitter, the TFDSG000 combines the functions of two

ICs and eliminates a large number of discrete components. A current-limiting resistor and bypass capacitor are the only external components needed for a complete transceiver.

The highly efficient device operates at 5 V with standby power consumption at 30 mW to prolong battery life. The transceiver is

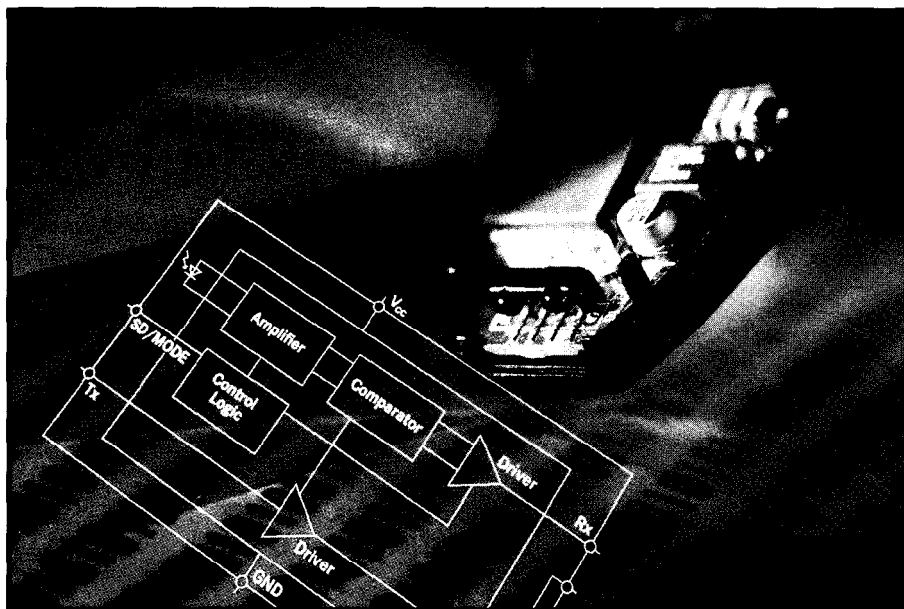
housed in an 8-pin epoxy resin SMP and measures only 13 mm x 5.3 mm x 5.6 mm.

Devices complying with the IrDA Standard 1.1 must be able to send and receive data over distances of up to 3 m and be effective at up to 15" off the mechanical axis. The TFDSG000 meets the specification while providing an automatic gain function to ensure maximum sensitivity, even with a high level of optical interference.

The TFDSG000 sells for \$7.25 in quantity.

Temic Semiconductors
P.O. Box 54951
Santa Clara, CA 95056
(408) 567-8220
Fax: (408) 567-8995

#505



NEW PRODUCT NEWS

DSP BOARD

Sonitech announces the **SPiRiT-40 PCI**, a universal acquisition and processing platform for the high-performance compute-intensive applications of communications, multimedia, image processing, and real-time high-speed data processing. The **SPiRiT-40 PCI** provides complete PCI bus mastership and supports PCI block transfers, which are necessary to achieve high-speed PCI-bus data transfers.

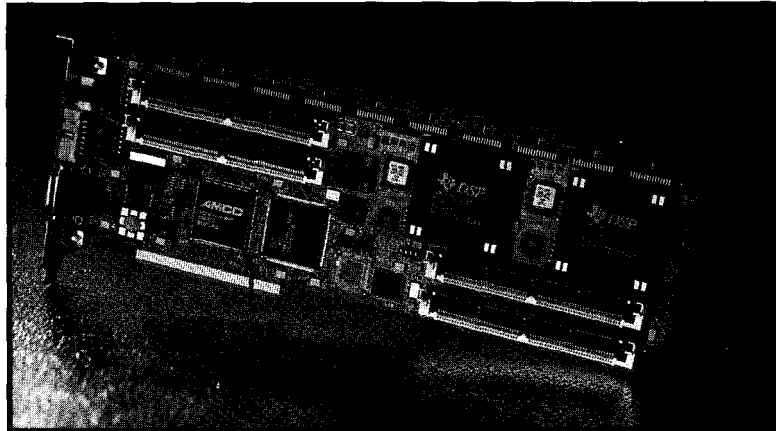
Two TMS320C40 32-bit DSP chips from Texas Instruments operating at 50 or 60 MHz form the core of the unit. Each C40 provides six bidirectional buffered 20-MBps comm ports for I/O.

Due to a limited number of available PCI slots in today's computers, **SPiRiT-40 PCI** has been integrated with a wide

range of acquisition and processing boards, located in either ISA slots or external boxes. Such boards include an array of ADC and DAC boards linked by comm port connections and a frame grabber board for image processing.

SPiRiT-40 PCI is available with up to 16 MB of zero-wait-state SRAM. Each C40 has a private local memory with a maximum size of 4 MB and both C40s share a global memory bus, which can be as large as 8 MB.

Development tools include a DSP compiler, assembler, and linker for C40s; a debugger; and hundreds of highly optimized DSP math, image-process-



ing, and spectral-analysis library routines. In addition, the **SPiRiT-40 PCI** Run-Time Library, which hosts **SPiRiT-40** communications, contains several example routines and source code, which can be modified and customized for any application.

The **SPiRiT-40 PCI** with two 50-MHz C40s and 768 KB of zero-wait-state SRAM sells for \$6995. A development system which includes a C compiler for

C40s, a debugger, and the Run-Time Library software is \$9785.

Sonitech International
14 Mica Ln.
Wellesley, MA 02181
(617) 235-6824
Fax: (617) 235-2531
info@sonitech.com

#506

RS-232 TO RS-422/485 CONVERTERS

B&B has introduced four port-powered RS-232-to-RS-422/485 converters, each packaged in a compact DE-9 hood. The **422LP9R** and **485LP9R** convert the RS-232 TD and RD lines to balanced RS-422/485 signals. The pinouts of these units match a SMPTE-compatible (video standard) controlling device.

All of the converters are powered from the RS-232 DTR and RTS handshake

lines (one must be high). The RS-422/485 driver is enabled and disabled with RTS. The receiver is constantly enabled on RS-422 units. On RS-485 units, the receiver is disabled when the driver is enabled and is enabled when the driver is disabled.

All of the converters have a female DE-9 connector on the RS-232 side. The **422LP9R** and **485LP9R** have a female DE-9 connector on the RS-422/485 side. The **422LP9TB** and **485LP9TB** have terminal blocks on the RS-422/485 side and can be externally powered with a 12-VDC supply. All units are priced at \$59.95 each.

B&B Electronics Manufacturing Co.
707 Dayton Rd.

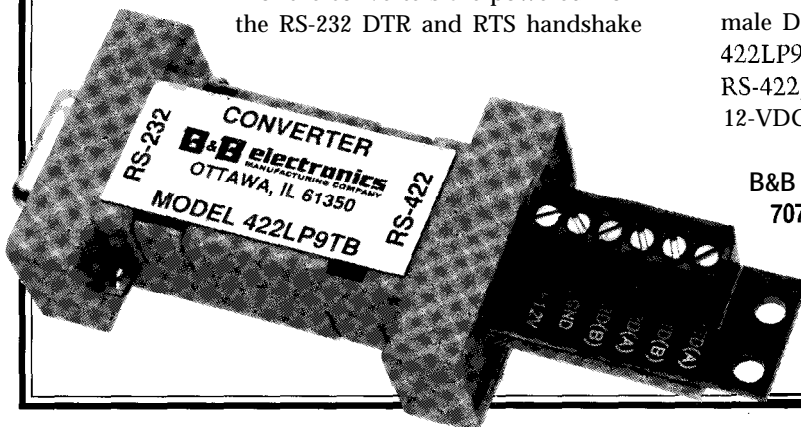
Ottawa, IL 61350

(815) 433-5100

Fax: (815) 434-7094

<http://www.bb-elec.com/>

#507



NEW PRODUCT NEWS

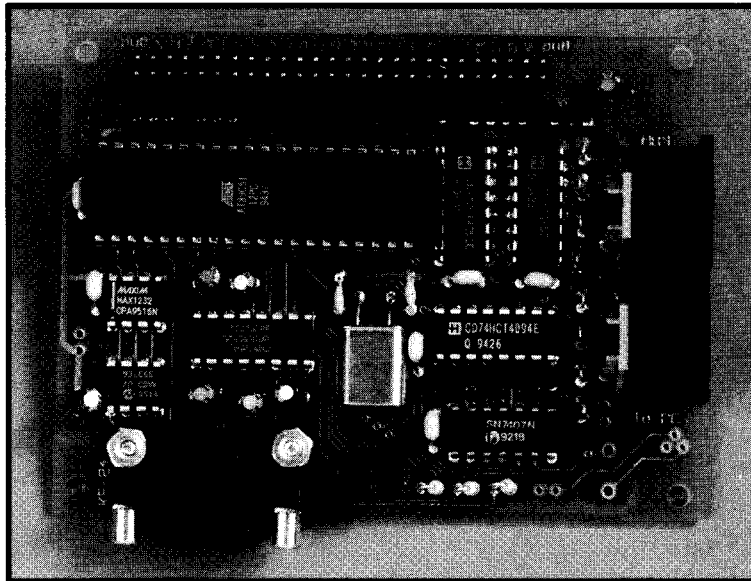
KEYBOARD ENCODER

The KE24 from Hagstrom Electronics interfaces keypads or switches to a PC/XT, AT, PS/2, or compatible keyboard input. It provides all necessary communication to the PC and can operate with or without the keyboard attached.

The 24 I/O lines of the encoder may be configured as either a matrix of rows and columns (up to 12 x 12) or as individual inputs. Each input may be programmed to emulate the standard keys found on a 101-key keyboard or to output a macro of up to 16 key-strokes.

The unit also features an RS-232 port for conversion of the received characters into PC-compatible keystrokes. Baud rates are selectable from 300 to 19200 bps. The KE24 can operate alone, or an external PC keyboard may also be connected and used simultaneously with the KE24.

The KE24 Keyboard Encoder sells for \$99.95 plus shipping and handling.



Hagstrom Electronics
2 Green Lantern Blvd.
Endicott, NY 13760
(607) 786-7523

#508

B.G. MICRO, INC.

P.O. Box 280298
Dallas, Tx 75228
Orders Only 1-800-276-2206
Tech Support 214-271-9834
Fax 214-271-2462
Local Orders 214-271-5546

Our Internet address is

BGMICRO@IX.NETCOM.COM



ANOTHER LCD NETWORK CARDS

ETHERNET CARDS
These 1x16 LCDs were pulled from medical equipment. The Dim. are 3"x1-1/8" x3/16". Char.Height .26". They all come in the plastic bezel. Easily removed -6 screws. 8 Bit ASCII in-put.... **\$3.74 or 10/\$35.00**

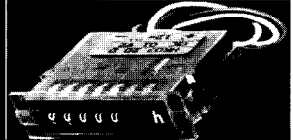
16 Bit Combo NE2000.....
FEATURES: 16TB Model provides UTP/BNC ports to comply with IEEE 802.3 10Base 2/T standard. Supports Novell's NE2000 compatible and enhanced mode. Fully software configurable on the selections of connector types, interrupt, I/O base address, and PROM address with no jumper setting. Auto-detection and correction of a polarity reversal problem on the unshielded twisted pair (UTP) wire. Full LED indicators for Link, TX/RX, Polarity Reversal & collision status. **\$27.95**

TIME OUT

Miniature Hour Meter
Brand New!! Super low power. Hours and 1/100th resolution. Use this handy gadget for printed circuit board warranty, or anything where low power consumption is required. (solar panels, etc.) The hour meter uses a quartz controlled oscillator that generates an impulse every 36 seconds, accurate within .005%. 12VDC operation-Clear plastic housing-Flush mount w/clip-stop device(snap-in). Measures 2-1/4"x1-1/4"x2" Original cost --- \$40.00 ea. Our price **\$7.95**

THE 66 MHZ CPU

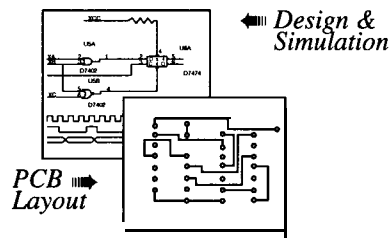
80486SX-2 66MHZ CPU chip. Keep reading, there is not that much difference. This CPU chip comes in a PGA package and operates on 5 Volts. These chips are brand new factory package by AMD. Replace that slow, tired, old CPU with this new, fast, 66MHZ "speed demon". The only difference between an SX and a DX is that the DX has a math coprocessor. This means you can replace a slower 80486SX-xx or a slower 80486DX-xx with this "screamer". While they last. **\$17.50 or 2/\$30.00**



TERMS: (Unless specified elsewhere) Add \$5.00 postage, we pay balance on Orders Under 5 lbs. Orders over \$50.00 add .85 for insurance. No C.O.D. Texas Residents add 8 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 or write for a copy of our free catalog

Low Cost CAD Software for the IBM PC and Compatibles Now In Windows™ 95



- Easy to use schematic entry program (**SuperCAD**) for circuit diagrams, only \$149. Includes **netlisting**, bill of materials, extensive parts libraries.
- Powerful, event-driven digital simulator (**SuperSIM**) allows you to check logic circuitry quickly before actually wiring it up. Works directly within the **SuperCAD** editor from a **pull-down** menu and displays results in "logic analyzer" display window. Starting at \$149 this is the lowest cost simulator on the market. Library parts include TTL, and CMOS devices.
- Analog simulator (**mentalSPICE**) for \$149. Allows AC, DC and transient circuit analysis. Includes models of transistors, **discretes**, and op amps.
- Circuit board artwork editor and autorouter programs (**SuperPCB**), starting at \$149. Produce high quality artwork directly on dot matrix or laser printers. You can do boards up to 16 layers including surface mount. Includes Gerber and Excellon file output. Autorouter accepts **netlists** and placement data directly from the **SuperCAD** schematic editor.
- Low cost combination packages with schematics and PCB design: 2-layer for \$399, 16-layer for \$649.

Write or call for free demo disks:
MENTAL AUTOMATION
5415 - 136th Place S.E.
Bellevue, WA 98006
(206) 641-2141. BBS (206) 641-2846
Internet: <http://www.mental.com>

FEATURES

14

A Simple DSO Circuit for the Macintosh

20

Autorouter and Board Layout Software Tool Analysis

30

The Evaluation Board Saga Continues

36

The Embedded Sun

42

Vehicular Control Multiplexing with CAN and J1850

A Simple DSO Circuit for the Macintosh

Designing a digital sampling oscilloscope for a Mac can be a nightmare. There's no inexpensive MCU fast enough. So, Francis designed his own interface using some clever money-saving techniques.

FEATURE ARTICLE

Francis Deck



When I tried to design a digital sampling oscilloscope (DSO) circuit for my

Mac, I ran into a long-standing problem in microcomputer interfacing—the lack of a satisfactory, inexpensive microcontroller (MCU) fast enough for the task.

A survey of 8-bit flash ADC chips suggested that a sampling rate of 15-20 MHz was a feasible design goal. Notably, low-end commercial DSO units provide this sampling rate.

I ruled out a bus-interfaced design because it would tie me to a single hardware platform. A minimal device might use an MCU to control the ADC, collecting a “sweep” of data at high speed and then transmitting that data to the Mac over a standard serial line.

Unfortunately, such a simple task is outside the speed range of common MCU chips, such as the popular and well-supported 8051 and PIC families. These chips have impressive clock speeds—units up to 20 MHz are readily available—but instructions take several clock cycles to execute.

As well, a firmware loop to read an ADC and store the result in an array requires several instructions. A simple circuit, in which an ADC interfaces to an MCU directly, is probably limited to sampling rates on the order of 1 MHz.

In large-scale microprocessor systems (e.g., personal computers), direct memory access (DMA) overcomes this hurdle. During high-speed data transfers, the CPU is essentially idle, and the input/output device “talks” directly to system memory.

But, DMA requires either extensive memory management facilities or an I/O device that is nearly as smart as the CPU itself. The increased chip count, however, spoils the attractiveness of an inexpensive MCU.

I found that a FIFO memory chip also serves as a single-chip alternative to DMA for virtually any MCU family. Unlike a conventional RAM chip, the FIFO manages the address lines of an internal RAM and has an asynchronous dual-port structure. You can therefore avoid any arbitration or "glue" between the I/O device and the MCU. Thus, a FIFO seems ideal for high-speed collection of sequential data such as a digitized analog signal.

The device is a DSO circuit with 8-bit resolution and 15 sampling rates ranging from 900 Hz to 14.7 MHz. Component cost is roughly \$120, and all of the parts are readily available. An unpretentious analog preamp is compatible with a standard scope probe and provides a $\pm 2.5\text{-V}$ input range.

The intended host computer for the circuit is a Macintosh, but a baud-rate constant can be changed in the firmware to suit the slower RS-232 port of a PC-compatible. The circuit can also be used as an 8-channel logic analyzer simply by replacing the ADC chip with an 8-input logic buffer.

Be cautious, however. This circuit is not suitable for testing line-powered devices since it contains none of the safety features built into commercial test equipment. Its use should be limited to experimentation with circuits that are powered by fully-enclosed low-voltage supplies.

CIRCUIT DESCRIPTION

The main part of the DSO circuit is shown in Figure 1. I chose a PIC 16C84 microcontroller for U6 because it stores its firmware on an EEPROM rather than an EPROM. It's nice to avoid the UV erasure stage during debugging.

The clock oscillator (U1) drives a 74HC4040 counter (U2) to provide five different sampling frequencies. The 7.3728-MHz output of U2 is a popular multiple of standard baud rates and provides the clock signal for the MCU.

An external clock requires setting the Power Up Timer fuse on the MCU during programming. The timer provides a delay so the MCU won't start up until the external clock stabilizes.

The National Semiconductor DS-8921 RS-422 transceiver (U7) drives the serial line to the Mac, using a standard printer cable. For RS-232 operation, the Rx_D- and Tx_D- lines provide the correct signal polarities, but a

proper RS-232 driver chip such as Maxim's MAX232 would be preferable.

The MCU uses a 74HC 15 1 S-input multiplexer (U3) to select one of the five hardware-generated sampling frequencies or to disable the sampling clock. A Harris CA33 18 flash ADC with a maximum sampling rate of 15 MHz was chosen for its availability.

As is typical of CMOS ADC chips, the CA3318 has an on-chip sample-and-hold which eventually "drips," so the sampling clock must be a negative-going pulse train with a maximum pulse width of ~ 500 ns. Square waves above ~ 1 MHz satisfy this criterion, but lower sampling frequencies require an asymmetric clock.

At clock frequencies below ~ 370 kHz, the MCU is fast enough to generate sampling clock pulses in firmware while simultaneously waiting for a transition in the *Full flag of the FIFO. An input of U3 is wired to the I/O line RB1 of the MCU for this purpose.

The eight logic outputs of the ADC are connected to the input lines of U5, a Dallas Semiconductor DS2013 FIFO with a capacity of 8 KB. Each rising edge from the SCLK signal causes a data byte to be written to the FIFO. The FIFO is implemented as a variable-length array, and data is read out of it in the same order as it is written.

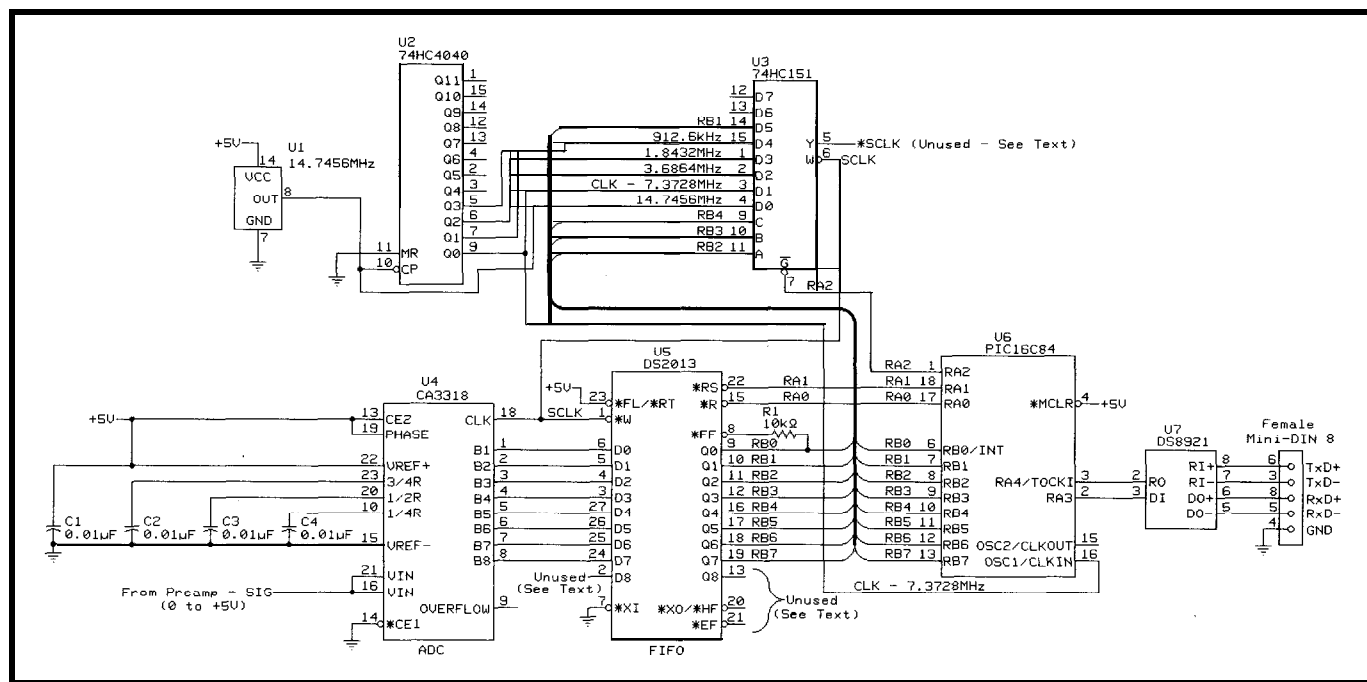


Figure 1—The main section of the DSO circuit requires only six chips and an oscillator. The combination of an ADC with a FIFO memory chip permits data acquisition in 8-KB bursts at sampling rates that exceed the clock speed for the microcontroller.

Each low-to-high transition on the . Write line causes a new data byte to be stored in the array. When the *Read line is pulled low, the next available data byte appears on the FIFO's output lines. When the *Read line is high, the output lines of the FIFO are high impedance, so the MCU can read the aptly named *Full status line. The additional *Half-Full and *Empty lines as well as data lines for a ninth bit are unused but are on the schematic.

Since there are no address lines, bus buffers, arbitration circuitry, or handshake protocols to worry about, this must be the simplest form of DMA. The MCU can erase the FIFO by lowering the *Reset line, but this must be done with both the *Read and *Write lines in a high state. The sampling clock must occasionally be disabled via the *EN line of U3.

The MCU erases the FIFO prior to collecting a "sweep" of data, and then it reenables the clock. A high-to-low transition on the *Full line of the FIFO indicates to the MCU that data is available for transmission to the Mac.

Some care must be taken when using other peripherals instead of the CA3318 ADC. The FIFO requires stable data on its inputs during the rising edge of the SCLK pulse. The Phase setting on the CA3318 selects the correct clock phase for this application. Other peripherals may require an inverted SCLK signal, which is available from U3, as shown.

Most flash ADC chips have an inherent delay of two clock cycles due to internal data-holding registers. One doesn't know exactly when the SCLK signal starts up after being enabled, so we must assume the first clock pulse to the ADC might be a "runt" pulse.

Rather than risk a bogus data byte, it is best to ignore the first few data bytes in the FIFO. That's no big loss for displaying data, since the typical Mac window only displays between 500-1500 horizontal pixels at a time.

The analog preamp shown in Figure 2 was deliberately kept simple to avoid a profusion of components. A Burr-Brown OPA671 high-speed FET-input op-amp at U8 is configured for unity gain, but a National Semiconductor

LM336-2.5 precision zener reference (U9) is placed in the feedback loop.

A reasonably constant bias current is drawn through U9 to the negative supply by R3. The effect of U9 is that the output of the op-amp is held at a constant offset of +2.5 V above the input voltage. A tantalum capacitor provides added stability to U9 at high frequencies.

A FET input is crucial to the pre-amp's accuracy because bias current must be minimized. Any mismatch between the source resistances seen by the positive and negative inputs of the op-amp results in an offset voltage.

Normally, this problem is overcome by deliberately matching the two resistances. The pair of 1-M Ω resistors at R2 and R5 accomplishes this as long as

op-amp: the OPA671. It passes the test with no more than an 1-mV offset, which compares favorably to the ADC's ~20-mV resolution.

The input to the ADC is protected from overloads by diodes D1 and D2. A small resistor at R4 limits current flowing into the ADC in case either of its internal protection diodes has a lower forward voltage than D1 or D2.

The reference voltage for the ADC is unceremoniously taken from the +5-V power supply. If an independent reference is preferred, the surprisingly accurate LP2950 voltage regulator from National Semiconductor is an excellent choice.

Fortunately, the supplies are very clean in this circuit, owing to the use of low-current CMOS chips.

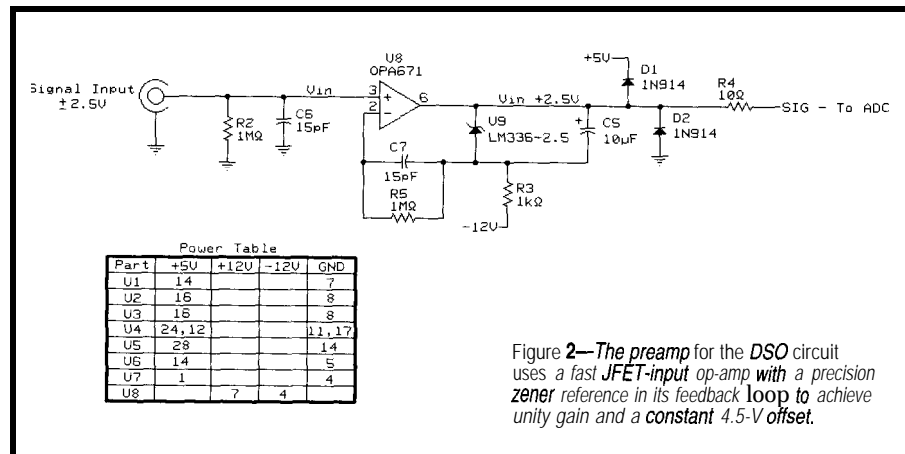


Figure 2—The preamp for the DSO circuit uses a fast JFET-input op-amp with a precision zener reference in its feedback loop to achieve unity gain and a constant 4.5-V offset.

the input is connected to a high-impedance source such as an $\times 10$ probe.

However, it is common practice to dispense with the probe when measuring signals produced by low-impedance sources such as op-amps. The problem is compounded by the fact that high-speed op-amps have very high bias currents to minimize time constants due to stray capacitances.

The preamp's acid test is in the amount of offset due to a mismatch of 1 M Ω between the two inputs of the op-amp. I applied a short across R2. A good high-speed bipolar op-amp such as the Linear Technologies LTC1360 shows a change of -250 mV! Such an offset could easily mimic the real signal offset you're trying to measure.

As of this time, I have identified one monolithic high-speed FET-input

CONSTRUCTION AND OPERATION

The DSO circuit presents no serious layout or construction challenges. I had no trouble getting it to work on a cheap solderless breadboard, probably owing to the exclusive use of CMOS logic with push-pull output drivers. In fact, I gave up designing its PCB when I found out how inexpensive small breadboard strips are.

When using flash ADC chips, a ground plane is essential as well as copious decoupling capacitors. Use at least 0.01 μ F at each IC power-supply terminal. The extra capacitors shown on the ADC stabilize three points in its resistor ladder for added noise immunity. By the way, use one of the outputs of U2 as a probe-test point for adjusting the trimmer capacitor on your scope probe.

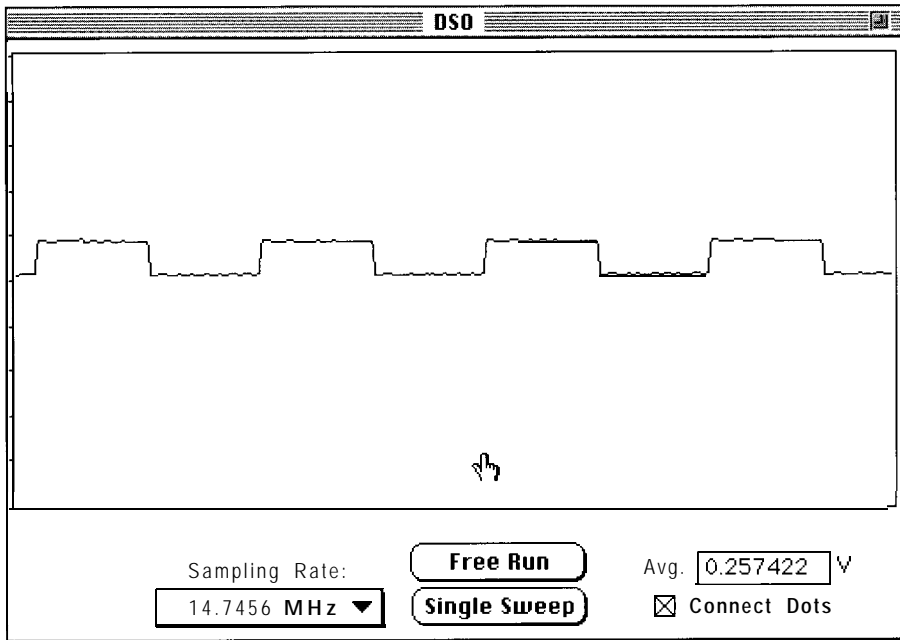


Figure 3—Apple's HyperCard development system provided an excellent framework for quickly prototyping the DSO support software on a Macintosh. Here, a 115-kHz logic-level signal was measured using a $\times 10$ probe to show the fast rise time, lack of overshoot, and low noise level.

The EEPROM firmware for the DSO board is unchallenging and fairly long, so I've uploaded it to the CCBBS. It responds to a very simple protocol consisting of single-character ASCII commands for collecting data at different sampling rates and then downloading varying numbers of bytes of data.

Since I was in a hurry to get the DSO circuit up and running, I wrote the support software on my Mac using HyperCard for the user interface and plug-in XCMD external code resources for the serial port driver and for fast graphical output. An additional XCMD implements an FFT option.

HyperCard is string-oriented and uses C's string format. In other words, its strings are of arbitrary length, terminated with a null. If the binary output of the DSO is read into a long string, zero readings terminate the string prematurely and data is lost.

A properly written XCMD takes care of this, but for the time being, I gave up an entire 0.03 dB of dynamic range by programming the MCU firmware to translate all occurrences of 0 to 1 prior to serial transmission. There is nothing magical about having a resolution of a part in 256. Conservative programming would probably identify data at the extremes of the input range as overloads anyway. A typical display from the DSO is shown in Figure 3.

CONCLUSION

A garden variety microprocessor or microcontroller can be souped up for high-speed data acquisition using one additional chip. The DSO circuit demonstrates remarkable economy of design, and I didn't have to worry about interfacing to the bus of a sophisticated computer at a time when bus standards are in a state of flux.

The sampling rate I chose for the circuit is not an absolute performance limit since faster FIFO and ADC chips can be obtained. However, the example demonstrates data acquisition at speeds exceeding the maximum sampling rate for an unaided microcontroller by an order of magnitude.

My design involved some compromises to keep component count low. There is definitely potential for improvement. Although the preamp is adequate for line-level audio signals and logic levels with an $\times 10$ probe, a multiple-gain preamp would make the circuit much more versatile.

A multichannel design is also quite feasible. Each additional channel requires its own ADC and FIFO, but all channels should share the same SCLK signal to guarantee synchronization. The ninth data bit should be implemented as a proper post-trigger input.

Whatever you build, don't get too attached to it. New high-speed data-

acquisition chips are being introduced rapidly and costs are plummeting. Analog Devices has announced the AD9022, a 12-bit, 20-MHz ADC. This chip with two parallel FIFOs would be a marriage made in heaven.

This design shows that powerful computerized instrumentation can be built with minimum circuitry and simple software. Good luck using this design for your own purposes! □

Francis Deck earned his Ph.D. in physics from Notre Dame in 1993. He is employed at an optics company in Fort Worth, Texas, where he pursues diverse interests in optics, process control, ultraprecision machine tool design, automated test systems, and technical management. You may reach him at fdeck@aol.com.

SOFTWARE

Software for this article is available from the Circuit Cellar BBS, the Circuit Cellar Web site, and on Software on Disk for this issue. See the end of "ConnetTime" for downloading and ordering information.

CONTACTS

DS2013-50, Sharp LH5499-50, and CA3318

Newark Electronics
12880 Hillcrest Rd.
Dallas, TX 75230
(214) 459-2528

PIC16C84 (10 MHz), CA3318, LM3362.5, and DS8921

Digi-Key Corp.
701 Brooks Ave. South
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

OPA671

Insight Electronics
9980 Huennekens
San Diego, CA 92121
(619) 587-1100

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

William Rogers

Autorouter and Board Layout Software Tool Analysis

Board design can be a monumental task. Working out an automated path gives greater project control and faster turnaround time. Choosing the correct autorouter is what this analysis is all about.



In choosing a PC board layout package, engineers need to know which ones

provide what the design requires and which special features are needed on the board. This is especially true for analog boards which require certain traces to track one another or where multiple ground planes or other layout-sensitive circuit portions exist.

Even for minor changes, turnaround time from engineer to administration, layout, and back again can take weeks, particularly in large companies. Picking the right layout package is a real time saver.

The goal is for the engineer to go from point A (the concept) to point B (the layout-Gerber tape) in a smooth, seamless, and painless fashion. The same philosophy holds true for integrated circuits, where fast turnaround has always been due to a silicon-compiler approach. Here, the engineer goes from concept to PG (Pattern Generation) tape for wafer fabrication.

INITIAL GUIDELINES

I wanted an autorouter that:

- routed 100% on all boards
- had a user-friendly interface
- was sufficiently inexpensive that one copy of the software could be on

- each engineer's desk or multiple users could access it via the network
- didn't require a guru to run the systems.

The routers considered for analysis were: Maxroute, Protel, Power router, Super router, MaxEDS, EDTcad, PADS-PCB, and Cooper & Chyan. Here's the approach I used for analyzing the autorouters.

Since vendors always have a set of demo boards which run on their specific autorouter, the question shifts from "Can you route your own demo board?" to "Can you route everyone else's demo board plus some user boards?"

I also evaluated the human-interface portion of each package. Although this evaluation is by its very nature subjective, I ordered and ranked them relatively and absolutely. The evaluation might look different if someone else, associated with a particular vendor or a long-time user of any given tool, did the analysis.

Only a partial analysis has been done because of the work required to convert data files, long route and re-route times, and frequent new releases from various vendors. Some software also had to be returned before evaluation was finished.

SPECIFICATIONS

The main computer system used to perform the testing was a 66-MHz '486 with local bus video, 1.5GB hard drive with ISA SCSI controller, 32-MB RAM, and 256-KB external to CPU cache. Peripherals included a Kyocera EcoSys laser printer, HP7475A plotter, 21" Hitachi and 21" NEC Multisync 6FG color monitors.

A second, similar computer system with a 55-MHz '486 was used for Cooper & Chyan.

DESIGN FLOW

Ideally, PCB layout complements the current engineering flow. Figure 1 illustrates a flow often used by engineering groups. Names of tasks may change, depending on the CAD tools used. Both digital and analog hardware-circuit design methods are incorporated along with software, math, and VHDL design techniques.

This particular flow is for an FPGA and board layout. The initial timing is determined for the system's digital portion. Schematic capture is followed by simulation and a loop back for corrections. The internal route is followed by a static-timing analysis.

The analog portion of the board goes through a similar path. SPICE is used for such things as op-amp design. Eventually, both hardware-design techniques arrive at board layout.

After conversion to the appropriate format (preferably, an automated approach is taken), the netlist must be placed and routed. The PCB layout portion of the chart appears small in relation to the other tasks.

Clearly, each portion of design is in itself a monumental task. Other por-

tions of the design flow may include mathematical computations for derivation of system algorithms, software for numerous functions and control of the system, and VHDL for upper-level architecture and simulation.

DEMO BOARD DETAILS

Twelve demonstration and user boards were used for this analysis. Table 1 gives the layers, clearances, track widths, grid, via and pad sizes, number of pins, dimensions, technology type, and the vendor who provided each board.

Demonstration boards DemoA and DemoB were available from multiple vendors and served as screening devices. In reality, it means little if DemoA and DemoB are routable. If

they are not routable, however, a real problem exists.

DemoD was a challenge for some routers. This board uses through-hole technology with an edge-card connector thrown in (i.e., a surface-mount component).

DemoE provides a combination of through-hole and 40-50% SMT components. It introduces fill areas that routers must avoid.

Demo5 is more of a memory-algorithm test bench with an edge-card connector. (The edge-card connector is included in all board measurements.)

There was a discrepancy in the number of components on the DemoD2 board. Mastech's version had 26 large components, while Protel's had 36 ICs and connectors. The actual size of the Mastech board was 6.1" x 5.0", but it had far fewer components. Protel's had 39% more components to route with only 10% more board area.

Demonstration board Tapeif, furnished by EDTcad, uses basic through-hole technology with an edge connector. The board dubbed Wavbrd is a small circuit which combines SMT and through-hole technology. Some memory is added as well.

The user board Recdbrd is the only one which uses PGAs (Pin Grid Arrays). It includes five PGA packages at 239 pins each, with four 96-pin connectors plus other miscellaneous components on a VME bus-style board.

The boards used here provide reasonably varied demonstrations of the autorouters' capabilities, especially when cross-pollinated between vendors.

ANALYSIS

I present the test results in graphs and tables. Keep in mind that the graphs only depict what each router did, and don't fault them for areas not attempted.

Figure 2 and Table 2 show the raw time it took each router to reach a certain percentage (ideally, 100%) of completion for any given board. On certain boards, most routers achieved no better

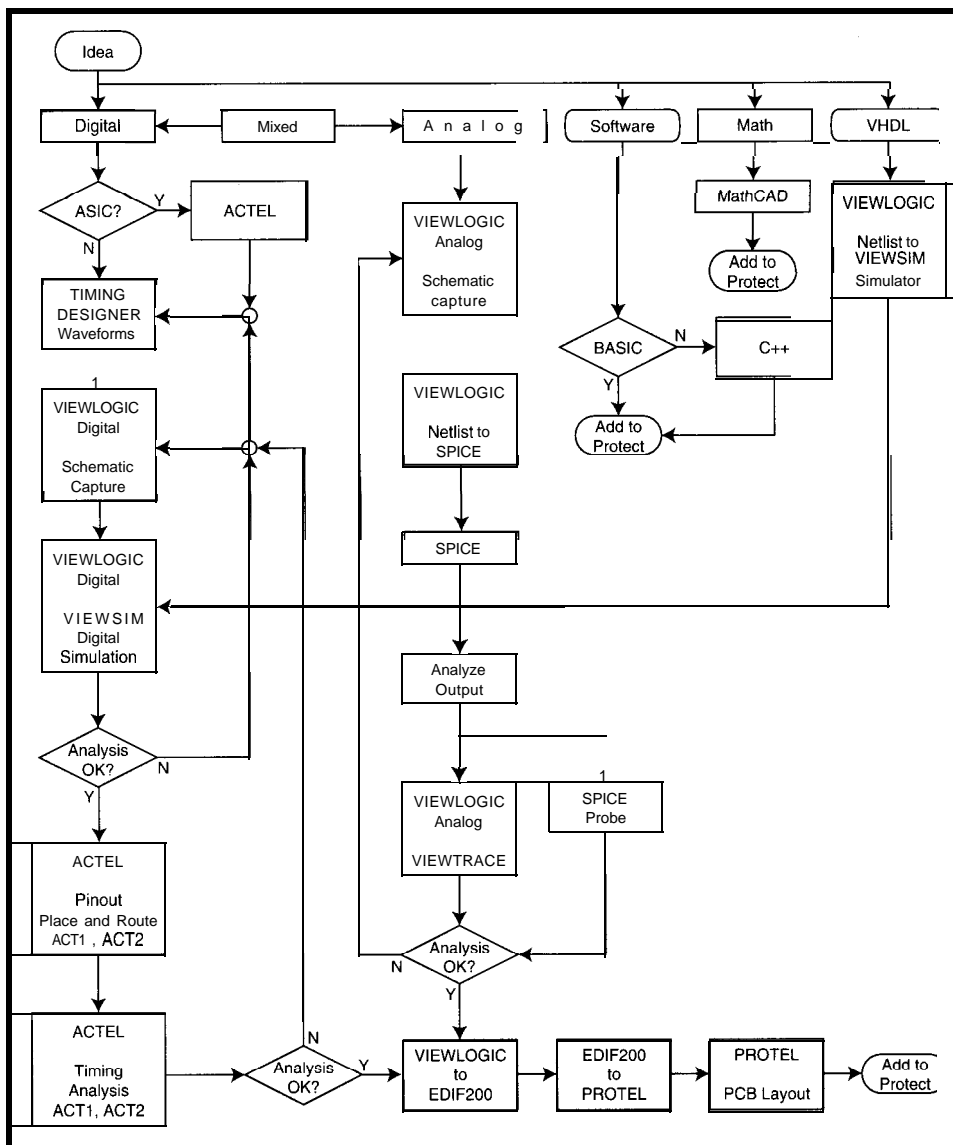


Figure 1-In a typical engineering flow for a given circuit design, the tools may vary, but the overall concept and functions are still valid. This design flow indicates where and when layout tools would be used in the design process.

Source	Board	Layers	Clearance	Track	Width	Grid	Via	Size	Pad	Size	Autoplace	Pins	Size	Technology
PADS-Power	DemoA	2	10	10	20	10	50	50				56		through-hole
PADS-Power	DemoB	2	10	10	20	20	50	50				36		through-hole
PADS-Power	DemoD	2	10	10	20	20	50	50				938	3.4" x 5.1"	through-hole, edge card
PADS-Power	DemoE	2	10	10	20	20	50	50				522	5.2" X3.7"	through-hole, SMT
PADS-PCB	Demo5	2	10	10	20	20	50	50				446	6.2" x 3.1"	memory algorithm, edge card, through-hole
Protel	Demo1	2	10	10	20	20	50	50				428	4.6" x 3.6"	through-hole
Protel	Demo2	4	10	10	20	20	50	50	Protel			2317	12.4" x 7"	through-hole
Mastech	DemoD2	4	8	10	20	20	40	50				945	6.8" x 5"	through-hole
Mastech	DemoE2	4	8	8	25	25	25	50				522	4" x 3"	through-hole, SMT
EDTcad	Tapeif	4	10	10	20	20	50	50				462	8" x 4.5"	through-hole with edge connector
User	Recbrd	8	10	10	20	20	50	50	Protel			2034	9.2" x 8.6"	through-hole with PGA
User	Wavbrd	2	10	10	20	20	50	50				392	13" x 4.6"	through-hole, SMT

Table 1—Several layout parameters were used by each autorouter to route the associated board. Using these parameters makes the input to each router the same so an apples-to-apples comparison of the output statistics could be made.

than the initial completion percentage it had within minutes or hours, even if it had all month.

Such situations call for a reasonable shut-down time that declares it done at less than 100% completion. But, it's difficult to know whether you use an 87% completion rate (done in 21 minutes) or a 90% completion rate done in three days? Although the answer is subjective, most of us don't have three days to wait for an extra 3%.

The data shown in Table 3 is also straightforward, with the same caveat as Table 2. Table 3 indicates the percentage of completion of any given board for each autorouter. These two pieces of information, along with some board and vendor statistics, help create the rest of the analysis graphs.

Figure 3 shows weighted completion percentages for the boards routed by Cooper & Chyan. The equation is:

$$P3^{10} \times P4^{10} \times P5^{10} \times P6^{10} \times P7^{10}$$

where P_x equals the route percentage to completion for a board in Table 3.

Since EDTcad routed just one board in this category, the "low but higher than it should be" result is ignored.

MaxEDS, on the other hand, routed all the boards and comes in third at 63%. Cooper & Chyan comes in second at 93%. The winner for this graph is Protel 2.0d at 100%.

Figure 4 graphically shows the combined weighted figure of merit where all boards routed by each vendor are summed with the following equation:

$$\text{numbers of pins} \times (\text{Completion Rate board 1} \times \text{CR2} \times \dots)^{10}$$

This shows a definite trend. Three routers excel in this category: Protel, MaxEDS, and Cooper & Chyan in that order. When comparing Cooper & Chyan to all other routers for only those boards routed by Cooper & Chyan, the weighted figure of merit results show the same trend.

The graph on human-interface efficiency shown in Figure 5 is totally subjective—a user must decide which type and style of interface they like. For me, the clear winner is Protel, which offers ease through autoplace, autoroute, setup, help manuals, library functions, file manipulation (this faltered slightly in later releases), and so on.

Since no library exists in Maxroute, Maxroute and MaxEDS came in at a distant second. Sloping off due to the DOS interfaces [even if they do pop up inside windows) are Power router and Super router. Also DOS-based is the PADS-PCB router. The DOS interfaces are similar and somewhat clunky.

Cooper & Chyan is only a router and depends on a third-party interface. In this area, it rates only 50%. The lower rating results from the 60% rating of the third-party interface and the extra involvement required to operate Cooper & Chyan on top of that interface.

The final DOS-interface candidate is EDTcad, at a very low 20%.

Board	Layers	MaxRoute V. 4.02	Power router	PADS-PCB	Protel V. 1.12	Cooper & Chyan	EDTcad	Super router	MaxEDS V. 2.02	Protel V. 2.0d
DemoA	2	0.05	0.11	0.22	0.1	0.22		0.15	1.00	0.10
DemoB	2	0.2	0.3	0.66	0.22	0.05		0.13	0.57	0.30
DemoD	2	4.8	5	10	44	4		50	5.57	6.75
DemoE	2	331	18	36	252	3.3		8	30	8.4
Demo5	2	80	6.5	4.2	14.5	25.95		11.5	50	3.97
Demo1	2		19.3	7.2	4.65	7.7	0.92	4	12.3	6.77
Demo2	4		270		112	11.85		538	53	180.5
DemoD2	4				73				49	106.5
DemoE2	4		32		170			28	16	13.25
Tapeif	4				9.5		3.23		23	13.1
Recbrd	8				86				690	88.5
Wavbrd	2				218				37.5	65

Table 2—This table indicates the time in minutes for each autorouter to route a given board. This information and the percentage completion data of Table 3 are the main sources of raw data. These numbers were derived from the router statistics and/or from using a stopwatch. Blank spaces indicate the board route was not attempted by the associated autorouter.

It is unfriendly. If it was all you ever used, I'm sure you could make it work, but who has the time?

In Figure 6, I look specifically at cost. I combined the weighted figure of merit data from Figure 4 with software cost. Here, Protel 2.0d is a clear winner. Its closest competition is 3% times away!

MaxEDS comes in second, while Cooper & Chyan comes in third to last. Cooper & Chyan is 15.5 times worse than Protel. Bear in mind, the cost of Cooper & Chyan doesn't include the third-party interface required for a complete system. Also, PADS-PCB was not figured into this portion of the analysis since it is freeware and thus would skew other values.

Figure 7 graphs Protel 2.0d's percentage completion across time. It illustrates the near-exponential rate most routes have. Cooper & Chyan is linear in nature, however. It ignores all DRCs until the end and at that time proceeds with a cleanup routine.

The other routers can do similar tasks if the track widths are very

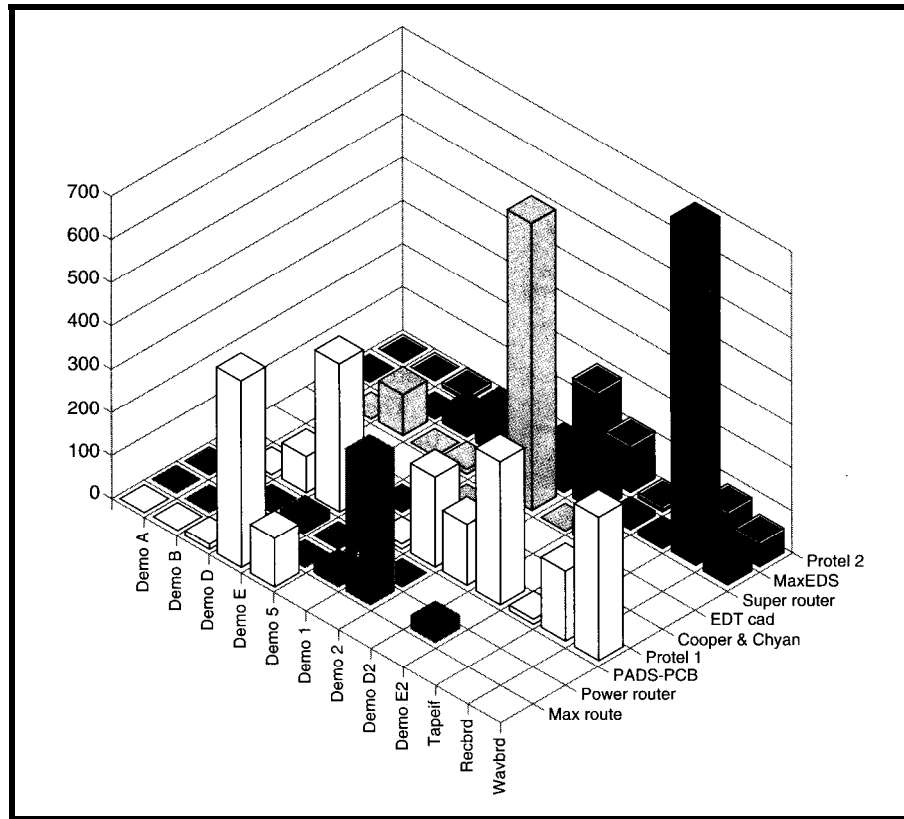


Figure 2-The time in minutes to completion graph highlights the boards' capabilities. The higher the columns on the graph the slower the autorouter was for that specific board. Notice that some routers have a very difficult time with certain boards.

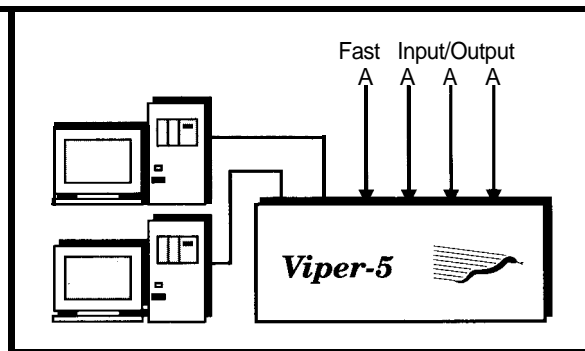
GFLOPS for the Masses

Super Vector Processing System

- 5 GFLOPS (4 Nodes)
- 2048 MB System Memory
- 6 I/O Ports (200 MB/sec each)
- FFT, Matrix Multiply, Convolution
- Scientific Math Library Included
- Real-Time Processing and I/O

Texas Memory Systems, Inc.

11200 Westheimer, #1000, Houston, TX 77042
 (713) 266-3200 Fax: (713) 266-0332
www.texmemsys.com



Viper-5



The *Viper-5* provides economical GFLOPS to meet the demands of the high-end scientific user by attaching to **Sun**, **SGI** and **Digital** workstations as a fast back-end scientific processor. The *Viper-5* is optimized for **FFT** (1M CFFT in 21 msec), **matrix multiply**, and **convolution** routines by the use of the custom-designed **TM-66 FFT** chip (680 MFLOPS). The *Viper-5* is programmed in C or FORTRAN with calls to an extensive Scientific Math Library

With a balanced architecture of 1-4 vector processing nodes (1¼GFLOPS each), large system memory (2048 MB), six fast (200 MB/sec) I/O ports, and system bandwidth of **1000 MB/sec**, all in a compact 10% " rack space, the *Viper-5* can be used for both real-time and high-end vector processing applications.

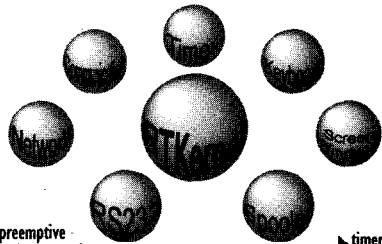
Professional Real-Time Tools

RTKernel Real-Time Multitasking for DOS and 16-Bit Embedded Systems

Real-Time Multitasking for: Borland C/C++, Microsoft C/C++, or Borland Pascal

RTKernel supports:

MS-DOS 3.0 or higher, Embedded Systems without DOS, Paradigm Tools (info available), Turbo Debugger, CodeView



- ▶ preemptive and cooperative scheduling
- ▶ up to 64 priorities
- ▶ your program's functions run as parallel tasks
- ▶ Windows can run in parallel with RTKernel tasks
- ▶ inter-task communication
- ▶ timer with 1 μs resolution
- ▶ real-time capabilities through event-driven scheduling
- ▶ drivers for screen, keyboard, up to 38 COM ports, IPX services, floppy disks, hard disks, etc.
- ▶ through Shared Code/Data, semaphores, mailboxes, message-passing
- ▶ unlimited number of tasks, about 1K per task needed
- ▶ timer rate adjustable from 0.1 to 55ms

Developer's License: \$550
complete Source Code: add \$500
no run-time royalties

RTTarget-32

Cross Development System for 32-Bit Embedded Systems

32-Bit Development System for: Borland C/C++, Microsoft C/C++, Watcom C/C++

RTTarget-32 supports: Intel 386/486 EX/SX/DX(2), as little as 16K RAM/ROM

RTTARGET-32

- ▶ boot code for floppy, hard disk, EPROM disk
- ▶ optional paging and RAM remapping
- ▶ FLAT memory model with physical = logical addresses
- ▶ program download at 115200 baud
- ▶ supports standard PCs and controller boards
- ▶ fully supports the C/C++ run-time systems (printf, malloc, etc.)
- ▶ interrupt latency < 5 μs
- ▶ serial communications library
- ▶ privilege level 0 or 3
- ▶ locator for Win32 PE-files

Developer's License: \$1700
complete Source Code: add \$1000
no run-time royalties

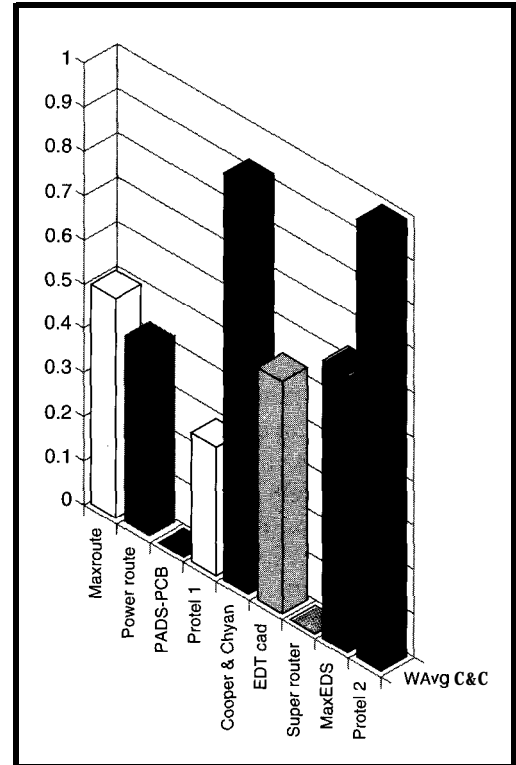
On Time MARKETING
Professional Programming Tools

Free demo disk! <http://www.on-time.com>
Please request Info Kit I <ftp://on-time.com>
info@on-time.com

In North America, please contact:
On Time Marketing
88 Christian Avenue
Setauket, NY 11733
USA
Phone (516) 689-6654
Fax (516) 689-1172

Other countries:
On Time Marketing
Karolinenstrasse 32
20357 Hamburg
GERMANY
Phone +49-40-437472
Fax +49-40-435196
email 100140.633@compuserve.com

Figure 3-A weighted-percentage-completed graph shows the routers which completed the majority of each board while accentuating the routers which did a lesser job percentage-wise on each of the boards.



small. After global track-sizing changes occur, you can use the DRC checker to locate errors. Errors are corrected with a combination of manual and automated methods.

The time per pin is the total time for the board to route divided by the number of pins on the board. The larger the value, the more time it took to route and the worse the situation. Figure 8 gives the combined average time per pin for all the boards routed by each package.

Keep in mind that not all boards are routed by all packages. Of the tools routing all boards that Cooper & Chyan routed, the order of preference is: Cooper & Chyan, Protel 2.0d, Power router, MaxEDS, and Protel 1.12. However, PADS-PCB also has a fairly low time per pin. PADS-PCB routed 86% of the boards routed by Cooper & Chyan.

Let me give some general observations on each of the routers tested.

MAXEDS

MaxEDS has no ongoing statistics to inform you of progress throughout the autorouting process. Some demos

do not route power and ground traces. I had to accommodate more resources on the computer for MaxEDS. There are no default values on the menu, and maximizing routes is strategy dependent. It has no library capabilities since it's only a router.

Also, the routing area is too large for the RAM (32 MB!). Swap space had to be increased from 4 MB to 16 MB. It was the only application which required additional memory. MaxEDS routes outside the boundaries of some

boards on the top. Reports must be printed from the initial engineering design shell rather than the more graphical shells, which require too much memory.

MaxEDS' human interface is reasonable but responds slowly to commands. It also has an extremely slow video-response interface. With screen-saver software, the screen takes ten minutes to return. Software accesses the disk drive every route, hampering speed.

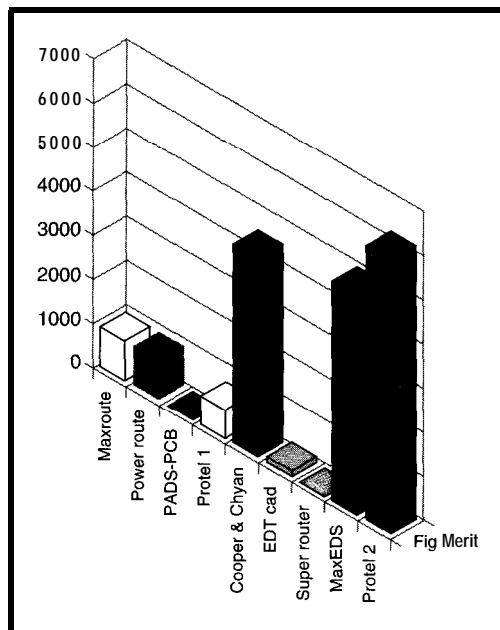


Figure 4-With the combined weighted figure of merit, autorouters with a falsely high percentage of completion (e.g., fewer pins of boards) are eliminated. It also tends to quantify somewhat the ease of importation and conversion into a layout tool itself.

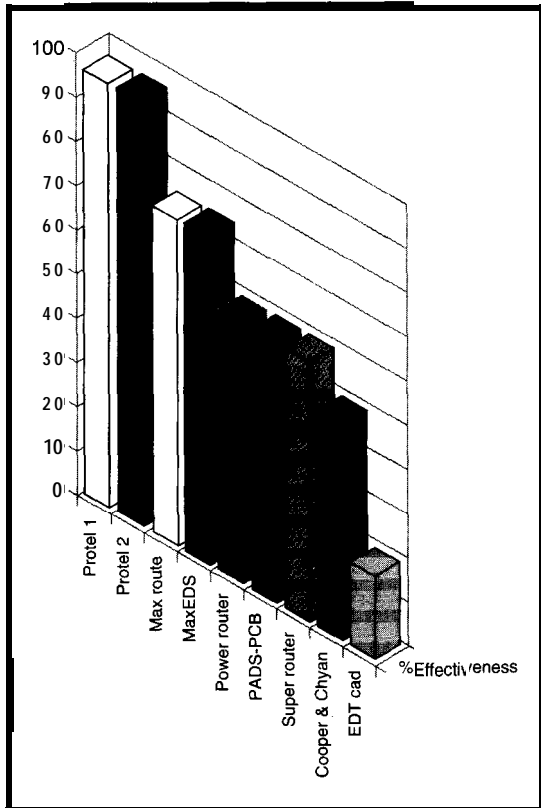


Figure 5—Human interface efficiency is purely subjective. The graph shows my rating of the ease of use for each autorouter. Note that Protel 1 has a higher rating than Protel 2 since the file-save capabilities were changed so you could no longer place your files exactly where you wanted them.

The good news is that the help system in MaxEDS is among the best. Import capabilities are also the most flexible and complete.

MAXROUTE

Maxroute does not free up the machine for other duties [i.e., it can't multitask]. Redraw is slow. The software doesn't remember the file names from previous file manipulation as current defaults. In fact, there are no default values on the menu. And, the menu changes names based on other menu selections (i.e., you can't see all menu commands at all times since many depend on the display).

The human interface isn't bad, but it's cumbersome. The help system is slow and relatively poor, and has a poor index. No error listings are included in the documentation.

PADS

PADS does not have a true Windows interface. Its DOS-like structure fitting under a DOS shell under Win-

dows is nonstandard. It doesn't have the file-traversing capability of true Windows packages.

PADS is highly strategy dependent! The strategies chosen for each board to maximize routing take numerous attempts to get the optimal strategy.

Some routers route two complete traces without making a connection to the middle of the trace—they only connect at vias. There's no ruler function on some routers.

The human interface is moderate. In most cases, the user must remember file names and directory structure. There are no default values on the menu. The help system is nonstandard and has a poor index.

PADS interfaces with thermal-analysis and multiple third-party tools, including a third-party library with 15,000 Digi-Key parts. It interfaces with Electrodynamic checking for crosstalk and impedance control with aggressor and victim signals. There's no stress-analysis interface, though.

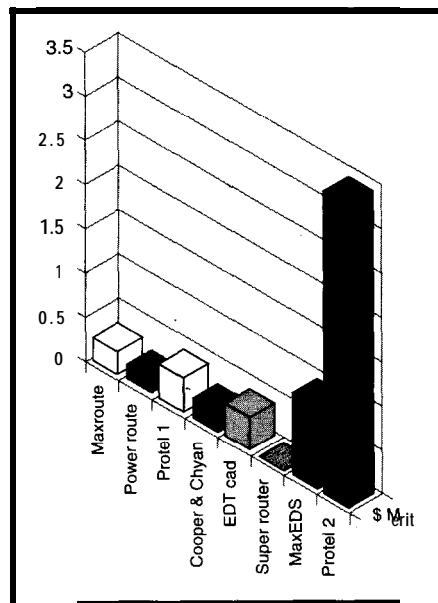


Figure 6—The weighted figure of merit with cost compares the combined weighted figure of merit in Figure 4 with the cost of the router. Protel 2.0d obviously does an excellent job of completion at a low cost.

BIG THINGS COME

DOMINO

Microcontroller



DOMINO Microcontroller
MODEL 8M

Starting at
\$79

Micromint's Domino 52 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, I²C 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 I²C
- 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 1-800-695-3355

(860) 871-6170 • Fax (860) 872-2204

PADS-PCB

As freeware, the price of PADS-PCB is right. Unfortunately, it can't handle larger PCBs and has no default values on the menu. However, the human interface is moderate.

POWER ROUTER

As with PADS-PCB, there are no default values on Power router's menu. Its human interface is again moderate.

SUPER ROUTER

Again, there are no default values on its menu, and the human interface is moderate.

PROTEL

Manual routing, a real bear on many packages, is only slightly unfriendly. On routes, however, it's not as strategy dependent as most routers appear to be.

The autoplacement tool is the only one I tested which does any good at all. The only others are the matrix placers, which are at best difficult to set up and nearly worthless when finished. Pro-

tel's autoplacement tool is quite easy to use, effective, and achieves its results elegantly.

The global autoplacement tool (with manual placement on some caps and alignment for memory chips onto the FIFOs and large VLSI) enables the Protel router to demolish its own previous routing times. Using local autoplacement and manual placement, route time for a specific board was 483 minutes to 96% completion. The global autoplacement tool and manual placement of some components (moved slightly for alignment with different part types) reduced the time to 71 minutes for 96% completion. What a difference!

I didn't use this technique anywhere while timing-board routing, but did check out the time savings. Hiding all display options (tracks, vias, etc.) while routing enhances the speed of the router by 2%.

Protel 1.12 had difficulties with SMT boards. Layer direction is critical if the library parts do not have stringers. Warning: Do not use "no prefer-

ence" on the direction for layers when routing or you'll end up with difficult-to-route spaghetti.

Version 1.12 also had trouble with edge connectors. For the best solution, route the edge-connector component(s) first with no preference for layer direction before routing the rest of the board. This order eliminates some of the autorouting hands-off, though.

A list of input routing parameters associated with the autoroute setup put into the log file would help immensely.

Version 1.5 more than doubled the route time and messed up the user interface accessing the file locations. File selection is poor. It chooses the directory above the selected one, dumps at root, and so on. It also keeps asking for the access code every time the tool is invoked. The older versions can't read the database it generates.

The route slows down exponentially toward the end (as do all others, except the shape router).

The prerouter supposedly places edge-connector stringers on automati-

E4 EPROM EMULATOR

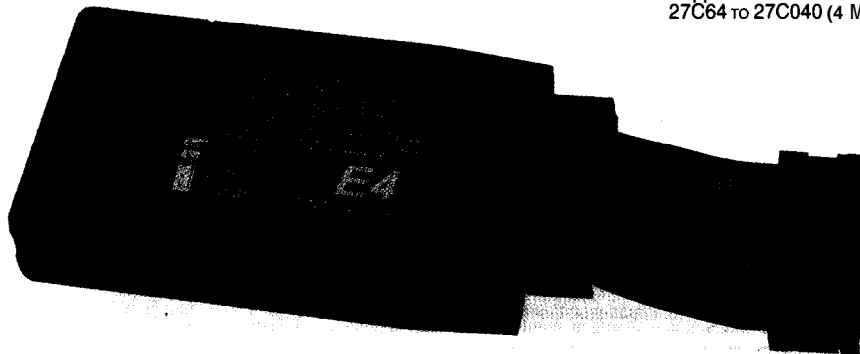
AMAZING Product.
AMAZING Price. Only \$249.
Supports EPROMs to 5 12Kx8.
Nothing extra to buy.

\$249

Supports all devices from
27C64 to 27C040 (4 MEG)

- Powerful PC software tools.
- Full screen and command line modes.
- Supports all data formats.
 - *Hardware error checking & correction.
- Software configurable.
- 100ns access time.
- Power-off data retention.
- High speed downloading (LPT1-3).
 - *Non-intrusive CMOS LP design.
- Chain up to 8 units together any configuration.
- Compact size with hard protective case.
- 1 year warranty & lifetime support.
- Discounts on 2+ units

Powerful tools, reasonably priced.



SDI

SCANLON DESIGN INC.

TEL: (800) 352-9770 Toll Free in
(902) 425-3938 North America
FAX: (902) 425-4098 Prices are in US Dollars.
EMAIL: 71303.1435@COMPUSERVE.COM
5224 BLOWERS STREET, HALIFAX, NS. CANADA B3J 1 J7

Order today or Call or FAX for details.

(800) 352-9770

Board

DemoA
DemoB

DemoD
DemoE
Demo5
Demo1
Demo2
DemoD
DemoE
Tapeif
Recbrd
Wavbrd

Table 3-7
1.00 indicat

cally wi
which h
and y le
not in t
feature
2.0d!)

The:
zoom 01
put whe
section
routing.

38



\$C

U

PR

- DOEE
- CMO
- EASIE
- POWI
- MICR
- PLCC
- SUPE

OTHI
8088
PC F
16 B
WAT

'EVA
BRP
5 Y

HRB:

Board	MaxRoute V. 4.02	Power router	PADS- PCB	Protel V. 1.12 & Chyan	Cooper	EDTcad	Super router	MaxEDS v. 2.02	Protel V. 2.0d
DemoA	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00
DemoB	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00
DemoD	0.99	1.00	1.00	0.94	1.00		1.00	1.00	1.00
DemoE	0.95	1.00	1.00	0.97	1.00		0.48	0.98	1.00
Demo5	0.99	1.00	0.93	0.97	0.99		1.00	0.98	1.00
Demo1		0.93	0.52	1.00	1.00	0.94	1.00	1.00	1.00
Demo2		0.99		1.00	1.00		0.99	1.00	1.00
DemoD2				1.00				1.00	1.00
DemoE2		0.92		0.99			1.00	1.00	0.99
Tapeif				1.00		0.92		1.00	1.00
Recbrd				1.00				1.00	1.00
Wavbrd				0.89				0.98	0.98

Table 3—This table shows the percentage of completion that each router achieved on each board routed. A value of 1.00 indicates that 100% of the board was done by that router.

cally with some unknown algorithm, which has something to do with the x and y length of the pads. But, this is not in the documentation. (Note: this feature works much better in version 2.0d!)

The zoom control goes back to full zoom on some versions-it won't stay put when you want to look at one section at the same time that it is routing.

When routed with zero-width lines 0.001 mils and 0.005 mils, the router was about 12.3 times faster finishing the board to 100% compared to only 94%. A selection of "all" with a global-trace change to the proper width gave the near-final result. The manual cleanup due to DRCs took approximately 10 minutes. This specific example was demonstrated on the demo board.

The previous route time to 94% was 44 minutes while the current route time to 100% was 3.58 minutes. Added to the manual cleanup, this time totals only 14 minutes. There were 6-10 DRC errors on the board before manual intervention. This example was run on 1.12.

Version 2.0d has serious enhancements which affect SMT and edge-card-connector routing capabilities significantly. Routability enhancements yielded 97.5% to 100% completion rates with speed-ups: 3.35, 3.65, 6.5, 12.8, and 30.0 times on various boards.

This improvement over 1.12 is enough to send off hearty congratulations. All the boards with speed increases belonged to other vendors. Protel actually slowed down on their own demo boards.

Screen redraw is much faster than PADS. The Help utility is true Windows style and is reasonable in content. It rates as one of the most helpful. Default values on menu are good, and there's a good human interface.

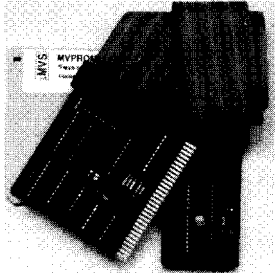
386 SBC \$83



OEM (1K) PRICE
INCLUDES:
- 5 SER (8250 USART)
- 3 PAR (32 BITS MAX)
- 32K RAM, EXP 64M
- STANDARD PC BUS
- LCD, KBD PORT
- BATT. BACK. RTC
- IRQ0-15 (6259 X2)
- 6237 DMA 6253 TMR
- BUILT-IN LED DISP.
- UP TO 8 MEG ROM
- CMOS NVRAM
USE TURBO C,
BASIC, MASM
RUNS DOS AND
WINDOWS
EVAL KIT \$295

\$95 SINGLE PIECE PRICE UNIVERSAL PROGRAMMER

- DOES 6 MEG EPROMS
- CMOS, EE, FLASH, NVRAM
- EASIER TO USE THAN MOST
- POWERFUL SCRIPT ABILITY
- MICROCONT. ADAPTERS
- PLCC, MINI-DIP ADAPTERS
- SUPER FAST ALGORITHMS



OTHER PRODUCTS:
8088 SINGLE BOARD COMPUTER OEM \$27 95
PC FLASH/ROM DISKS (128K-16M) 21 75
16 BIT 16 CHAN ADC-DAC CARD 55 195
WATCHDOG (REBOOTS PC ON HANGUP) 27 95

- EVAL KITS INCLUDE MANUAL BRACKET AND SOFTWARE.
- 5 YR LIMITED WARRANTY
- FREE SHIPPING
- HRS: MON-FRI 10AM-6PM EST

MVS MVS BOX 850
MERRIMACK, NH
(508) 792 9507

CONTROL RELAYS • LIGHTS • MOTORS MEASURE TEMPERATURE • PRESSURE • LIGHT LEVELS • HUMIDITY INPUT SWITCH POSITIONS • THERMOSTATS • LIQUID LEVELS

MODEL 30 \$79



• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

MODEL 45 \$189



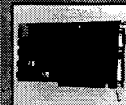
• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

MODEL 100 \$279



• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

MODEL 60 \$179



• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

MODEL 40 \$99



• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

MODEL 70 \$239



• 24 BIT A/D
• 16 CHANNELS
• 16 BIT OUTPUT
• 16 BIT ADDRESS
• 16 BIT DATA

NEED A CUSTOM PCB? TRY US.

PRAIRIE DIGITAL, INC.

PHONE 608-643-8599 • FAX 608-643-6754

146 SEVENTEENTH STREET • PRAIRIE DU SAC, WISCONSIN 53157

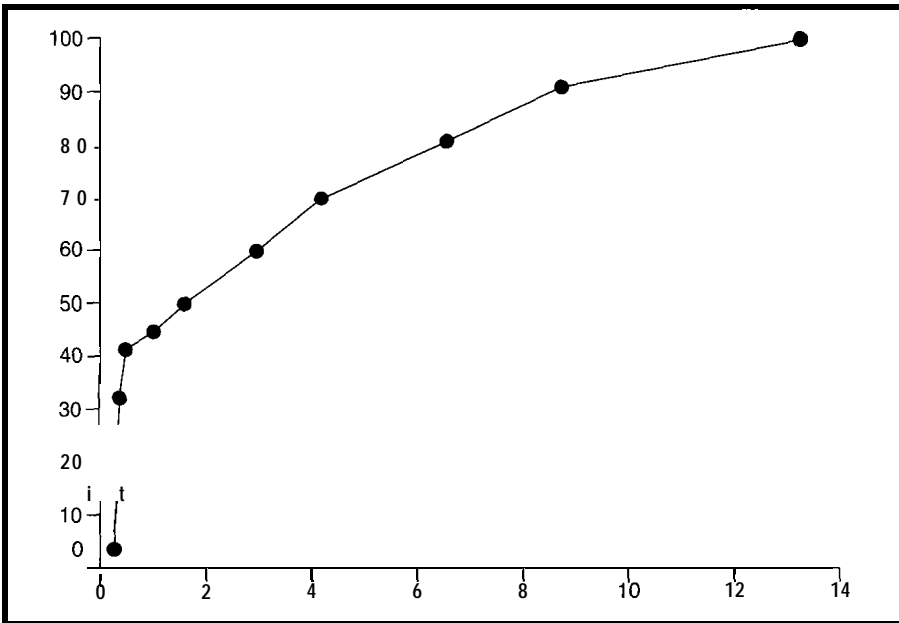


Figure 7-The typical Protel percentage completion over time graph shows that the first few seconds of routing are at a rapid rate. As the number of routing channels reduces, the amount of time increases.

Library components are easy to use and can be modified quickly.

EDTCAD

EDTcad is extremely difficult to use. It has a router, library, schematic

capture-the whole enchilada! However, this product is an enigma since it has all functions, but no appeal and no user-friendly interface capabilities.

The library lacks components and includes nonexistent ones. It's difficult

to use and add components. I needed much help from the technical-assistance folks to accomplish seemingly standard tasks.

There's no push-and-shove or rip-up-and-retry route capability. (A future version may go to a shape-based and gridless approach.) There's no auto-place, and this is the only product that didn't route its own demo completely.

The difficulty in using this product forces one to consider slightly more expensive systems with far better user interfaces.

COOPER & CHYAN

Cooper & Chyan has no library capabilities; it is only a router. Its interface is dependent on third-party or direct input of commands.

There are various options for advanced capabilities for design rule checks, design for manufacturing, design for test, auto insertion and spread of traces as well as hybrid, wire-bond, and auto insertion of vias under SMT pads.

HOME AUTOMATION

Software for Windows

Wireless Sensors
Rule-based Logic
Wireless Controls

CYBER HOUSE

Visit our web site for a complete product description and to get a working demo

<http://www.savoysoft.com>

or call 800-527-2853

SAVOY
Software Development

Designing Something?

This Parts List Software will help you stay organized.

For Engineers, Product Designers and Prototypers

Easily create and manage multi-level parts lists for products in development.

Keep track of:

- Part Specs
- Drawings
- Suppliers
- Product and Parts Costs
- Engineering Stock

V1.5

Parts Vendors™

for independent and departmental projects. Use alone or in a workgroup.

Version SE: \$99 + shpg Call 800-280-5176

EXtended: \$299 + shpg

TrilogY DESIGN voice.916.273.1985 fax.916.477.9106

Requires Win 3.x or Win95, 486, 8 meg min. ram.

P.O. Box 2270, Grass Valley, CA 95945

<http://www.trilogdesign.com>

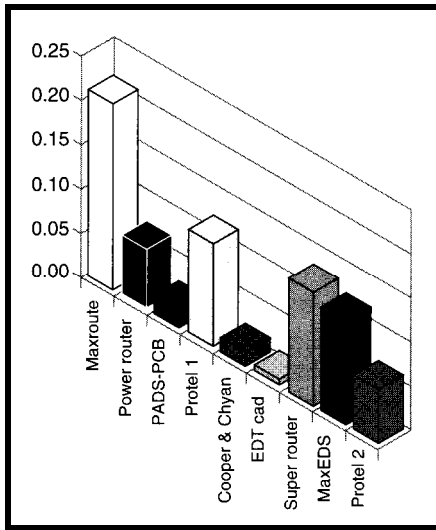


Figure 1 The graph visualizes the average time per pin. Here! as always, you must be careful about the data available. Autorouters with a lower time per pin may have only done the easy or fewer boards. An autorouter that did few boards and had a high time per pin indicates a timeconsuming router.

This is the only router handled by a field representative and not by me.

GENERAL OBSERVATIONS

All routers are extremely board dependent.

I would prefer to see a larger library for all the routers. Some interfaces to existing third-party libraries would be helpful, especially for companies with whole catalogs of components.

None of the vendors had relative locking of components such as bypass capacitors to integrated circuits. Protel was the only vendor who did a decent job of placing bypass capacitors automatically.

It would be nice for certain blocks of logic or analog functions to have a relative locking mechanism. Grouping would allow the circuit in its entirety to move or route at various locations around the board without affecting the subcircuit being prerouted. This tool is especially helpful for analog boards.

I would like to see these companies merge and produce a product with:

- the human interface, autoplacement, and some routing features of Protel
- the libraries and third-party tool interfaces of PADS
- the routing capability of Cooper & Chyan
- the import and export capabilities of MaxEDS

As another bell and whistle, I'd recommend that the routers interface to point-to-point wire-board manufacturers in some very difficult cases.

I'd also suggest a different routing methodology—route with zero-width lines, convert those to full-size tracks, then do DRCs on the interim route. Go back and clean up the DRCs (automatically, of course). This method would probably be similar to a shape router.

The user can accomplish this routine with satisfactory results on small boards. Larger ones have a significant number of errors which are difficult to clean up manually, since the user must edit each trace with a DRC error. No automated clean up currently exists.

Total route time for one medium-to-large board was less when done in this fashion than when using the autorouter. A partial manual and autoroute capability can be used by unrouting the net and then autorouting it again with the proper widths and DRCs in place.

EDTcad is old product-style software. You'd be better to purchase Protel for two to four times as much but one-tenth the headache.

RECOMMENDATIONS

Engineers need to analyze which package matches a project's needs. A moderately priced autorouter usually fits your current and future needs.

This particular analysis leans most often towards Protel as the autorouter of choice.

It offers numerous functions such as autoplacement, library edits, good routing at reasonable speed and price, and has expandable interfaces to schematic capture, shape routers, and so on. For the best Computer Aided Engineering package, my recommendation is Protel 2.0d. □

I'd like to thank all the vendors for their support in this analysis. This type of in-depth analysis required a total full-up system. I also looked at other vendors. The ones who did not release a complete software tools package for analysis were not considered.

William Rogers, P.E., is a senior electronics design engineer with almost 20 years of experience in telecommunications and video imaging. He has been consulting for over 3 years in the field of ASIC design using VHDL, Mentor and Viewlogic simulation tools, and various synthesizers such as Synopsys, Viewsynth, and Autologic. He may be reached at wrogers.scc@worldnet.att.net or (214) 2157835.

SOURCES

Cooper & Chyan

Cooper & Chyan Technology, Inc.
1601 Saratoga-Sunnyvale Rd., Ste. 255
Cupertino, CA 95014
(408) 366-6966
Fax: (408) 252-9565
info@cctech.com

EDTcad

Electronic Design Tools, Inc.
2700 Flora St.
Dallas, TX 75201
(214) 871-9495

Maxroute, MaxEDS

[now called Layout]
OrCAD
9300 S.W. Nimbus Ave.
Beaverton, OR 97008
(503) 671-9500
Fax: (503) 671-9501
info@orcad.com
<http://www.orcad.com/>

Power router, Super router, PADS, PADS-PCB

PADS Software, Inc.
165 Forest St.
Marlboro, MA 01752
(508) 485-4300
Fax: (508) 485-7171

Protel

Protel Technology, Inc.
4675 Stevens Creek Blvd., Ste. 200
Santa Clara, CA 95051
(408) 243-8143
Fax: (408) 243-8544
<http://www.protel.com/>

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

FEATURE ARTICLE

Mike Smith

The Evaluation Board Saga Continues Low-Cost Educational Tools

Strapped by ongoing budget cuts, Mike struggles to get the resources he needs to teach his students without limiting them to convoluted freeware. With SDS's demo program, he hits pay dirt. Read on for details.



It's been over a year since I last discussed using the commercial Advanced

Micro Devices' SA-29000 RISC and Motorola's M68332EVK CISC evaluation boards. There, I suggested that freeware provided a cheap route to keeping a university laboratory up to date (*INK 44*). Since then, another class of 58 fourth-year computer engineering students has worked through my course.

But next year, because of curriculum changes, a combined third- and fourth-year class of about 190 will come my way. Most students will be taking the course a year earlier in their program. It's time to make things as straightforward as possible!

All the things I said in the original article about the suitability of the 68332 evaluation board for education are still valid. We've developed an add-on board with PITs and a DUART for interfacing to student projects. We can download common C and assembly-language-callable I/O utilities into the board's RAM and protect them by modifying the 68332 chip-select options.

However, you'd have thought that the supplied Motorola freeware was written by the White Rabbit and all the students were called Alice. If there

was a possible hole presented to them by the public domain software, they fell into it!

Motorola provides a freeware cc68k compiler as part of their board kit. The students complained that the compiler was old-style C and would not accept prototypes, even though that's the sort of problem you have to cope with in real life.

A bigger problem was the fact that somebody had built the compiler to use only the lower 16 bits of a number with sign extension even for 32-bit integers. Just how did Motorola expect you to access the MC68332 special registers which are up at the high end of memory (0xFFxxxx)?

I didn't like Motorola's freeware assembler as it would not support names for registers, which is important in developing structured assembly code. I ported an MC68000 cross-assembler from the Commodore Amiga to a Sun running UNIX. It worked perfectly on both systems.

However, when it was compiled with Turbo C and run on a PC, it developed a bad habit I couldn't cure. After an `include` directive, it would always eat either the next five lines or characters.

Mind you, this is easily solved. Any old hand at programming could see the easy work-around for this problem—simply put five lines of comments after each `include` statement!

Then, there was the fact that this old 68000-based assembler didn't always correctly handle the 68332 `MULTIPLY`. Imagine the students' problems and complaints!

Don't they realize software is never perfect? Can't they appreciate that they were experiencing real-life situations as a no-charge bonus?

By the end of this last session, and with the prospect of another 200 students pounding on my door next year, I'd had enough. With a 30% cut in education funding, I sent this plaintive fax to a number of software developers listed in Motorola's source book:

I have 20 Motorola M68332EVK evaluation boards running on MS-DOS machines (no Windows). I need a good compiler and assembler

for teaching. Funding cuts mean I have little money. What can you do for me?

I'm still waiting to hear from some companies, but I got a fast response from Mark Bracey of Software Development Systems in Illinois. Like a good salesman, Mark first tried to sell me complete systems at several thousand dollars a shot.

But, no matter how big an educational discount you offer, 20 times anything to get legal copies is still a lot of money!

Mark then offered SDS's free C/C++ Starter Kit for the 68k, which is used to demo a 68k product line. I shuddered as he mentioned limited capability.

I agreed to try the kit and planned on investigating it on sabbatical. But, if the kit proved unsuitable for teaching, I intended to call him back. I'd mutter those magic words that send any salesman on a guilt trip, the prelude to getting freebies:

"My students will be future engineers and managers. You don't want to leave them with a bad opinion of your product. They'll buy from your competitors."

MUTUAL BENEFIT?

In the '70s, engineers and computer-science students were trained on DEC's PDP-8s and PDP-11s.

When those students hit the marketplace, they wanted to work on more PDP-8s and -11s. It was a deal

that benefitted both the university and DEC. The university got the equipment, and DEC got a good chance at increased sales later on.

SDS is playing the same stakes. By providing a kit now, students pay with their own time and money (i.e., university fees) to train on SDS software.

To SDS's further advantage, the students get more detailed training than could be achieved otherwise. With a good product, SDS has a head-on advantage when the computer engineering student hits a managerial position in the next few years:

Surprisingly, the SDS demo kit lived up to Mark Bracey's hype. I'll show you how to get the SDS utilities to run. The kit is a real step up from Motorola's freeware. Instructors, try it out.

RUNNING THE ASSEMBLER

Because of funding cuts, we don't have Windows on our laboratory systems, making unpacking the SDS demo software a bit of a hassle. The install program runs only from Windows, but an ftp from another system solved that problem.

The C/C++ compiler, assembler, and linker work directly from the DOS command line or Windows. There was little documentation sent with the kit.

However, I was able to work most things out using the tiny help file provided when you activate any SDS utility with a U option. (The manuals are worth their \$90 fee.)

I teach my microprocessor course with a C bias. You need to know

enough assembler to appreciate what a processor does and the limitations imposed by its instruction set and architecture.

You need a little more assembler to develop device interfaces and customize the code on special processors or situations.

Finally, you need to know stack operations to interface reliably to other code. For the rest of the time, I encourage the students to work in C with the optimizer turned on.

Because of this bias, I developed a series of C-like utilities to use with the M68332EVK evaluation board. The students call these from their assembly or C code exercises and projects.

The code is essentially I/O and math routines which make use of SY S trap calls to the onboard Motorola monitor. These routines communicate with the user over a serial line to the screen and keyboard of an MS-DOS machine.

With about 2000 lines of code, the only problems during the changeover to the SDS assembler were the old freeware assembler's:

- **EQU R**-to give symbol names to registers
- **EVEN**-to align instructions on word boundaries
- **ABSOLUTE** addresses-to generate S-records to download to the board
- no code sections or external references
- lack of linking object file groups

The first problems were solved with a couple of defines at the start of the SDS code:

```
#define EOUR EQU
#define EVEN .align 2
```

The other problems required a few minutes' work with the editor to introduce global (X D E F) and external (X R E F) references.

What I liked about the SDS assembler was that it accepted a variety of comment symbols including my favorite-the double-slash // from C++. I think this comment format makes the code easier than the standard format * or ;.

Listing 1—SDS's memory specification file is very flexible.

```
partition overlay {
  region reset: /* reset vector */
  region vects; /* other vectors */
  region code[addr=0x0]; /* executable code */
  region init[addr=0x4000]; /* C++ init thunks */
  region exit: /* C++ exit thunks */
  region const; /* constant data */
  region string; /* constant strings */
  region data[roundsize=4]; /* initialized on reset */
  region ram[roundsize=4]; /* zeroed on reset */
  region malloc[size=0x1000]; /* malloc space */
  region stack[size=0x800]; /* stack */
  STKTOP =; /* SP reset value */
  DATA =; /* "data" download addr */
  example; RAM[addr=0x300000];
}
```

USING THE LINKER

The linker in the SDS utilities was a definite plus over the original evaluation board software. Combining several object files is a more realistic task than always working with one large source file.

In the SDS starter kit, the linker has been crippled to handle only 64 KB of code, three object files, and the provided C library. Both float and integer libraries are provided.

This handicap does not pose much of a limitation with the way I teach and use the boards. My I/O libraries are downloaded into the extra RAM on the M68332EVK boards.

The chip-select lines are then modified to make the RAM read only. This change protects the library from student program runaways. Having the RAM library as a large single block of code causes no problems.

However, students don't like to download all the library functions every time. This complaint is understandable since the full library consists of a long S-record file transmitted from a well-used Sun server to a host PC to the board over a serial line. It takes a long time to move 16K of code every time the evaluation board hangs.

Instead, students wanted to download small sections of the library as needed while having the proper links maintained between the various sections. To do this, parts of one library must overlay another library already in RAM. In other words, it works as a sort of self-modifying load.

For example, you might place a small help file into RAM to provide online support at the board level when debugging. A more detailed help file could be downloaded later if needed.

Although it may have not been the way SDS expected their linker to be used, I found it simple to get these link overlays to occur. Listing 1 presents the linker's memory specification file which describes how the various sections of code are placed in memory.

It is possible to detail the location in memory where each assembly code section resides with this format of linker-specification file record. Getting the load overlaying as I wanted was easy. I just placed one section of the utilities library in section code and the other in section `i n i t`.

LINKING TO THE LIBRARY

Having the utilities library locked in RAM solves two problems. First, the students only have to load the code once during a lab session, which saves a lot of transmission time over the serial link. The second thing is that I have no choice—neither the original freeware nor the SDS demo kit came with a library generator.

The problem remains: how to generate an equates table the student can include in their code which directs the assembler to the correct function locations in the library.

Previously, with the Motorola free assembler, I dumped a symbol table and then ran the information through a filter written in C. It's a straightforward method if you know what you're trying to achieve. However, for students having trouble with basic addressing modes, it was just another level of "magic."

Again, it may not be the way SDS planned it, but their symbol extractor utility enables you to generate equate files with a few command line options using a `printf()`-style format.

For example, take a look at Listing 2. There, the `h` suppresses symbol-table column headings, `P` is the print format for the output file, `%n S` prints the name of the symbol, `%v x` prints the value of the symbol, and `-o ut.i l s . e q u` is the symbol table file.

THE C COMPILER

The C compiler works great. It not only supports prototyping, a necessity these days, but also C++.

However, I am not looking forward to explaining how to link an assembler

Listing 2—SDS's symbol extractor utility generates equate include files using a `printf()`-style format.

```
sym -h -P "%n S EQU 0x%v x\n XDEF %n S" -o utils.equ utils.out
```

RELAY INTERFACE
PROVIDES SOFTWARE CONTROL OF RELAYS

AN-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.
AN-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

ADC-16 A/D CONVERTER* (16 channel/8 bit)...\$ 99.95
ADC-8G A/D CONVERTER* (8 channel/10 bit)...\$ 124.90
Input voltage, amperage, pressure, energy usage, light, dynamics 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). Includes Data Acquisition software for Windows 95 or 3.1
ADC-8E TEMPERATURE INTERFACE* (8 ch)...\$ 139.95
Includes term. block & 8 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, keypads, and other devices.
PS-4 PORT SELECTOR (4 channels RS-422)...\$ 79.95
Converts an RS-232 port into 4 selectable RS-422 ports.
RS-422 (RS-232 to RS-422 converter).....\$ 39.95
*EXPANDABLE...expand your interface to control and monitor up to 512 relays, up to 576 digital inputs, up to 128 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 QuickBasic, GW Basic, Visual Basic, Visual C++, Turbo C, Assembly and others are provided.
*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).
*FREE INFORMATION PACKET...use our 800 number or e-mail to order, or visit our Internet on-line catalog.
URL: <http://www.ecei.com>
Technical Support (614) 464-4470

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

Internet E-mail: ecet1@hm.net
International & Domestic FAX: (614) 464-4499
Call for information, technical support & orders

ELECTRONIC ENERGY CONTROL, INC.
390 South Fifth Street, Suite 604
Columbus, Ohio 43213-5491

subroutine as a private function in a C++ class.

Although I complained about Motorola's freeware C compiler, it optimizes code well, even though it confused the students. The SDS kit also supports various optimization capabilities which can be used to illustrate the effect of optimization. But, they can also be turned off to avoid confusing the students early in the course.

The most useful options in this context are:

- L-lifetime analysis for register use demonstrates register activity. It naturally leads into a discussion of the consequences of the increased number of register operations found on RISC processors
- - S-size versus speed optimization is useful for embedded systems. This feature shows the effect of poorly coding frequent loops and various switch-optimizing techniques

Like the assembler, the SDS C compiler produces code for various Motorola processors. This flexibility should make it possible to demonstrate the effect of architecture on code generation and performance.

The C/C++ compiler in the kit is crippled so that it produces an object file and an assembler listing, but not a straight assembler code file. To me, this was not a disadvantage.

However, I must admit that I nearly lost it with the frustration of figuring out how to link the SDS C code and assembler object files together.

It was simple to link assembly and C code for functions such as:

```
void Test(void).
```

But, a strange error message occurred for functions like:

```
void Test(int).
```

Then I'd get the grievous message:

```
Function calling convention  
mismatch
```

At first, I thought that SDS had introduced prototyping into assembler code,

which is a neat concept! However, in checking with the SDS support line, I found out I was far off the mark.

Since I was stuck in the ivory tower, nobody bothered to tell me there were two approaches to stack management on the 68332. Either the calling procedure cleans the stack [what I always taught], or else the called procedure cleans up (what everybody else in the world is doing!).

The default for the SDS compiler was opposite to what I'd used with the SDS assembler. Invoking the compiler's - Od option switched its stack management and solved that problem.

Although it didn't seem so at the time, it was neat that SDS designed into their linker the ability to recognize programmer inexperience [i.e., the student)! This kind of small detail gives you confidence in probable functionality and usefulness of SDS's other utilities in a teaching environment.

ACTIVATING THE C LIBRARY

The starter kit includes both integer and float C library as object files and not the source. I persuaded SDS to provide me with a library compiled with the Od option so that its stack operation matched the way I wanted to teach my course.

As it turned out, SDS designed the library to work with their own on-board monitor, but I could not afford the upgrade. However, SDS thought ahead again. They left the `f g e t c ()` and `f p u t c ()` utilities out of the compiled library and provided their C code.

It won't take long to link those to the Motorola system character-oriented functions provided with the M68332EVK board monitor. Thus, the library is completely functional as an educational tool in my environment.

WHERE TO NEXT?

Many of you may think I'm too enthusiastic about this demo system. But, I look forward to less work because of it, so why shouldn't I be. However, you've seen nothing yet!

According to Mark Bracey's hype, the SDS starter kits include simulators for the Motorola 68000 series and PowerPC processors. These simulators can be activated using a Windows-

based debugger. It sounded like a neat set of teaching tools and something students could use at home.

I wanted to use these simulators in the lab and for assignments. However, I didn't have any copies of Windows 95 in the lab yet! I wonder how Microsoft would react if I sent Bill this fax:

Dear Bill,

I have 20 Motorola M68332EVK evaluation boards running on MS-DOS machines. Educational funding cuts mean I have little money. I need a good Windows program. What can you do for me? ☐

Special thanks to Mark Bracey of Software Development Systems for coming up with a solution that benefits both of us. Also, many thanks to Anthony Skiba on the SDS support line for some fast clear answers.

I would also like to thank the University of Calgary for the opportunity to follow up some ideas in research and teaching.

Mike Smith is a professor in electrical and computer engineering at the University of Calgary in Canada. He recently won the Sanford Fleming Foundation's Wighton 1994 Fellowship for innovative teaching in engineering undergraduate laboratories. He teaches courses on C, microprocessor interfacing, and comparative processor architecture. He may be reached at smith@enl.ucalgary.ca.

SOURCE

SDS C/C++ starter kit for the 68k
Mark Bracey
Software Development Systems
815 Commerce Dr., Ste. 250
Oak Brook, IL 60521
International: (441) 442-876065
Support: (708) 368-0400
Fax: (708) 990-4641
markb@sdsi.com
<http://www.sdsi.com/>

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

FEATURE ARTICLE

**Doron Hendel &
Rowena Turner**

The Embedded Sun Part 2: Exploiting the Microkernel

Doron and Rowena are quick to show us that using the Chorus OS is simple. It lets developers write applications or use UNIX and other operating systems that work directly on top of the microkernel.

Ooday, hardware technology evolves rapidly, and the software industry is under intense pressure to create value-added open systems and cooperative computing environments. These conflicting forces complicate platform issues, creating major barriers.

This barrier is triggering two major shifts in the information technology industry. On the one hand, operating systems are supporting increasingly complex systems. On the other, an increasingly competitive market is spurring a business shift in the systems industry for faster time-to-market.

More and more systems engineers and programmers are finding that the Chorus Systems open microkernel technology, based on C++, provides them the best OS route to support newer processors like the microSPARCIIe (see *INK 69*).

The microkernel enables the move toward a more object-oriented approach and provides the most basic requirements. In effect, it enables developers to design such systems as UNIX and other operating systems on top of the microkernel. It also permits them to write applications directly on top as real-time embedded operating systems (RTOS).

THE NUCLEUS

The microkernel architecture is based on the Chorus message-based Nucleus, which controls communications within the operating system itself. The Nucleus implements a minimum set of generic operating services necessary to support a processor like microSPARCIIe with its local memory or a group of shared memory processors.

As shown in Figure 1, the Nucleus comprises four major components that provide local and global services. The supervisor dispatches interrupts, traps, and exceptions which the microSPARCIIe delivers. The real-time executive controls the allocation of processors and provides fine-grained synchronization and priority-based preemptive scheduling.

The memory manager manipulates the virtual memory hardware and local memory resources. Lastly, the inter-process communications (IPC) manager provides both synchronous and asynchronous message exchange and remote procedure call (RPC) facilities in a location-independent fashion.

Table 1 shows the basic abstractions implemented and managed by the Nucleus. Unique identifiers (UIs) are names generated for all actors, virtual memory segments, and IPC addresses such as ports and port groups. These UIs are generated so they are unique in both time and space. Over the life of the system, no two objects in a distributed Chorus OS ever use the same UI.

The IPC manager in the Nucleus delivers messages between actors on the same site. However, a network manager external to the Nucleus keeps track of ports throughout the operating system. An actor provides an execution context for one or more threads. It is attached to a site and all the site's resources. Chorus can allocate the individual threads within an actor to different processors on a multiprocessor site.

Threads of one actor send messages to threads of another by means of ports (queues attached to actors). Sending messages via ports rather than directly to the other thread decouples communication from execution.

Communication thus becomes transparent with respect to distribution. One thread doesn't need to know where another is executing to communicate with it. A thread can only belong to one actor, but a port can migrate from one actor to another, redirecting all messages to the new actor.

The site is the basic unit of computing hardware under Chorus. It consists of one or more processors, memory, and I/O devices. A site could be a whole computer or just a board in a rack. Each site runs one nucleus.

Lastly, a thread is the unit of execution and has the same meaning as it does in Windows NT and OS/2. Unlike a heavyweight UNIX process, a thread doesn't need a private address space. It only needs its own stack, and many threads can share the same address space. Under Chorus, that address space belongs to an actor.

The memory manager, real-time executive, and communications unit are all portable, while the other two parts of the Nucleus are machine dependent.

There are no interdependencies among these four components. Consequently, distribution of services provided by the Nucleus is virtually hidden. Local services deal with local resources and can be largely managed using only local information.

To provide distribution, Global services involve cooperation between Nuclei. Each microkernel cooperates with other microkernels to form a unified virtual machine, transparently spread over a set of tightly or loosely coupled processors.

Most operating systems don't provide loosely and tightly coupled setups. Here's what we mean by loosely as opposed to tightly coupled.

Traditional computers are tightly coupled. There's a motherboard with a CPU and a bus interface which communicates with such peripherals as disk drives and I/O devices. There's

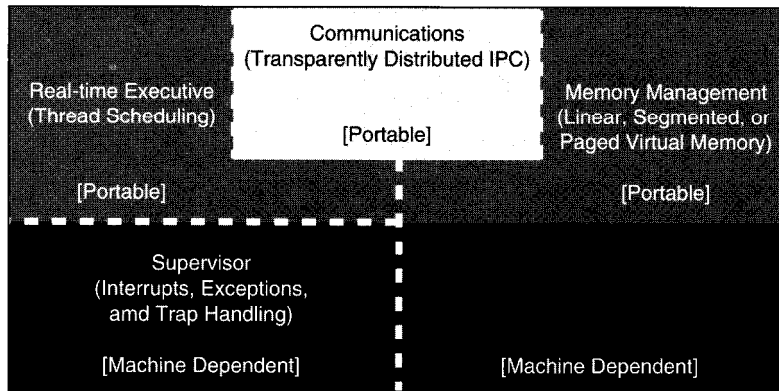


Figure 1—The microkernel architecture is based on the Chorus message-based Nucleus, which controls communications within the operating system itself.

also onboard memory to perform local transactions.

Conversely, a loosely coupled system enables these same components to be fully distributed. They can have n CPUs located throughout the system's topology. Also, disk drives can be totally and physically isolated from the CPUs. However, they are connected together via a cell communication method.

This loosely coupled setup is transparent to the programmer, regardless of whether it is a bus, communications protocol, or another means of communication. In this instance, the microkernel is the element making that connection.

An operating system based on Chorus is composed of a Nucleus and a set of system servers which cooperate in the context of subsystems (see Figure 2). This overall organization provides the basis for an open operating system.

It can be mapped onto either a centralized or distributed configuration. The Nucleus isn't the core of a specific operating system. Rather, it provides generic tools to support a variety of host subsystems which can coexist on top of it.

This structure supports application programs, which already run on an existing operating system, by reproducing the its interfaces within a subsystem. The idea of separating an

operating system's functions into groups of services provided by autonomous servers is key to the Chorus philosophy.

In monolithic systems, these functions are usually part of the kernel. Separation of functions increases modularity and therefore the portability of the overall system.

The Nucleus enables you to perform a fast port to a new hardware architecture like micro-SPARCIIe. To port a full microkernel-based operating system to a new hardware platform, you only have to deal with the two machine-dependent portions.

If you design the servers on top of the microkernel system programming interface (SPI), the hardware and its topology are transparent to you.

OPERATING SYSTEM EXAMPLE

Chorus/MiX (Figure 3) is an example of how a UNIX system would be built on top of the microkernel to offer either a standard System V.3.2 or V.4 interface with multithreading and real-time features.

All traditional operating system kernel services such as file services, device management, UNIX communication functions, and even device drivers are implemented outside of the microkernel as separate programs running in independent, distributed, cooperating servers.

As shown in Figure 3, you can use the system programming interface to program such objects as the process, file, and streams managers. Then, porting this operating system to a hardware platform is relatively trivial.

With Chorus/MiX, communications services provided by the microkernel give subsystem servers the

Table I—There are several basic abstractions the Nucleus implements and manages.

Unique Identifier (UI)	Global Name
Actor	unit of resource allocation
Thread	unit of sequential action
Message	unit of communication
Port, Port Groups	unit of addressing and (re)configuration basis
Region	unit of structuring of an Actor address space

ability to cooperate without needing to know where they are being executed in a distributed or cluster configuration.

Operating-system (re)configuration decisions can be performed dynamically (i.e., while it's still running). Using this operating system in a SPARC platform like microSPARCIIe is relatively simple.

You'd use Chorus application-programming interfaces (APIs)

to program to the SPI level. Chorus provides a variety of APIs such as POSIX real-time and multithreading APIs. That's all you need from a system programming level.

As for programming the software for special functions like the PCI bus interface on the microSPARCIIe, you only need to change the microkernel.

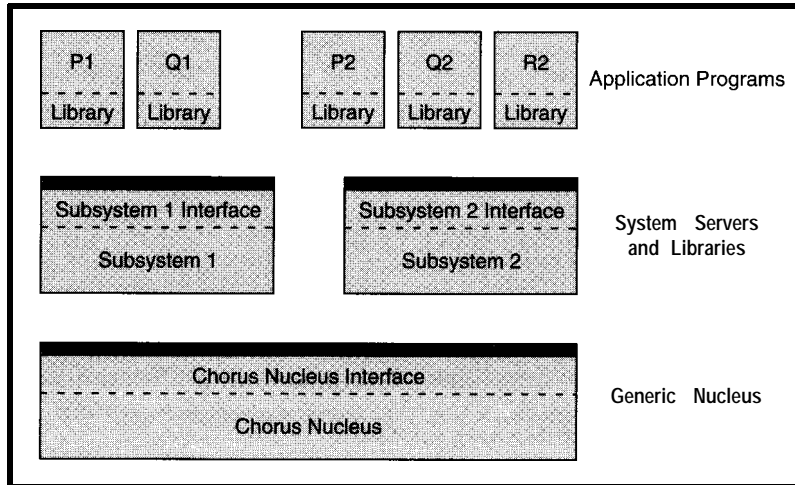


Figure 2—An OS based on Chorus is composed of a Nucleus and a set of system servers which cooperate in the context of subsystems and provide the basis for an open OS. It can be mapped onto a centralized or distributed configuration. The Nucleus is not the core of a specific OS, but it provides generic tools designed to support a variety of host subsystems which can coexist on top of the Nucleus.

All you have to do is include a device driver to support the PCI bus interface function. Otherwise, the SPI is transparent and oblivious to the fact that the hardware includes SBus or PCI bus interfaces. The APIs on top of the microkernel won't change.

In essence, you've programmed an architecture that supports an SBus

today and a PCI bus tomorrow, which is highly cost effective since you won't need to reprogram your applications. The work needed to support PCI is already done and isolated in the microkernel.

The microkernel also provides a single system image (Figure 4) for distributed systems that involve a number of boards with a microSPARCIIe and microkernel on each. This single system image is created for the system programmer and the administrator who manages such systems. It provides an efficient solution for the high-end embedded market that uses more than one CPU per application.

MODULARITY

Developers are modularizing the Chorus microkernel into an assortment of building blocks. These building blocks enable you to structure a scalable microkernel. Thus, you can use the same operating system from a single vendor to deploy a variety of SPARC-based platforms.

Systems range from high-end elements or servers which require a full, complete UNIX on top of the microkernel, to a low-level hardware controller which needs a simple real-time executive.

In distributed heterogeneous systems, the system developer has the flexibility to run the same basic operating system on low-end controllers all the way up to the mid- and high-range embedded systems and maybe even on the servers.

The main functional building blocks of a scalable microkernel (see Figure 5a) are executive, memory management, and IPC functions. The executive modules are composed of a core executive plus modules which provide it with basic policies or additional functionality.

The scheduler (SCHED) is mandatory. The core executive doesn't imple-

Under the hood of a stock
Cimetrics RS-485 Protocol Stack ...

... plenty of software power to drive
an embedded microprocessor network!

- > Performance
- > Robust Implementation
- > Payback
- > Reliability

Exactly what you would expect from the leader in microcontroller networks for RS-485, BACnet™, Ethernet™ and ARCNET™ environments. Have your RS-485 network up and running smoothly in just hours – at a fraction of the development and support cost – with full support from the experienced team at Cimetrics. Please call, or FAX us at 617-350-7552.



Cimetrics • Boston, MA • 617-350-7550

ment any scheduling policy. It relies on an additional scheduler module that implements the scheduling policies appropriate to a given hardware and software configuration. Scheduling policies range from basic priority-based scheduling to user-defined scheduling classes.

The scheduling modules handle symmetric multiprocessing when available. Modules implementing alternative scheduling policies such as deadline or fair share will be provided.

Fault management (FAULT) is mandatory as well. A fault-management module implements the policy applied when a system fault occurs. Functionality ranges from simple panic-handling to complete exception handling. Some modules integrate an interactive kernel debugger or memory-dump generators.

Timer management (TIMER) is also mandatory. The time-management service provided by the core executive is limited to the recording of the system boot time and the management of timeouts.

A timer-management module is responsible for programming hardware clocks and timers. It provides clock ticks to the core executive. It may also provide time-of-day management and export the API for getting access to fine-grain logical timers.

Interrupt management (INT) is optional. The interrupt-management services (IMS) of the core executive export a simple interface, enabling applications to handle interrupts at the lowest level and thus implement their own interrupt-handling strategy (application-based interrupt management, or ABIM).

A default interrupt-management module (kernel-based interrupt management, or KBIM) also uses the core-executive interface and implements high-level interrupt-handling services. It handles pre- and postprocessing of interrupts and deals with interrupt nesting, the interrupt stack, and interrupt priorities. The KBIM enables porting of most applications without the need to develop a specific ABIM.

Synchronization (SYNC, MBOX) is optional. A set of modules provides

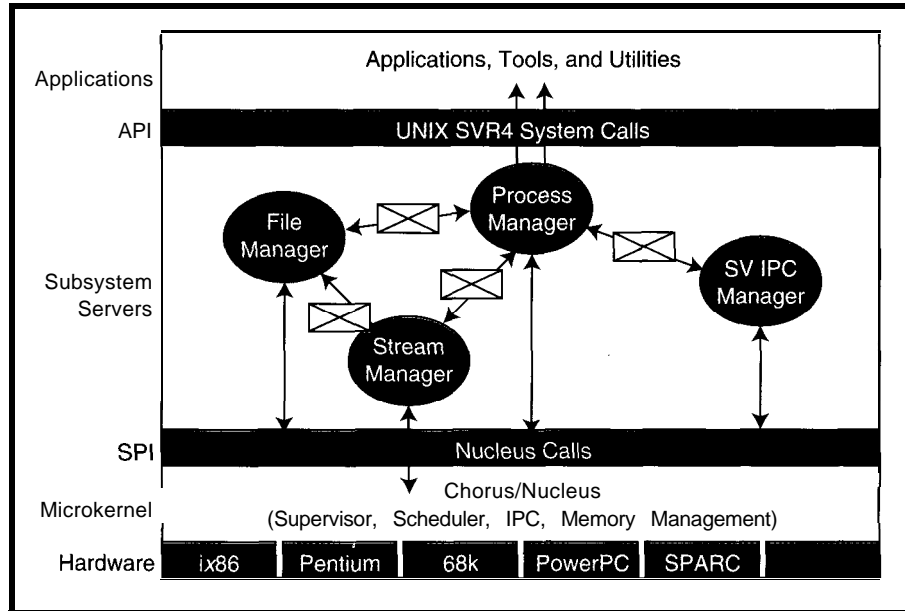


Figure 3—Chorus/MiX shows how a UNIX system would be built on top of the microkernel to offer either a standard System V.3.2 or V.4 interface with multithreading and real-time features.

various flavors of synchronization services, such as spin and masked locks, mutexes, semaphores, event-flag lists, and mailboxes.

Mailboxes (MBOX) are also optional and provide a private communications

environment between one or several actors of a given application.

A memory management module is mandatory. Each one must implement the basic memory management API exported by the core executive. Cur-



rently, four memory management modules implement four different levels of functionality.

Flat memory (FLM) implements a linear address space with the kernel and all actors running in the same (physical) address space and with no memory protection. It provides simple memory allocation services.

Protected memory (PRM) provides memory between application actors. It may or may not use address translation, depending on hardware support.

Full virtual memory (VM) supports standard virtual memory with an extended Chorus/Nucleus interface. It supports swapping in and out on secondary devices.

This module has been specifically designed to implement distributed UNIX and UNIX-like subsystems on top of the kernel. It exports a generic interface to implement shared memory functionality over a network. One coherency mapper can be present on each site to implement application-specific memory sharing strategies.

Several optional modules offering IPC services are also provided. They

offer subsets of the original Chorus/IPC services and are referenced as LIPC (Local IPC), SPIC (Simple IPC), and IPC [full IPC]. The set of services ranges from efficient local communications mechanisms to sophisticated network-based mechanisms with enablers for dynamic reconfiguration and high availability in a consistent fashion.

In Figure 5b, you can see the microkernel scalability factor of taking, for example, a Chorus minimum configuration, using it on a portable phone, and then sliding it through set-top boxes, switching platforms, interactive TV servers, ATM servers, all the way up to fault-tolerant UNIX servers.

This extensible and open OS translates into cost-effective development cycles for system developers. They only have to become familiar with a single SPI and IPC. There's no need to learn any other operating systems or interfaces to provide applications across a variety of platforms.

CONCLUSION

More and more, industry is focusing on object orientation. Chorus object-oriented layer (COOL) is an ongoing research project. Development efforts include COOL-object request broker (ORB), which is an OMG CORBA-compliant piece of software. CORBA (Common Object Request Broker Architecture) is a standard developed by the Object Management Group (OMG).

ORB, an object-oriented layer tailored to work on top of the Chorus OS, enables you to provide real-time applications using object-oriented techniques.

These applications are totally transparent and independent from the hardware underneath and (in some cases) from the microkernel itself. They provide a unique and easy program-

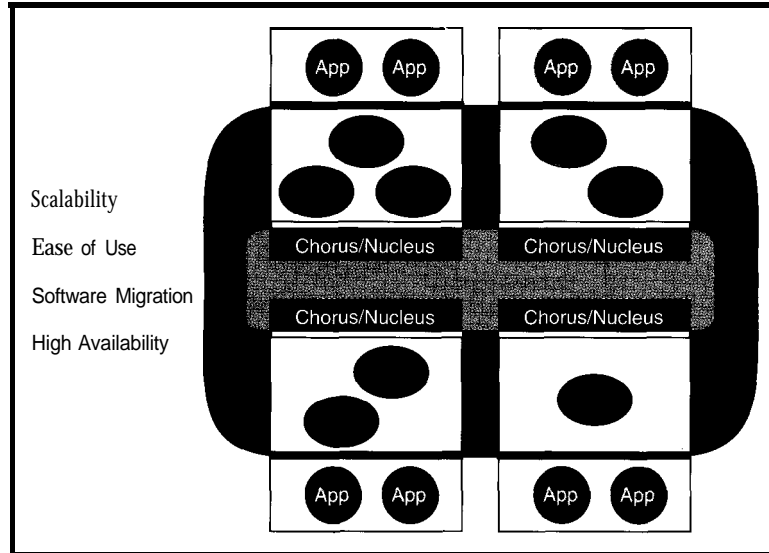


Figure 4-The microkernel provides a single system image for distributed systems involving a number of boards with a microSPARCle and microkernel on each.

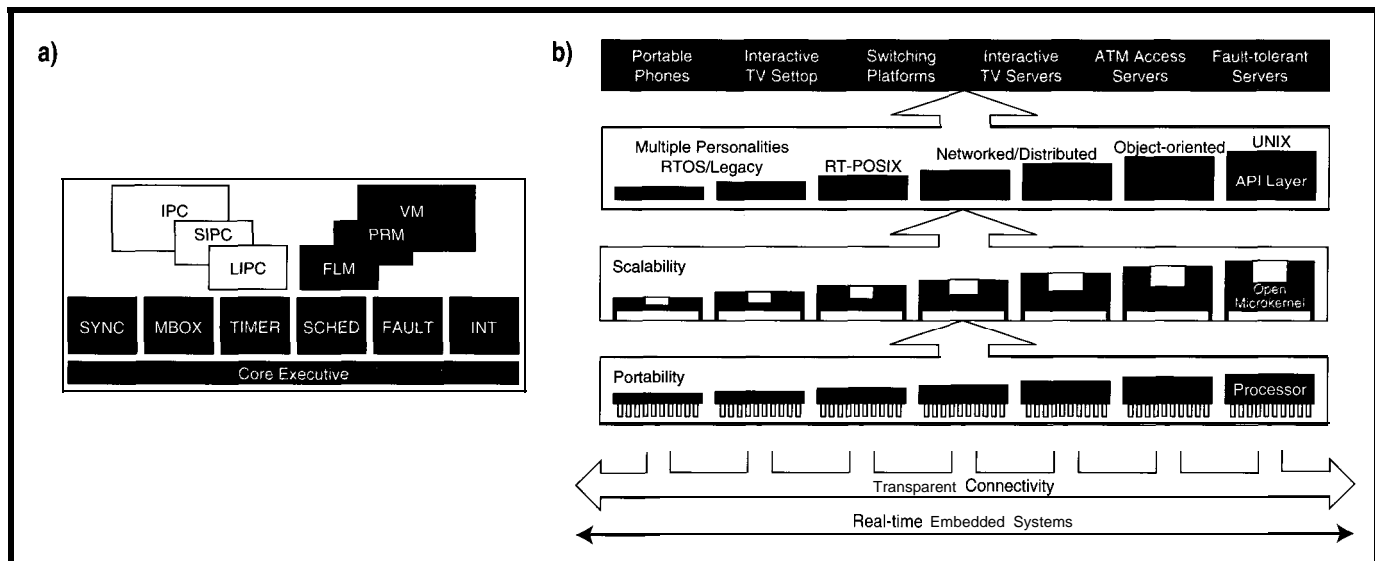


Figure 5-a) The main functional building blocks of a scalable microkernel are executive, memory management, and interprocess communications functions. b) This extensible and open OS translates into cost-effective development cycles for the system developer.

ming interface on top of embedded systems through the ORB.

COOL-ORB also runs on a traditional operating system like Solaris or Sun. So, you can have applications which interact between SPARC workstations and SPARC embedded systems. You get the cost effectiveness of programming to one IDL (Interface Definition Language) interface, even if you're running it on a SPARC workstation, on top of Solaris, or on top of Chorus in an embedded system. □

Doron Hendel is a project manager and system engineer at Chorus. He has worked on various embedded systems with real-time emphasis for telecommunications and networking applications as well as numerous networking protocols. Doron may be reached at doron@chorus.com.

Rowena Turner is marketing manager for operating systems at Sun Microelectronics. She previously held positions in marketing and engineering at Tandem Computers and Hewlett-Packard. She has supported operating systems such as Guardian and MPE, as well as real-time operating systems including Wind River, Lynx, and Chorus. Rowena may be reached at rowena.turner@eng.sun.com.

SOURCES

microSPARCIIe

SPARC Technology Business
Sun Microsystems
M/S USUN03-305
2550 Garcia Ave.
Mountain View, CA 94043- 1100
(408) 774-8685
Fax: (408) 774-8769

Nucleus microkernel
Chorus Systems
1999 South Bascom Ave.
Campbell, CA 95008
(408) 879-4137
Fax: (408) 879-4102

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful



Your 1st Choice for C Compilers

Byte Craft's optimizing C compilers are fast and efficient. C extensions provide control over bit manipulations, I/O port and memory definitions, as well as support for direct register access and interrupts.

We respond to your C compiler needs.

Byte Craft Limited

421 King Street N., Waterloo, Ontario CANADA
Tel: (519) 888-6911 Fax: (519) 746-6751
BBS: (519) 888-7626 Email: info@bytecraft.com

PIC16/17Cxx

MELPS740

COP8

MC68HC08

Z8

MC68HC05



#118

Be a hero – put Team *Paradigm* to work on your next embedded x86 design and reap the rewards you deserve. *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.

So forget about making excuses and instead enlist *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, NY 13760
(607) 748-5966 /FAX: (607) 748-5968
Internet: 73047.30310 compuserve.com

©1995 Paradigm Systems, Inc. All rights reserved.

**TO BE
CONTINUED.**

#120

FEATURE ARTICLE

Willard Dickerson

Vehicular Control Multiplexing with CAN and J1850

Part 2: Application to Motorola Embedded Controllers

Willard looks at the Motorola controllers carrying CAN and J1850 protocols. After giving an overview of the chip's features, he presents the CPU, mux, and physical interfaces, and protocol handler with examples.

Oast month, I overviewed vehicular multiplexing before looking specifically at the CAN and J1850 protocols. This month, I'd like to go a step further.

As you can see in Table 1, a broad range of Motorola embedded controller families use the CAN and J1850 protocols. I'd like to focus on using the J1850 and CAN protocols on these controllers. I'll present an example for each vehicle protocol on each separate controller.

J1850 APPLIED

The J1850 protocol can be implemented in an embedded controller's hardware as a peripheral. It can be viewed as a slave peripheral configured through the microcontroller's software. It could also be conceptually implemented through a combination of basic microcontroller peripherals such as timers, register memory, software, and an external J1850 bus driver.

Motorola offers their BDLC-D J1850 peripheral on a 68HC708A48 64-pin QFP, 8-bit embedded controller chip with 48 KB of on-chip EPROM or OTPROM, on-chip programming firmware for use with host PCs, data security for programmable memory,

640 bytes of on-chip EEPROM, and 1.5 KB of on-chip RAM. The chip offers both SPI and SCI as well as a 16-bit, 6-channel timer-interface module (TIM); clock-generator module (CGM); and 8-bit, 16-channel ADC. Figure 1 shows an overview of the chip.

As you can see in Figure 1, the J1850 controller—also known as the Byte Data Link Controller (BDLC)—is connected to the main internal interface bus in addition to the external pins CL2RxD and CL2TxD. This bus provides an interface between the J1850 device and the CPU, registers, memory, and other internal devices. For example, the user can store control code for the J1850 device in the user program area.

The BDLC-D peripheral supports a standard J1850 in-frame response (IFR) and standard message lengths of a maximum of 112 bytes (see J1850 frame-format section of Part 1). These messages include the following restrictions:

- communication at 4x frequencies of 41.6 kbps are receive only. (Transmissions are restricted to lower frequencies.)
- break symbols can be received but not sent

In addition, the BDLC offers the user two main features not always offered on stand-alone parts: five main modes of operation and a programmable transmission delay. These modes include power off, reset, run, BDLC stop, and BDLC wait.

The main difference between the stop and wait modes is that the device is initiated by a transmission out of the stop mode. Operation is not guaranteed to be coherent since time is required for proper initialization. Conversely, if initiated by a transmission in wait mode, correct continued operation is guaranteed, although more power is consumed in this mode.

The programmable transmission delay provides the user with the ability to interface with a broad range of transceivers by compensating for various delays which are inherent in transceiver design and often vary between manufacturers. The default delay is 16

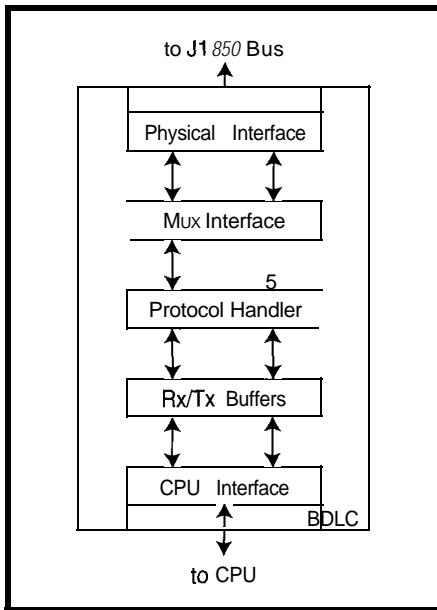


Figure 2—The BDLC consists of five major sections.

The handler is composed of a loopback multiplexer, state machine, and Rx and Tx shift registers. The loopback multiplexer selects the J1850 communication path between the physical interface and the digital or analog loopback mode, depending on how the bits in the BCR2 register are set.

The state machine controls all functions associated with performing the J1850 protocol, framing, collision detection, arbitration, CRC generation/checking, and error detection.

The Tx/Rx register section includes a shift register and shadow register for each transmit and receive. After data fills either the transmit or the receive shift register, it is copied to the corresponding shadow register and an interrupt or flag can be set so that the CPU can either read received data or load the next byte of transmit data.

The mux interface provides both bit decoding and encoding and digital noise filtering between the protocol handler and the physical interface.

The digital filter decodes and filters on a symbol basis. It performs successive samples of each bit received from the physical interface and determines if the bit is valid or invalid. If the symbol appears to be valid and not an anomaly, the

symbol is presented to the output of the filter. Otherwise, it is ignored.

Incidentally, the term “symbol” is used in vehicle multiplexing to refer to some shape on the bus that has significance to the corresponding protocol. You can use, for example, EOD or IFS symbols.

Digital encoding is done by structuring the messages into the standard J1850 formats, depending on how the bits in BCR2 are configured. In general, all messages are structured to include idle, SOF, header, data field, CRC, EOD, IFR (optional), EOF, IFS, and idle. The general structure is depicted in Figure 3.

The physical interface provides the J1850 bus signal’s wave shaping, driving, and digitizing of data. Motorola devices such as the 68HC705V8 include the physical interface on-chip, whereas others, like the 68HC08AS20, allow for an external physical interface of the user’s choice.

The BDLC provides a Variable Pulse Width (VPW) modulation signal on the J1850 bus. This interface converts a

Register Name	Address
BARD	\$003B
BCR1	\$003C
BCR2	\$003D
BVSR	\$003E
BDR	\$003F

Table 2—Five registers provide a means to control the BDLC.

standard nonreturn-to-zero digital signal to a VPW signal.

TRANSMISSION

This section outlines the basic software that initiates a simple J1850 transfer. The flowchart in Figure 4 shows the general steps involved in initiating a BDLC transmission.

The BDLC is initiated through its control registers located in the CPU interface section of the BDLC. Once

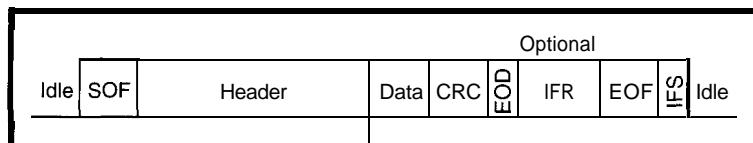


Figure 3—The BDLC supports J1850’s In-frame Response (IFR).

again, the BDLC operates without CPU intervention. The CPU is only used to configure control and access buffers.

Although details for programming the HC08 microcontroller can be found in the *HC08 Central Processing Reference Manual* [1], 1’11 go into some of the details here.

First, you configure the BARD register for onboard analog transceiver, normal polarity, and a delay of 9µs. The value in BARD is then equal to 11000000.

```
lda #$C0 Load A with config
          for BARD
sta $3b Store in BARD
```

Next you configure the two BDLC control registers BCR1 and BCR2. In BCR1, you need to disable the receiver until start of frame, use the integer frequency clock and the 2-MHz crystal, enable the interrupt, and stop the internal clocks during the wait mode. The value in BCR1 equals 10010011.

```
lda #$93 Load A with config
          for BCR1
sta $3C Store in BCR1
```

BCR2, which controls transmitter operations, drives the J1850 bus by the BDLC. Do not invoke the digital loop mode to drive the bus normally. You receive and transmit at 10.4 kbps. Set the normalized bit to 0 when the in-frame response (IFR) ends in a CRC byte. Transmit an EOD symbol and a type 3 IFR (010) transmit multiple byte IFR with CRC. BCR2 should therefore be set to 00011001.

```
lda #$1A Load A with config
sta $3d Store in BCR2
```

Third, you need to observe BVSR—the interrupt vector register—for an interrupt. BVSR is a read-only register in which bits 5-2 indicate the source of a pending interrupt’s priority, where the lowest priority is 0 and the highest is 8 (see Listing 1).

The user can provide the location of the ser-

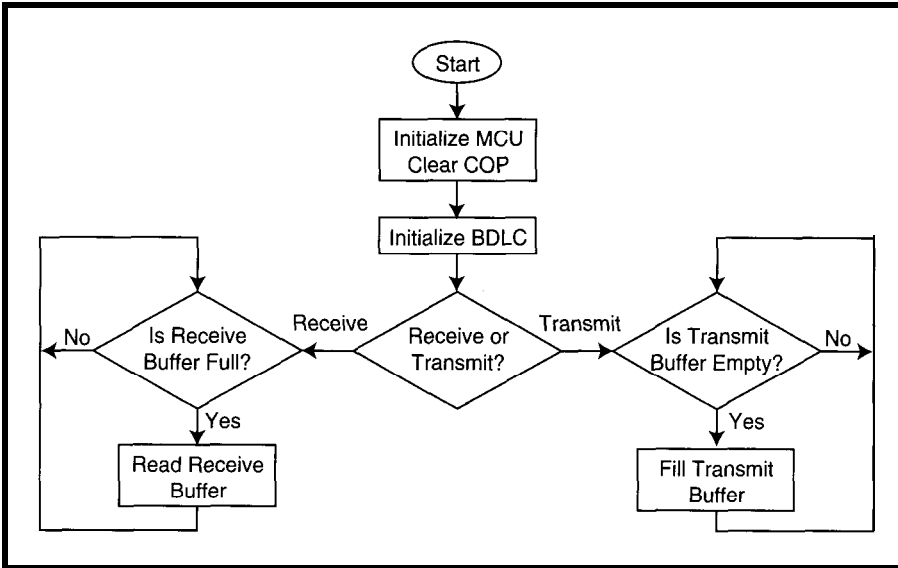


Figure 4—The CPU views the BDLC as a device that requests register access on certain interrupts.

vice routine at the address provided by the vector. These interrupts can be caused by an IFR byte received, full or empty buffer conditions, loss of arbitration, CRC errors, symbol invalid, out of range, or wakeup.

For interrupt routines located above page 1 (i.e., greater than SFF), indirect

or indexed addressing should be used. Otherwise, the processor doesn't arrive at the expected location.

Last, you need to access BDR, which is the BDLC data register. This step depends on the type of interrupt that was received. If a request is made to access the BDLC data buffer, it can

be done by a read or write to the BDR register:

```

1da #0f Load A with BDLC data
sta $3f Store in BDLC
  
```

Keep in mind that for a full message to be built, 1–8 data bytes can be sent. Therefore, a single byte at a time is written to this register.

MOTOROLA AND CAN

The CAN protocol can also be implemented in an embedded controller's hardware as a peripheral or viewed as a slave peripheral configured through the microcontroller's software. It can also be conceptually implemented through a combination of basic microcontroller peripherals such as timers, registers, memory, software, and an external CAN bus driver.

Motorola's MC68HC05X16 embedded controllers offer low-cost (less than \$10 in quantities over 10,000), effective hardware to implement the CAN transfer layer, which represents the kernel of the CAN bus protocol.

<p>RECHARGEABLE BATTERY 12 VOLTS 9 A/H</p> <p>Here's a great deal on new 12 volt 9 amp/hour rechargeable lead acid batteries. These batteries were recently manufactured and prepped for use in a product which hasn't yet come to market. The OEM didn't want to hold them in stock, so we got them at a greatly reduced price. They are 5.9" x 2.55" x 4.42" high and have a strip of foam padding stuck to one side. They have standard 1/4" quick-connect/ solder terminals. Wt: 6 lbs.</p> <p>CAT # GC-129 \$28.50 each 8 for \$208.00</p> 	<p>SUB-MINIATURE SNAP ACTION SWITCH</p> <p>CHERRY DG1C-BJ07 S.P.D.T. Body size: 0.51" long X 0.255" high X 0.255" wide. Lever with ridged end simulates roller action. Rated 3 amps @ 125 Vac, 2 amps @ 30 Vdc. PC pins on 0.2" centers. Large quantity available. CAT# SMS-133</p> <p>2 for \$1.00 100 for \$45.00 1000 for \$350.00</p> 	<p>Visit Our World Wide Web Site... http://www.allcorp.com/</p> <p>RANGE 20 METERS (65.6 FEET) INDOOR/OUTDOOR PHOTOELECTRIC SWITCH</p> <p>SICK Optex # WS/WE260-M230 Invisible infrared beam and switch is useful for a variety of security, safety and sensing applications such as perimeter alarms, electric gate or garage door controls or conveyor line sensing. Modulated beam covers a 20 meter distance even with poor visibility. IP-66 enclosure rating makes it dust proof and splash resistant, suitable for most outdoor conditions. Light/dark switching selector. Sensitivity adjustment control. LED power and sense indicators. Two position cable entry gland (90 deg. apart) accepts 1/2" threaded conduit and a variety of wire sizes. Glass fiber-reinforced plastic housings are 3.05" x 2.5" x 1". Supply voltage: 22-240 Vac. Solid State Output: 0.5 amp (must have load to operate). Includes mounting brackets and hardware UL and CSA listed.</p> <p>CAT # PES-4 \$40.00 each</p> 
<p>PHOTO RESISTOR</p> <p>Silonex # NSL 4562 Cadmium Sulfide photoconductive cell. 22 ohms in bright light. 2 meg ohms dark. Plastic encapsulated. CAT # PRE-12 75¢ 0.29" x 0.26" x 0.07". 1.5" long wire leads.</p> <p>10 for \$6.00 • 100 for \$50.00</p> 	<p>PHOTOVOLTAIC DETECTOR</p> <p>Miniature silicon solar cell converts light impulses directly into electrical charges which can easily be amplified, using a transistor for example, to activate a control mechanism. Unlike a conventional photo diode or transistor, it generates its own power and does not require any external bias. Generates 0.4 vdc in moderate light. Silicon cell is mounted on a 0.31" x 0.23" x 0.07" thick plastic carrier and has pc leads on 0.2" centers. Large quantity available. CAT # PVD-2</p> <p>75¢ 10 for \$6.50 • 100 for \$50.00</p> 	
<p>ALL ELECTRONICS CORPORATION</p> <p>ALL ELECTRONICS CORP. • P.O. Box 567 • Van Nuys, CA 91408-0567 FAX (818)781-2653 E-Mail -allcorp@allcorp.com 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.</p> <p>ORDER TOLL FREE 1-800-826-5432 Visa, Mastercard, American Express or Discover accepted</p> <p>CALL, WRITE, FAX or E-MAIL for a FREE 64 Page CATALOG Outside the U.S.A. send \$2.00 postage.</p>		

μs with timing adjustments of 9-24 μs in steps of 1 μs.

The BDLC consists of five major sections: CPU interface, Rx/Tx buffers, protocol handler, mux interface, and a physical interface (see Figure 2). See Motorola's CISC reference manual, 'HC08 Central Processing Reference Manual, for more hardware and software details about the M68HC08 CPU [1].

CPU INTERFACE

The CPU interface facilitates communication between the J1850 controller and the embedded controller. This interface consists of five user registers that enable the data and status from the J1850 device to be communicated. These registers are shown in Table 2.

The BDLC Analog and Round Trip Delay (BARD) register provides three things: user selectable on-chip or off-chip analog transceiver enable, receiver pin polarity, and compensation delay.

The BDLC Control Registers (BCR1 and BCR2) control global features and specific transmitter functions, respectively. The globally controlled features available through BCR1 include disabling the receiver (communication can be ignored then started by an SOF or break symbol), select a binary (1.048567 MHz) or integer (1.00 MHz) frequency (system clock), clock-rate selection, interrupt enable, and wait clock mode. The wait clock mode selects either stop or run internal clocks during a CPU-initiated wait mode.

The second control register (BCR2) controls the BDLC transmitter. Six features can be controlled:

- analog loopback mode-selects whether transmissions loop back from the analog physical layer (if present on the chip)

Controller	ROM	EPROM	J1850	CAN
68HC05V8	N	Y	Y	N
68HC05V7	Y	N	Y	N
68HC708AX48		Y	N	
68HC708AS20	Y	N		
68HC05V12	Y	Y	Y	N
68HC05X4	Y	Y		Y
68HC05X16	Y	Y	N	Y
68HC05X32	Y	Y	N	Y
68HC08AZ16		N	Y	
68HC05X32	Y	Y	N	Y
68332*			N	Y

* based on the TOUCAN stand-alone module

Table 1-Vehicular multiplex controllers can be found on Motorola 8- and 32-bit embedded controller families.

or are sent to the J1850 bus. The analog physical layer is not commonly integrated on vehicle multiplex devices.

- digital loopback-selects whether transmissions loop back through digital loopback path or the external bus path
- receive 4x enable-selects whether the receiver operates at 10.4 kbps or 41.6 kbps
- normalized bit format-defines the start of the IFR and allows two forms for the normalization bit with active short and active long periods. The variation determines what type of response is received during the IFR portion of the frame.

The SAE preferred method is to use the active short bit to indicate the CRC is included and the active long

bit to indicate CRC is not included [2]

- append EOD-determines whether to append the 8-bit EOD symbol to the end of an IFR transmission

- response type-the final three bits in the register determine the type of in-frame response being sent. Three in-frame response formats are available.

The 8-bit BSVR register provides an index offset directly related to the BDLC's current state. This offset can be used

with a user-supplied jump table to rapidly enter an ISR, thus minimizing the CPU overhead required to service interrupts. No duplicate state machine is then needed in software.

The BDR register's function is two-fold. It passes data to be transmitted from the CPU to the J1850 bus and communicates the J1850-received data to the CPU. Data is transferred to this register one byte at a time. Writing and reading this register is allowed after the Transmit Data Register Empty (TDRE) or Receiver Data Register Full (RDRF) condition has occurred, respectively.

The Rx/Tx Buffers are composed of the storage elements for data received from and transmitted onto the J1850 bus. A single byte of storage is available on the BDLC. In contrast, the

Message Data Link controller (MDLC) found on Motorola's 68HC705V8 allows for buffering of the entire message, which can include several bytes.

The Protocol Handler encodes and decodes data bits and special message symbols during transmission and reception. It includes framing, collision detection, arbitration, CRC generation/checking, and error detection. The handler conforms to the SAE J1850 Class B Communications Network Interface standard.

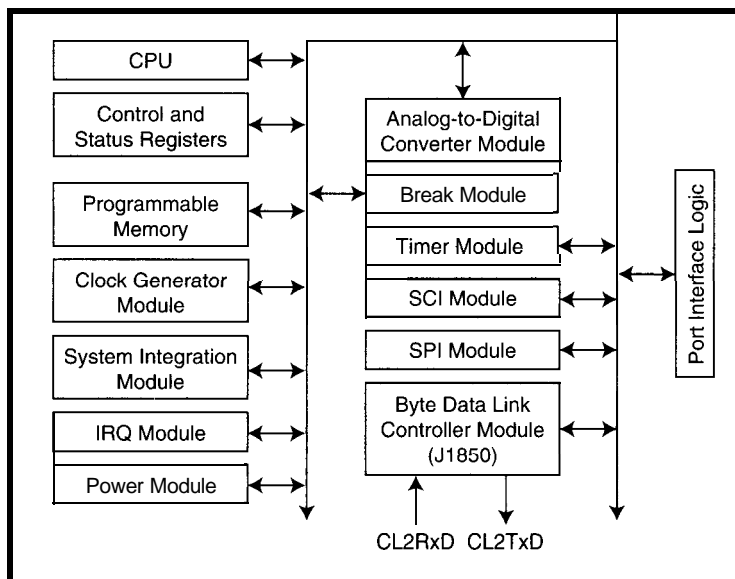


Figure 1-Motorola's 68HC708RAX48 offers a broad variety of peripherals integrated on a single chip.

Listing 1—The user can enter a jump table on receiving a BDLC interrupt

```
Bdlc_int ldx #3f      Acquire the state vector number
          jmp jmptab. x Enter service routine (must end in "RT ")

jmptab   jmp bdlc_int0 Interrupt condition #0
          nop          aligns jmps on 4-byte boundaries
          *           so the value of BVSR is intact
          jmp bdlc_int1 Interrupt condition #1
          nop
          jmp bdlc_int2 Interrupt condition #2
          *           nop
          jmp bdlc_int8 Interrupt condition #8
          end
```

Motorola refers to this hardware implementation of CAN as the MCAN.

Motorola denotes different feature sets they support on CAN modules with different names. Existing modules are the Motorola CAN (MCAN) and the 2.0B CAN (TOUCAN). A block diagram of the 'HC05X16, which incorporates the MCAN module, is shown in Figure 5. The MC68HC-05X16 offers a host of memory and peripheral devices as you can see.

The MCAN is fully compatible with CAN up to the message layer (also known as the transfer layer). The functional difference lies in how messages are handled in the

object layer. In a full CAN implementation, dedicated hardware handles messages by prioritizing messages, providing acceptance filtering, and buffering groups of messages.

Conversely, the MCAN offers complete message handling (filtering, buff-

ering, and prioritization) for transmitting or receiving messages on a message-by-message basis. A block diagram of the sections of the MCAN is provided in Figure 6.

MCAN logic can be partitioned into two related sections: microprocessor and bus line. The first partition relates to the transfer, object, and application layer while the second partition relates to the physical layer.

In microprocessor-related logic, four things can happen:

- messages are arbitrated and detected by the transfer layer
- messages are buffered, filtered, and prioritized by the object layer
- the embedded controller's CPU communicates through registers and common 'HC05 bus-handling controls
- the CPU provides application code from among 15118 bytes of user ROM

With bus-line-related logic, the following events can occur:

PUT YOURSELF IN THE

DESIGN CONTEST WINNER'S CIRCLE

Now is the time for you to start thinking about your entry in the 8th Annual Circuit Cellar Design Contest.

Entering is easy, just contact Rose at:

Circuit Cellar Design Contest
4 Park Street
Vernon, CT 06066
Tel: (860) 875-2199 Fax: (860) 872-2204

You'll be sent an official Entry Form and a complete set of rules.

All entries must be received by August 2, 1996.

You may enter as many projects as you wish, but each entry **must** be accompanied by a separate Entry Form.

Sponsored in part by Dataman Programmers Inc.

- the bits are sampled by the bit-timing logic to filter noise or invalid bits. Logic determines their validity
- CPU control register data configure the message type, length, acceptance method, and which flags to recognize
- CAN messages are received and sent through the physical interface
- line interface translates bit levels to the CAN bus line's recessive and dominant levels via the physical layer
- the error management system determines if the received message is valid by examining the acceptance registers and message identifiers
- bitstream logic encodes a message in standard CAN format

MCAN logic performs CAN transmissions with minimum CPU intervention through memory-mapped integration of 10 control register bytes, 10 transmit buffer bytes, and 10 receive register bytes.

Receive and transmit buffers are duplicated internally to minimize the number of transmissions lost due to buffer-full conditions which occur during a CPU access or transmission. CPU intervention of transmissions is essentially restricted to the configuration of control registers and responses to status or interrupts by reading a status register or accessing transmit or receive buffers

The control registers can selectively configure CAN transmissions and the overall operation of the MCAN module.

Each receive and transmit buffer consists of an 11-bit identifier, a remote transmission request bit, a 4-bit data code, and 8 data bytes. These control registers provide CAN with the necessary ingredients [not including the data located in the receive and data buffers] to build messages that

conform to one of the four basic frames. A combination of specific frame bits and conditions for frame fields are provided in these registers.

After the control registers are set up and data is provided to the appropriate buffers, the MCAN independently (i.e., without CPU intervention) performs all CAN bus interactions.

These interactions include physical bit conversion (actual voltage levels of recessive and dominant bits), reception filtering of each bit (selection of one- or three-time sampling), and error detection and action (responding to a bus error by waiting 128 successive occurrences of a sequence of 11 succes-

of the MCAN's input compactor), depending on the fault. For example, it can occur if a short is detected between the two bus lines or between one of the bus lines and ground, battery voltage, or some other potential. The faulty driver transistors should be switched off. The Rx0 and Rx1 control bits in the MCAN control register should therefore be configured to passive in this mode.

Again, although you should refer to Motorola's *Microprocessor, Microcontroller, and Peripheral Data* [3] for more detailed information on programming the Motorola's 'HC05 family of 8-bit microcontrollers, I've provided

guidelines for a simple CAN communication transfer using the MCAN:

- clear the COP (i.e., the watchdog timer)
- configure the 10 MCAN control registers to reflect the type of transmission desired
- access the MCAN data buffer on an interrupt, if the receiver flag indicates full, or the transmitter flag indicates empty

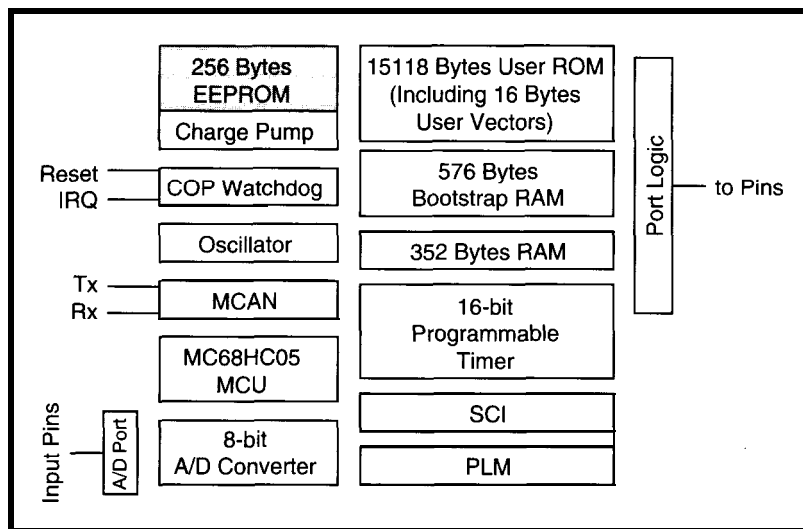


Figure 5—The MCAN module is among many peripherals included on Motorola's 'HC05X16 8-bit embedded controller.

sive bits before clearing the bus status flag, and resetting the read and write error counters).

Other interactions involve floating the bus while in sleep mode, encoding messages (which includes any of the four basic frame groups discussed last month), arbitration (giving highest priority to the CAN unit with the lowest numerated message identifier), CPU interrupt initiation, transmission abortion, acceptance filtering (through a commonly used 8-bit mask), and limited single-wire operation. (Single-wire operation can be done through a special software procedure not discussed in this article.)

The single-wire operation can occur automatically on a limited basis (i.e., a reduction in the common-mode range

Here is an example of how to configure the MCAN control registers located between \$20 and \$29. First, you need to configure CCNTRL. The controller is put in slow mode with the overrun interrupt disabled, the error interrupt enabled, the transmit interrupt enabled, and the receiver interrupt enabled. As well, with no reset request, it operates normally. The value in the CCNTRL equals 01001110.

```
lda #$4E Load A with CCNTRL
      config
sta $20 Store in CCNTRL
```

Notably, CCNTRL bit 7 must remain zero to prevent the transmit and receive buffers from being mapped out of memory.

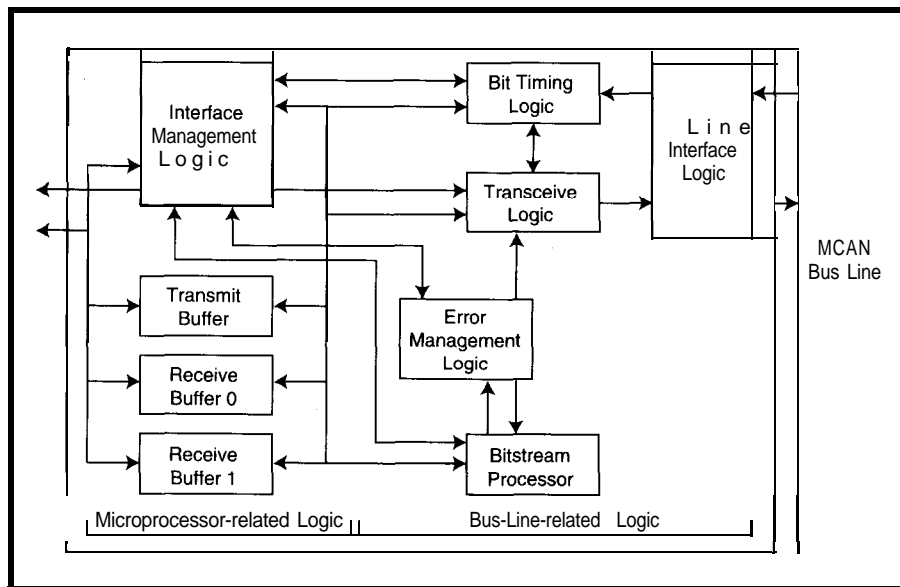


Figure 6—The MCAN can be viewed as two sections: microprocessor logic and bus-line logic.

The next step involves configuring the CCOM register. To do this, Rx0 is connected to the input compactor, while Rx1 is disconnected and the compactor is connected to $V_{dd}/2$. Rx0 and Rx1 are compared with $V_{dd}/2$ during sleep mode.

The MCAN sleeps if no pending interrupts issue a wakeup interrupt. No action is taken for the clear-over-run status. If the receive buffer is available to the MCAN, transmission of any pending message is aborted, and a remote frame is transmitted to request data. The contents of CCOM equal 01110111.

**lda # \$77 Configure
sta \$21 CCOM control reg**

Third, the MCAN CSTAT is a status register that reflects the bus state, error status, transmit status, receive status, transmission complete status, transmit buffer access, data overrun, and receive buffer status. CSTAT can be read by the CPU.

Next, the MCAN interrupt register, CINT, is read. This status register contains the wake-up flag detect, overrun interrupt detect, error interrupt detect, transmit complete detect, and receive buffer available detect flag.

**lda \$23 Load contents of in
interrupt reg**

You configure the MCAN's code-acceptance register, CACC, only if the request for reset in the CCNTRL register is set. As you can see, this bit was cleared. The CPU cannot access this register if the bit is clear.

However, this register contains the acceptance bits to compare the eight most-significant bits of the data identifier. If these bits are accessed and an acceptance occurs, then the buffer status bit is set to full and the receiver interrupt bit is set (provided RIE is enabled).

You configure the acceptance mask register, CACM, only if the request for reset is set like it is for the above register. This register specifies the corresponding register bits that are relevant for acceptance filtering.

Configure the MCAN bus timing register 0, CBT1 (at \$26), if the request-for-reset bit is set. This register contains two field types: synchronization jump-width bits and baud-rate prescaler bits.

The first field of 2 bits defines the maximum number of system clock cycles that can be sized. This field is stretched or compressed to achieve resynchronization of data transmissions on the bus.

The second 6-bit field determines the MCAN clock time which is used to build individual bit times. The second field is used by the bit-sample logic.

You then configure the MCAN timing register 1, CBT1 (at \$27), if the request-to-reset is set. This register determines the number of samples of the serial bus to be taken per bit time.

Next, configure the MCAN output control register, COCTRL (at \$28), if the request-to-reset is set. This register determines the type of output mode in which the MCAN operates, whether it be biphasic mode, not used, bitstream transmitted on both Tx0 and Tx1, normal mode 2, Tx0-bit sequence, or Tx1 bus clock.

The six remaining bits in this register configure the output drivers as low, high, or floating.

The next two registers are located in the transmit buffer section at \$2A and \$2B. These registers are named the buffer identifier register (TBI) and the remote transmission request and data length code register (TRTDL). The first register provides the most-significant eight bits of the message identifier. The lower three bits are found in the adjacent register at \$2B.

The second register completes the 11-bit identifier with the least-significant three bits. It can provide a remote transmission request and can configure the transmission frame from 0 to 8 bytes. Four bits configure a frame. For a remote frame, this field is ignored, forcing the number of bytes to be 0.

Similarly, at \$34 and \$35, two receiver identifier and data length registers are located. They can be set up like the transmitter registers.

Finally, you can access the data buffer to read or write data. A full or empty flag, respectively, is set.

After these steps are taken, simple transmissions can occur. Retrieving and loading the data buffers are done on a byte basis.

CONCLUSION

This paper provides an overview of vehicle multiplexing and discusses implementations for the CAN and J1850 protocols in Motorola embedded controllers.

Today's automobile manufacturers continue looking for ways to reduce cost while adding value. CAN and J1850 protocols lay a foundation for achieving that goal. The built-in sup-

port in
lers ge
stand:
Ch
aboar

Speci
Moto
review
work,
vehic
appe

Will
leade
trolle
croco
hard
bit er
react

RE
[1]
[2]

Ne
dc
te
pc
di
po
ni
nc
m

Pric
Fec

port in Motorola's embedded controllers goes a long way to furthering the standardization of these protocols.

Check out the references to jump aboard the CAN and J1850 buses. □

Special thanks to Nigel Allison of Motorola for his effort and time in reviewing this two-part article. His work, dedication, and expertise in the vehicle multiplex arena are greatly appreciated.

Willard Dickerson is a design project leader for 5xx RISC-embedded controllers in Motorola's Advanced Microcontroller Division. He develops hardware and firmware for 8- and 32-bit embedded controllers. He may be reached at willd@nambe.sps.mot.com.

REFERENCES

- [1] Motorola, *HC08 Central Processing Reference Manual*, Manual CPU08RM/AD, 1993.
- [2] "SAE Recommended Practice J1850 Class B Data Communi-

cation Network Interface," 1993.

- [3] Motorola, *Microprocessor, Microcontroller, and Peripheral Data*, vol. 1, Data Book DL139, 1988.

ADDITIONAL REFERENCES

Embacher, M. "CAN Networking Solution for Vehicle Body DC Motor Control," *Automotive Engineering* 20:2, 1995.

Carman, D. et al. "Serial Bus Throughput Doubling by Using Variable Pulse Width Modulation," Deleco Electronics, Data Book 91074, *SAE Transactions*, 946-954, 1991.

Halter, R. and F. Miesterfeld. "Survey of Encoding Techniques for Vehicle Multiplexing." Chrysler Corporation, Data Sheet 91075, SAE International Conference and Exposition, 125-136, 1991.

Heintz, F., R. Bosch, and W. Bremer. "Advanced Engineering Measurement and Information Systems of Future Vehicle Wiring Systems-Multiplex," *IMechE*. C391/035, 1989.

Louch, R. J. and D. C. Franks. "J1850 Development Tools." *Automotive Engineering* 99 (9), 17-20, 1991.

Malusardi, P. "Use of One ST9 Timer for Handling a J1850 50-kbps Implementation." Data Sheet 910710, SAE Intl. Congress and Expo., 1991.

Tanaka, M., K. Hashimoto, and Y. Himino. "High Reliability Physical Layer for In-Vehicle High-Speed Local Area Network." *SAE* 910464, Feb. 1991.

IRS

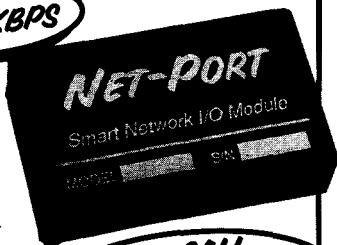
- 413 Very Useful
- 414 Moderately Useful
- 415 Not Useful

NET-PORT

TM

\$69.00

115 KBPS



Net-Port is a complete serial data acquisition and control system in a Y-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 115 kbps
- Sixteen parallel I/O lines and PC bus
- 4-channel, 8-bit ADC (Net-Port B)
- 2-channel, 12-bit ADC and 2-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

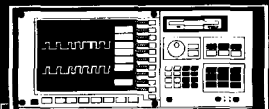
CALL
1-800-635-3355
TO ORDER

Products...

- AVT-1850-E
J1850 VPW Development System
- AVT-715
J1850 VPW & PWM Dual Interface
- AVT-921 & 922
PC Based & Hand Held Automotive Diagnostic Tools (Under Development)

Engineering Support...

- Custom Software Development
- Custom Hardware Development
- On-site & Off-site Software & Hardware Support



Advanced Vehicle Technologies multiplex bus products support the design and testing of vehicle network components.

- Automotive Multiplex Bus Engineering: J1850 VPW, PWM, & ISO-9141
- Analog & Digital Hardware Design
- Embedded Software/Firmware Development
- PC Based Software Development
- Hardware & Software Systems Engineering & Integration
- Custom Prototype Software & Hardware Development, Assembly, and Test

We can provide you with vehicle network expertise, products, and resources.

Contact us today.

AVT

Advanced Vehicle Technologies, Inc.

410-798-4038 (Voice)
410-798-4308 (Fax)

1500 Manor View Road
Davidsonville, Maryland 21035

DEPARTMENTS

52 Firmware Furnace

62 From the Bench

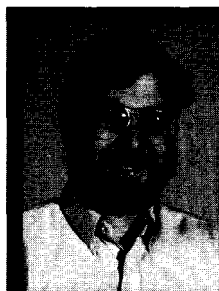
68 Silicon Update

81 ConneCTime

FIRMWARE FURNACE

Ed Nisley

80x86 Performance Probing the Cache



Building
on last
month's
founda-

tion, Ed looks at the CPU cache in more detail. After overviewing the hardware, he makes a point of pulling out of it all the timing information he can.

For reasons that made sense at the time, our house has a detached garage. My workbench, machine tools, parts bins, and raw-material stockpiles (pronounced "junk" by the untutored) line the walls around the cars. I generally find what I need without much searching, but going out to the garage on a rainy evening seems like a lot of effort.

As a result, my bottom desk drawer has a stash of tools, small parts, screws, nails, and electrical connectors. Checking that drawer is much easier than hiking to the shop, even though I may not find what I'm looking for. If I must trudge outside, rest assured I bring back a few extra widgets for the drawer.

And that sums up nearly everything you must know about memory caching. Closer is faster, but occasionally disappointing. Farther is slower, but you can almost always find what you want. When all else fails, gear up for Home Depot.

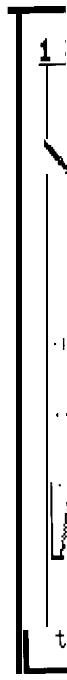
This month, we'll look at the CPU cache in more detail and pull timing information out of the hardware we built last month.

A PLACE FOR EVERYTHING

The starting point for understanding the '486 cache lies in a simple fact:

each ca
data fro
dress. V
value f
address:
compa
address:
find a
Sho
situati
contro
spondi
occurs
match
the ca
reques
the ex
ler de:
match
can ho
address

A I
leads
cache
address:
each c
are ge
ing m
to ove
WI
gram!
cluste
can s



Phot
a friar
output

each cache entry contains both data from memory and its address. When the CPU requests a value from a particular memory address, the cache controller compares that address with the addresses stored in the cache to find a matching entry, if any.

Should one entry match, a situation called a *cache hit*, the controller returns the corresponding data. A cache miss occurs when no addresses match and the CPU stalls while the cache controller fetches the requested value from DRAM or the external cache. The cache controller design prevents two or more matches, so at most one cache entry can hold data from any given memory address.

A naive approach to cache design leads to the absurd situation of each cache memory location holding four address bytes and a few control bits for each data byte. While LSI transistors are getting cheaper every year, devoting more than four-fifths of the cache to overhead makes little sense.

When you recall that most programs access memory locations in clusters rather than at random, you can see a way out. The '486 cache

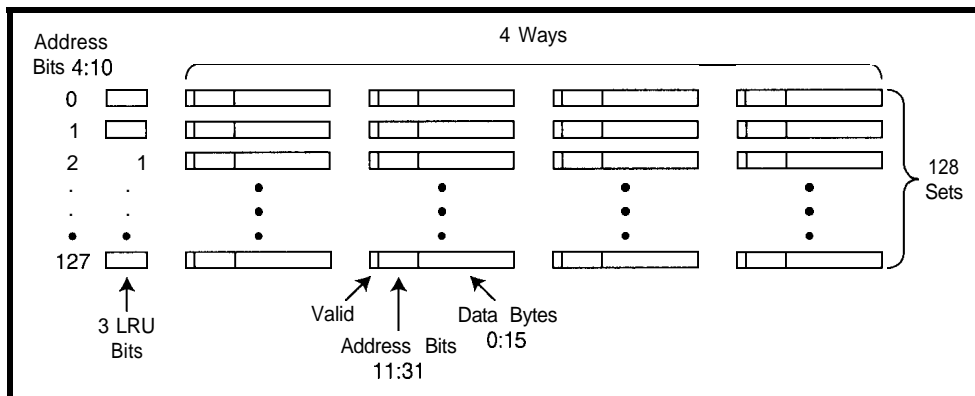


Figure 1—The Intel '486 cache holds up to 8 KB, divided into 128 sets of four 16-byte lines. The address field in each cache line holds the upper 21 bits of the data's memory address. Three Least Recently Used (LRU) bits in each set identify the oldest cache line.

takes advantage of spatial locality by storing 16 consecutive data bytes in what's called a *cache line*. A single Valid bit indicates whether the line contains usable data.

When a cache miss occurs, the controller fetches 16 bytes containing the requested byte from DRAM, selects and fills a cache line, and sets the line's Valid bit. A cache hit occurs when the CPU requests another byte from that line. If the CPU asks for more than one byte, but any part of the request lies outside the line, another cache miss occurs and the controller fetches the missing data from DRAM into another line in an adjoining set.

The '486 has what's called a *unified cache* because it holds both code and data values. Other CPUs, notably the Pentium, sport separate code and data caches. A single, unified cache obviously requires less control logic than two caches, but a unified cache can run into problems when code and data fight for the same cache lines.

EVERYTHING IN ITS PLACE(S)

Locating a byte in main memory requires nothing more than its address, a 32-bit number specifying a single byte in the '486 CPU's 4-GB address space. Finding a byte in the cache takes more effort because each cache line carries part of the memory address along with it. The cache is an associative memory: the CPU shows it an address and asks, "Do you have the data corresponding to this address?"

That naive cache design requires a fully associative memory in which the cache controller must examine the address stored in every line to find a match. Homework: design a circuit that can identify which of 512 cache lines (if any) has the matching address. Extra credit: implement your circuit with only three layers of metallization.

Rather than that maze, the '486 has a "four way, set associative" cache memory, as shown in Figure 1. Bits 4:10 of the memory address select one of the 128 sets of cache lines. If the cache has the data, it must be in one of that set's four cache lines. In a single stroke, a set-associative cache reduces the circuit complexity by about 99%!

However, a set-associative cache trades off generality for simplicity. A

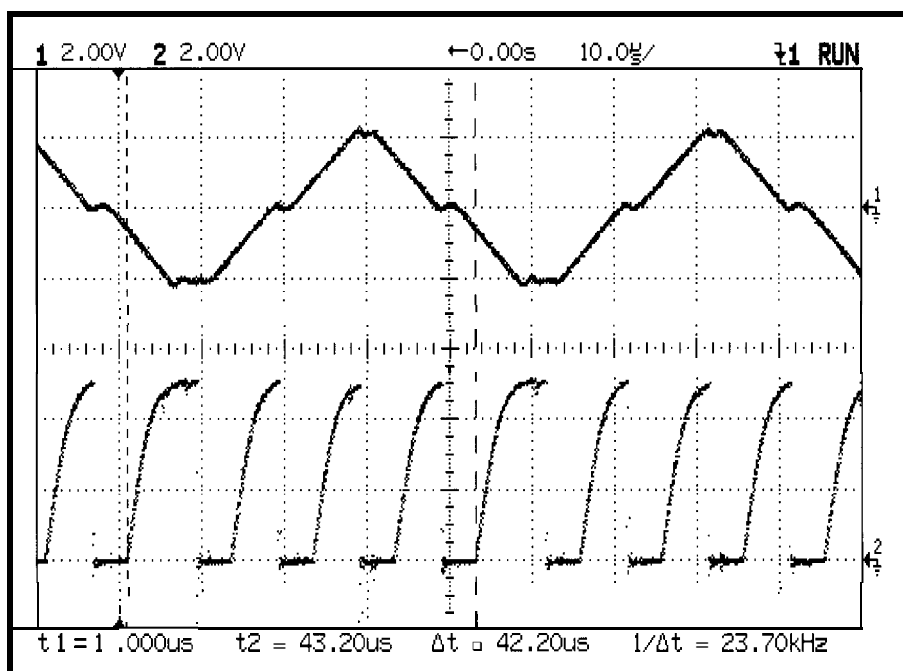


Photo 1—The DDS loop can produce a 23.7-kHz sine wave with only four steps per cycle. The waveform resembles a triangle wave rather than a collection of step changes because the LM324 op-amp buffer cannot track the DAC outputs fast enough. Each lookup table access occurs during the low sections of Trace 2.

fully-associative cache can put data from any memory address into any cache line. Each '486 cache set can hold data from just the $\frac{1}{128}$ of the system's memory selected by a particular value of bits 4:10. In an 8-MB system, for example, each cache set maps 64 KB of main memory into 64 bytes of cache. We'll see the implications of this tradeoff when we watch the cache in action.

The Address Tag field in each cache line holds the upper 21 bits of the memory address for the data. The cache selects a set of four lines based on bits 4:10, then compares all four Tag fields in that line with bits 11:3. If a line in the set has both a matching Tag field and its Valid bit set, it contains the data that the CPU requested. Bits 0:3 of the address then select the starting byte within the line and the cache returns the data.

The cache controller goes through a similar process when it writes new data into the cache based on a CPU read of a fresh area in DRAM. The address selects a set, and the controller writes the data into the first cache line with a zero Valid bit. After the first few hundred instructions, however, all the cache lines are valid. So, it then selects the oldest line in the set and stores the new data.

Each cache set has three bits that implement a "pseudo least recently used" algorithm. Although the details aren't relevant here, the controller updates the set's LRU bits each time it writes new data into a cache line in the set. As a result, the controller doesn't know which line holds the most recently read data and may sometimes replace an older, but more recently read, line instead of a newer, but subsequently unreferenced, line.

The '486 cache operates in "write-through" mode. Whenever the CPU writes data to memory, the cache controller also updates the cache line holding the address. If the memory address isn't already in the cache, the controller simply ignores the write and does not create a cache entry. The CPU encounters a cache miss when it attempts to read the new data later on.

A write-through cache can simply overwrite old cache lines because the

Listing 1--This Direct Digital Synthesis loop runs as fast as the CPU allows. The output frequency thus depends on both the phase increment and the CPU's execution speed. The GetInputs routine reads the four parallel-port DIP switches that select the phase increment.

```

MainLoop:
  MOV     DX,[ControlPort]   ; signal each output
  IN      AL,DX
  XOR     AL,FLAG_LOOP
  OUT     DX,AL

; compute and send new output value
  AND     AL,NOT FLAG_MEMORY ; signal table lookup
  OUT     DX,AL

  MOV     ESI,EBX            ; get high word of phase
  SHR     ESI,16             ; ..into SI for addressing
  AND     SI,OFFFCh          ; ..make dword offset
  MOV     EAX,[ES:SI+OFFSET SineTable] ; get table entry
  SHR     EAX,24             ; move MSB to AL
  MOV     DX,[DataPort]     ; send out the new value
  OUT     DX,AL

  MOV     DX,[ControlPort]   ; fetch control port address
  IN      AL,DX
  OR      AL,FLAG_MEMORY    ; signal table lookup
  OUT     DX,AL

; update phase angle
  CALL    NEAR GetInputs     ; fetch new phase increment

  CMP     AX,DI
  JE      NoChange
  XOR     EBX,EBX           ; new switch value, resync phase
NoChange:
  MOV     DI,AX             ; save new switches

  SHL     AX,3              ; make double dword offset
  MOV     SI,AX

  AND     EBX,[SI+OFFSET PhaseSteps+4] ; strip low bits
  ADD     EBX,[SI+OFFSET PhaseSteps]   ; increment phase
  JNC     MainLoop          ; ..if no wrap, continue

; if wrapping at 360 degrees, do some overhead
  MOV     DX,[ControlPort]   ; signal each wrap
  IN      AL,DX
  XOR     AL,FLAG_WRAP
  OUT     DX,AL

  MOV     DX,[DataPort]     ; aim at output again

  JMP     MainLoop          ; always loop when refresh is OFF

```

external memory already holds the same value. Another design, a "write-into" cache, does not update the external memory until the controller must reuse that line for new data. At that time, the controller copies the old data into memory before refilling the line.

A compromise design, a "posted-write" cache, updates the cache and delays the memory update until the external bus has a free cycle.

The references, notably the Intel hardware databooks, give complete details on how internal cache operates.

Those of you writing down-to-the-silicon embedded code should take note of the CPU's cache test facilities. If you have a desperate need for speed, the test registers can turn the cache into an 8-KB, zero-wait-state unchanging buffer that just holds vital lookup tables and code.

How does all this work in real life?

RUSHING THE CACHE

The '486 cache controller assumes, quite reasonably, that programs exhibit both temporal and spatial local-

ity. We can examine the cache in action by writing a program which carefully violates those assumptions on command. Last month, I introduced a Direct Digital Synthesis (DDS) loop that runs through a vast amount of data in a controllable manner.

Implementing a software DDS requires about half a dozen instructions. The key instructions in Listing 1 fetch a sine-wave lookup table entry, send it to the parallel port, select a phase increment based on the DIP switch inputs, and add it to the phase accumulator. The remaining instructions have nothing to do with DDS and everything to do with producing tracing outputs to show what's going on!

Each pass through the loop copies the high-order byte from a single SineTable entry to the parallel printer port. As you can see in Listing 2, SineTable contains 16,384 entries, each four bytes long. A normal DDS lookup table would have, at most, 24-bit entries. Even CD-quality audio requires only about 16 bits of sine-wave accuracy.

I put SineTable into its own 64-KB segment to force the linker to start it on a paragraph boundary. That means the first table entry starts at an address that's a multiple of 10 hex. Therefore, every four consecutive entries fit perfectly into a single cache line.

The PhaseSteps lookup table in Listing 3 converts the four input bits from the DIP switches into a full 32-bit phase increment value. My interest in cache bashing, rather than pretty sine waves, justifies the choice of the last eight PhaseSteps entries. They step through the cache in predictable ways that produce ugly results.

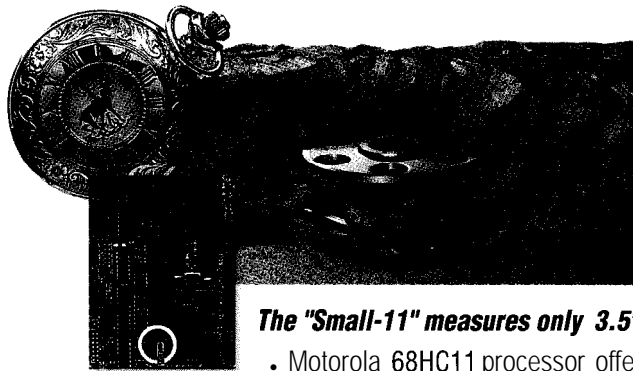
The last PhaseSteps entry, selected with DIP switches set to F = 1111, adds 40000000 to the phase accumulator and produces a 24-kHz "sine wave" with only four discrete values. Photo 1 shows both the analog output and the FLAG_Memory output bit marking the table lookup for each step. The longer pulse near the sine-wave minimum marks the overhead occurring when the phase accumulator wraps through the end of the table.

A single cache set must hold the four different cache lines containing

Listing 2—These are a few of the entries occupying 16,384 offset-binary doublewords in the 64-KB sine-wave table. Only the most-significant byte of each entry goes to the DAC through the parallel printer port. The other three bytes simply soak up space in the CPU cache.

LABEL	SineTable		Index	Radians	Sine
PUBLIC	SineTable				
DD	07FFFFFFFh	;	0	0.000000	0.000000
DD	0800C90FCh		1	0.000383	0.000383
DD	0801921FAh		2	0.000767	0.000767
DD	0DA70B257h	;	2046	0.784631	0.706564
DD	0DA799667h	;	2047	0.785015	0.706836
DD	0DA827998h	;	2048	0.785398	0.707107
DD	0DA8B5BEAh	;	2049	0.785782	0.707378
DD	0DA943D5Dh	;	2050	0.786165	0.707649
DD	0801921FAh	;	8190	3.140826	0.000767
DD	0800C90FDh	;	8191	3.141209	0.000383
DD	08000000h	;	8192	3.141593	0.000000
DD	07FF36F04h	;	8193	3.141976	-0.000383
DD	07FE6DE04h	;	8194	3.142360	-0.000767
DD	0258F4DA8h	;	10238	3.926224	-0.706564
DD	025866998h	;	10239	3.926607	-0.706836
DD	0257D8667h	;	10240	3.926991	-0.707107
DD	02574A414h	;	10241	3.927374	-0.707378
DD	0256BC2A3h	;	10242	3.927758	-0.707649
DD	07FDA4D06h	;	16381	6.282035	-0.001150
DD	07FE6DE05h	;	16382	6.282418	-0.000767
DD	07FF36F04h	;	16383	6.282802	-0.000383

When you don't have time to reinvent the wheel



The "Small-11" measures only 3.5" x 4.5"

Eliminate the time and hassle of extra design turns with the Small-11™ controller

- Motorola 68HC11 processor offers familiar programming and bullet-proof reliability
- 8K battery-backed RAM • Watchdog timer
- Up to 32K on board memory • Real-time clock
- RS422/485, and dual-RS232 ports
- Keypad and LCD interfaces • Expansion bus to I/O interface • Single 5 volt supply

Micro Control & Diagnostics™

Micro Control & Diagnostics, LLC
300 Main St., Suite 201, Lafayette, IN 47901

Phone: (800) 429-6797 or (317) 429-6777 • FAX (317) 429-6544 • Web: <http://www.mcontrol.com>

the four `SineTable` entries because

bits 20:26 of the phase increment are zero. Regardless of where the linker and loader put `SineTable` in RAM, each of the four entries that produce the sine wave have the same value in bits 4:10 and, thus, reside in the same cache set.

Figure 2 shows how the phase accumulator and phase increment produce physical memory addresses. The DDS loop uses the high-order 16 bits of the phase accumulator to read the 64-KB `SineTable` lookup table, thus mapping bits 20:26 of the 32-bit phase increment into bits 4:10 of the entry address. Remember that the cache acts only on the physical addresses presented to memory, not on the intermediate calculations, so we must keep track of where the bits wind up.

As long as the DDS program code uses different cache sets, every `SineTable` access is a cache hit. I examined the linker map and found that the main loop occupies sets 4 through B (hex, of course). Although the DOS loader adds a constant offset to those addresses that may move them to different sets, all of the DDS code and data resides in the cache. In this situation, the CPU runs as though it had zero-wait-state memory!

Photo 2 shows a magnified view of the `FLAG_MEMORY` pulse bracketing each `SineTable` reference. Over this stretch of code, the CPU runs 11 instructions in 4.1 μ s for a rate of 2.7 MIPS.

Keep that number in mind when you read phenomenal MIPS scores elsewhere. Just reading a table and sending the output to a DAC can be surprisingly CISCy.

So much for a perfectly cacheable program. It's time to get nasty...

THRASHING THE CACHE

Setting the DIP switches to 9 = 1001 selects a DDS phase incre-

Listing 3—The DIP switches select one of these phase increments for each pass through the DDS loop. The first eight entries generate various (very) low frequencies, while the remainder exercise the CPU's internal cache control logic.

LABEL	PhaseSteps	DWORD	
DD	00000001h,NOT	00000001h	; 0 0000 lowest possible freq
DD	00000010h,NOT	0000000Fh	; 1 0001
DD	00000100h,NOT	000000FFh	; 2 0010
DD	00001000h,NOT	00000FFFh	; 3 0011
DD	00010000h,NOT	0000FFFFh	; 4 0100
DD	00100000h,NOT	000FFFFh	; 5 0101
DD	01000000h,NOT	00FFFFFFh	; 6 0110
DD	10000000h,NOT	0FFFFFFFh	; 7 0111
DD	00010000h,NOT	0000FFFFh	; 8 1000 4 outputs per value
DD	00100000h,NOT	000FFFFh	; 9 1001 step through sets
DD	08000000h,NOT	07FFFFFFh	; A 1010 force index = set 0
DD	08010000h,NOT	0000FFFFh	; B 1011 4 outputs, set 0
DD	08100000h,NOT	000FFFFh	; C 1100 step through sets
DD	08110000h,NOT	000FFFFh	; D 1101 4 outputs, many sets
DD	10000000h,NOT	0FFFFFFFh	; E 1110 16 samples/cycle
DD	40000000h,NOT	3FFFFFFFh	; F 1111 4 samples/cycle

ment of 00100000 that produces a 4096-step sine wave. Each addition ticks bit 4 of the `SineTable` entry address, causing each loop iteration to pull data into a different cache set. Multiply 4096 by 16 bytes per line and you discover each table lookup causes a cache miss.

Photo 3 shows the result: each memory access requires about 520 ns more than the perfectly cached data in Photo 2. Most instruction fetches hit the cache, but a few misses cause the

scattering of traces on the rising edge of the pulse.

Last month, you saw that the '486 CPU fetches data from DRAM (or the external L2 cache) in bursts of four 32-bit accesses that fill an internal cache line. The BIOS settings for my system dictate a 5-4-4-4 burst read pattern.

Dividing the measured 520 ns by those 17 cycles gives 30.6 ns or very nearly 33 MHz. Looked at another way, dividing 520 ns by 25 ns gives about 21 cycles.

There are two possible conclusions. Either this 80-MHz '486DX2 system really runs with a 33-MHz external clock or there are four cycles not counted in the BIOS settings. I suspect the latter, for reasons that I'll mention shortly. Hint: the system board includes a pair of VLB slots.

Anyhow, the interesting fact remains that a single, isolated, uncached memory access costs half a microsecond on a system that could otherwise execute two or more instructions. The effect of several misses can add up dramatically in tight loops.

As a simple example, consider how array storage affects a program's execu-

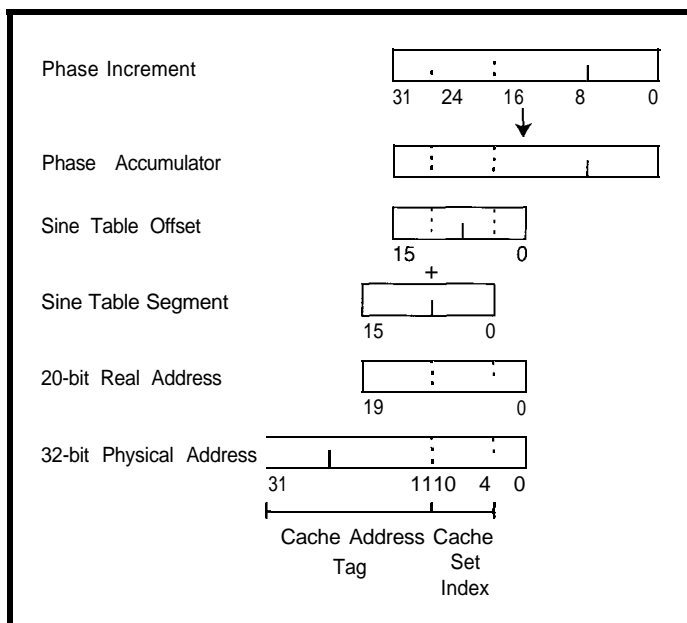


Figure 2—The 32-bit physical address of each memory reference determines where the data winds up in the cache. The DDS loop selects a lookup table entry using the high-order 16 bits of the 32-bit phase accumulator. Thus, bits 20:26 of the phase increment affect bits 4:10 of the physical address and select the cache set index.

tion. C specifies that arrays must be stored in row-major order, putting successive entries in each row in adjacent memory locations. If you scan along the rows, many accesses hit the cache. Quiz: explain why.

Should you scan along the columns, however, the cache controller busies itself fetching additional information in each row. If the array is bigger than the cache—a typical situation in real applications—those cache lines are discarded long before they're referenced again. Nearly every array access then causes a cache miss.

FORTTRAN, at least the classic pre-F90 versions, stores arrays in column-major order. Simply translating an optimized FORTRAN subroutine to a C function practically guarantees lower performance because the data moves underneath the algorithm. Essay question: how would you fix this!

The answer to last month's quiz about what happens with small phase increments should be obvious now. As long as the phase increment doesn't reach bit 4 of the physical address, every table reference after the first cache miss is a cache hit. The ratio of hits to misses can be extremely high and give the illusion of a nearly zero-wait-state memory.

Setting the DIP switches to 8 = 1000, for example, produces an increment of 00010000 and a sine wave with 64K steps. Four successive steps use the same table entry, so 15 out of every 16 table accesses are cache hits.

I don't have room to show the scope shot, but you can imagine what it looks like based on Photos 2 and 3. Even on a digital scope, a heavy trace marks repeated cache hits at 4.1 μ s and a lighter trace shows the occasional misses at 4.62 μ s.

Other effects come into play when you examine the detailed bus timings. The cache controller fetches the 16 bytes of data for each line from main memory out of order, returning the requested bytes to the CPU first and filling the rest of the cache line later. As a result, the instruction decoder can start up while the cache line is filling. The multiple times on the rising edge of Photo 3 show how this works in practice.

wanna be famous?

Are you or your company using PC/104 technology in an interesting or unusual way? Tell us about it.

PC/104 DESIGN CONTEST

You are invited to submit *unique* PC/104 projects or applications to our design contest. Be sure to include functional block diagrams with descriptions of the hardware, software, and peripherals used. Contest entries will be judged for technical merit, applicability, and originality. The judges: Circuit Cellar INK's Steve Ciarcia, Ampro's Rick Lehrbaum, and Embedded PC's Managing Editor Janice Marinelli. We'll highlight winning applications in Embedded PC, plus designers will be invited to submit a design write-up for a future PC/104 Quarter. And there's more! Winners will receive the following PC/104 design tools:

- 1st prize **Ampro CoreModule™/486 Development Kit**
- 2nd prize **Ampro CoreModule/386 Development Kit**
- 3rd prize **Ampro CoreModule/PC Development Kit**

All entries must be received no later than August 15, 1996. Winners will be announced at September's Embedded Systems Conference and the winning project descriptions will appear in December's issue of Circuit Cellar INK.

Contact us today for your entry form and then mail your contest entry to:

Janice Marinelli, PC/104 Quarter Contest
Circuit Cellar INK 4 Park Street Vernon, CT 06066
Tel: (860) 875-2199 Fax: (860) 872-2204

www: <http://www.circellar.com/> E-mail: PC104.contest@circellar.com



Ampro's CoreModule™ development kits for PC/104 and license software come with a complete hardware utility software and hardware resources.



Sponsored by Ampro Computers, Inc., the originator of PC/104, and Circuit Cellar INK, home of Embedded PC

First in performance First to market



**The Small-XA™
Controller
leaves 8051
boards in the
dust, without
time-consuming
source-code
rebuilt**

- 16-bit 80C51XA operates up to 30 MHz, yet remains 8051 compatible
- Up to 20-bit program address space
- Pipelined architecture



- 64K EEPROM
- 64K battery-backed RAM
- Expansion bus to I/O interface
- Real-time clock and watchdog timer
- RS422/485, and RS232 serial ports
- Available I/O expansion modules, development software, and other accessories
- Custom-engineered solutions

**Micro & Control
Diagnostics™**

Micro Control & Diagnostics, LLC
300 Main St., Suite 201, Lafayette, IN 47901

Phone: (800) 429-6797 or (317) 429-6777 • FAX (317) 429-6544 • Web: <http://www.mcontrol.com>

Listing 4—The DDS code depends on precise, repeatable timings. This startup code disables all interrupts and shuts off DRAM refreshing to eliminate those glitches. As a result, you must reboot your PC after running the DDS loop!

```

CODESEG
Start:
STARTUPCODE ; set up DS, SS:SP

; figure out port addressing
MOV     AX,0040h           ; fetch LPT 1 port address
MOV     ES,AX
MOV     DX,[ES:0008h]
MOV     [DataPort],DX
INC     DX
MOV     [StatusPort],DX
IN      AL,DX
MOV     DI,AX             ; save for change detection
INC     DX
MOV     [ControlPort],DX

; wait for diskette to stop spinning
FloppyWait:
TEST    [WORD PTR ES:0003Fh],003h
JNZ     FloppyWait

; shut off RAM refresh (yikes!)
CLI     ; prevent timing distractions
MOV     AL,TMR1_MODE      ; set Timer 1 mode
OUT     I8254A_CTL,AL
MOV     AL,1              ; set 0001 timeout
OUT     I8254A+1,AL

; set up data pointers
MOV     AX,SEG SineTable
MOV     ES,AX
ASSUME  ES:SEG SineTable

MOV     ECX,[PhaseSteps] ; set up phase increment
XOR     EBX,EBX           ; start at zero phase

```

I looked at those traces under higher magnification and found the various pulse widths differ by multiples of 25 ns. The earliest one begins at -250 ns and the latest one at +125 ns (refer to the heavy trace). Recall an 80-MHz '486DX2 runs with a 40-MHz external clock, giving a 25-ns basic time.

Clock-doubling CPUs depend on a very stable external clock because their internal phase-locked loops simply cannot cope with very much jitter. Thus, if a CPU has a few 25-ns cycles, all of them must be precisely 25 ns long. That immediately rules out a 33-MHz CPU clock.

However, recall those VESA Local Bus slots. The VLB spec dictates a 33-MHz maximum clock, not the 40-MHz one supplied to the CPU.

The cache controller fills cache lines with bursts of four accesses. Stretching each access by a single 25-ns cycle produces a 20-MHz basic rate, well within the specs. As a result, the burst memory pattern runs at 6-5-5-5 cycles, matching the 21 cycles we see in Photo 3.

I can't find anything in the system documentation or BIOS settings that mention this effect. The VLB jumpers select zero-wait states, too. Systems are now so complex that you simply cannot tell what's going on under the hood without actually measuring it.

A truly careful study would include probing the system board and measuring the buses directly, but without schematics and chip documentation, I'll pass. An hour spent watching a scope while trying various phase increments will reward you handsomely.

I heartily recommend keeping the CPU documentation handy to explain things like the interaction between the instruction prefetch queue and the cache, byte fetch order during burst reads, and so forth.

You will find a few surprises lurking in your system. If not, you're not looking hard enough!

STABILIZED TIMES

The scope shots in this column should recall Holmes' observation about "the curious incident of the dog in the night-time." Nothing disturbs the pristine clarity of the traces: no

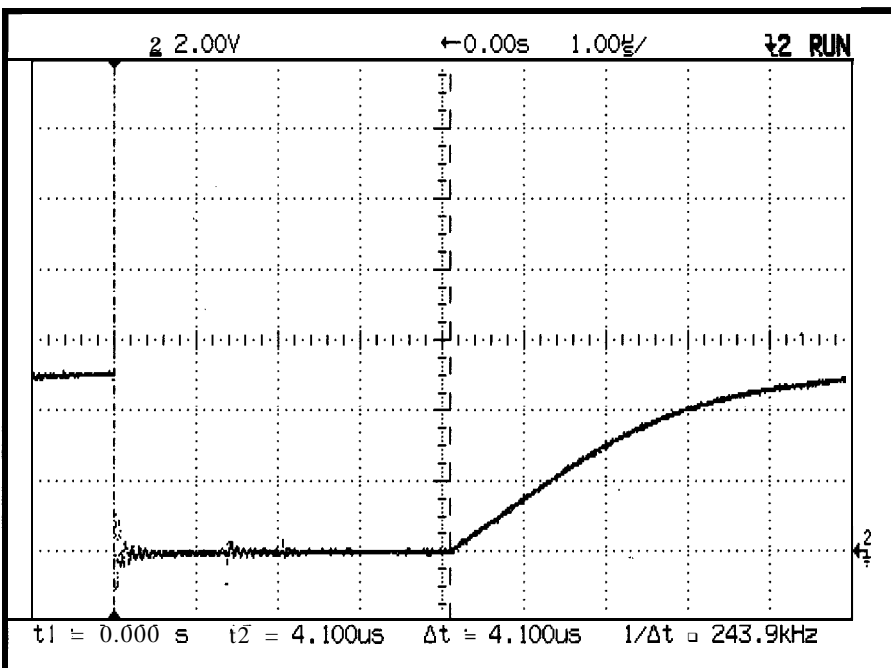


Photo 2—The low part of this trace brackets the DDS sine-wave table lookup operation. The '486 CPU executes 1 instructions while FL A G_ M E M O R Y is low for an average rate of 2.7 MIPs. The system acts as though it had zero-wait-state memory because every instruction and data fetch comes from the L1 cache.

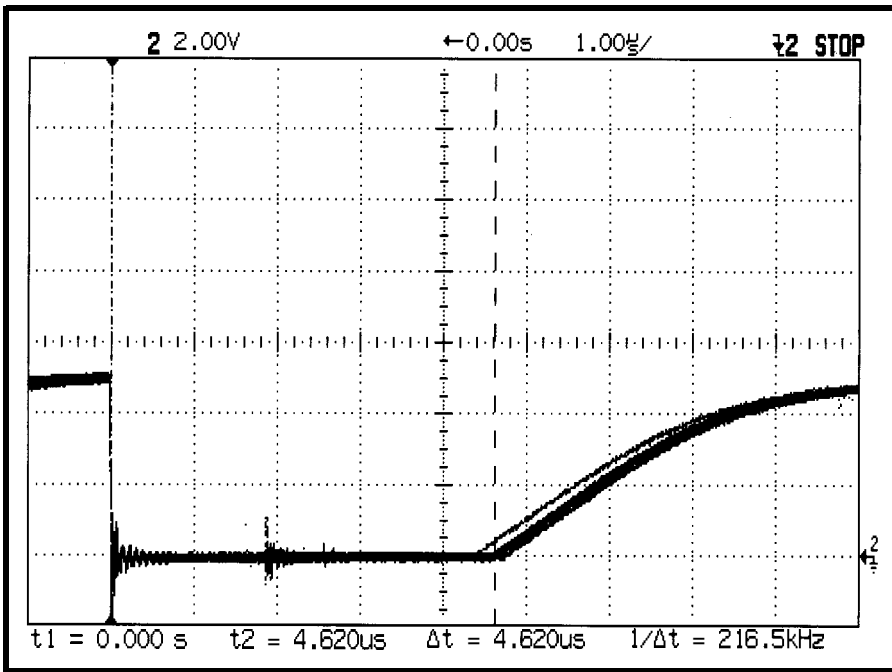


Photo 3—Placing each sine-wave step in a different cache set ensures that nearly every data access is a cache miss. Most instruction fetches hit the cache, although instruction prefetching produces the scatter at the rising edge. Compare the pulse width with that found in Photo 2 to see the effect of missing a single memory access!

interrupts, no DRAM refresh, no DMA, nothing. How can this be?

A quick look at Listing 4 shows why the DDS program may be the most hostile nonviral program you'll ever encounter. I deliberately shut off

all interrupts and disable the DRAM refresh timer before entering the DDS loop. The system's Reset button provides the only exit from the loop.

If you select very low output frequencies, the output waveform may

deteriorate after a few minutes. At those rates, the DDS loop doesn't read each DRAM location often enough, and the data stored there quietly vanishes. Fortunately, the precise cache timings don't depend on the data going to the DAC.

Other systems use different DRAM refresh hardware, some of which may not depend on Timer 1. Should your scope reveal small timing glitches, it may be that your system's refresh remains active. I'll leave killing your system to your own ingenuity.

The code detects when the BIOS turns off the floppy drive motor by polling a byte in the BIOS data area. With interrupts permanently disabled, the BIOS cannot get control to shut

down
second
it ma!
It s
run tl
multi
decen
vital :
vents
the p:
run, y
it anc
I a
syste
could
separ
Exert
witho
poses
of th:
or tip
REL
PI
simp
suffe
The
about

**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
(860) 871-6170 • Fax (860) 872-2204

in Europe: (44) 1285-658122 • in Canada: (514) 336-9426 • Distributor Inquiries Welcome

down the drive after the usual few seconds. Although floppies are durable,

vents the I/O operations or terminates the program. While the DDS loop may it and it'll probably crash the system.


separate the effects of the two caches. Exercising a 256-KB external cache without knowing its design details poses a problem well beyond the scope of this column. If you have any hints or tips, drop me a note on the BBS.

RELEASE NOTES

Photo 1 clearly shows that the simple LM324 op-amp buffer I used suffers from severe slew-rate limiting. The ramps connecting the steps run at about 0.25 V/ μ s, nowhere near the rate

needed to reproduce the DAC's abrupt step outputs. If you want a sine-wave output, however, filtering a triangular waveform poses fewer problems than flattening the steps.

Remember to reboot your system after running this month's code. Don't fire it up on your business computer!

Next month, a look at several different x86 CPUs on some real embedded-PC hardware. 

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.

REFERENCES

Robert Hummel's *The Processor and Coprocessor* (Ziff-Davis Press, ISBN 1-56706-016-5) has a brief but lucid description of the '486 caching machinery. This book may be out of print by now,

but you should snap it up if you find a copy.

Intel's *Microprocessors: Volume 2 Handbook* (ISBN 1-55512-197-7, Intel Order Number 24173 1-001) has all the '486 hardware details and a variety of app. notes on memory and L2 cache design. It's not for the fainthearted.

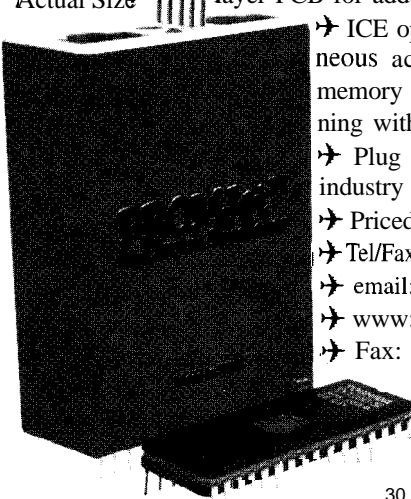
On the software side, their *Intel 486 Microprocessor Family Programmer's Reference Manual* (ISBN 1-55512-159-4, Intel Order Number 240486-002) gives a somewhat less detailed view of the L1 cache, but covers the essentials needed to initialize and test it.

You may download AMD CPU data sheets from their Web site: <http://www.amd.com/>.

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

PROMJet[®]-ICE

▶ Ultra compact EPROM and FLASH emulator with highest download speed (1-4 Mb/S), largest memory capacity (1~32Mb) and fastest access time (85~25ns) in the industry.
 ▶ Other features include 3V target support, jumperless configuration, battery backup, 128 bit bus support and external power supply. ▶ Fits directly into memory socket or uses extension cable for flexibility. ▶ Compact design based on high density FPGAs and double-sided surface-mounted 10 layer PCB for added reliable operation.
 ▶ Actual Size  ICE option allows simultaneous access to PROMJet's memory while target is running without waitstate signal.
 ▶ Plug & Play drivers for industry standard debuggers.
 ▶ Priced from U\$295 / MBit.
 ▶ Tel/FaxBack: 206.337.0857
 ▶ email: emutec@emutec.com
 ▶ www: www.emutec.com
 ▶ Fax: 206.337.3283

EmuTec Inc
 Everett Mutual Tower
 2707 Colby Av, Suite 90
 Everett, WA 98201, USA
 30 day money-back policy
 Visa & Mastercard accepted

2.2x1.7x0.7"

#127

TIRED OF WAITING FOR THE PROMPT ?

Speed up your programs. DOS and programs instantly. Also diskless workstations. Perfect protection from viruses. Easy to install half-size card.

VVDISK1 128k \$75
 VVDISK2 1.44m 5150
 VVDISK3 576m 5195

\$75

Quantity discounts!

DOS IN ROM!



\$95 EPROM PROGRAMMER

- Super Fast Programming
- Easier to use than others
- Does 2764/27080(8 Meg)



WORLD'S SMALLEST PC !!!

ROBOTS ALARMS RECORDERS DOS

THREE EASY STEPS: \$27 1K QTY
 1. Develop on PC
 2. Download to SBC \$95 SGL QTY
 3. Burn into EPROM

- 2 PARALLEL -LCD INTERFACE
- 3 SERIAL -KEYBOARD INPUT
- PC TYPE BUS -REAL TIME CLK
- BIOS OPTION -BATTERY OR 5V

FREE SHIPPING IN U.S.

5 YEAR LIMITED WARRANTY

MVS Box 850
 Merrimack, NH
 (508) 792 9507

8088 SINGLE BOARD COMPUTER



#128

Handcrafting Design Ideas

FROM THE BENCH

Jeff Bachiochi



Whether you've done your initial design with a CAD package, worked it out on the chalkboard of your mind, or documented it on the proverbial dinner napkin with a client, you're gonna want something a bit more tangible.

You may need thousands or only one. Do you go straight to a PCB? For many of us, this method is not a cost-effective way of proving a design. And, it does nothing for those of us who have the need to get down and dirty with the individual parts.

It's not easy to categorize this kind of hands-on engineer. After all, we come from different backgrounds with different norms. For some, it's the sweet smell of solder. For others, it's having complete control over each component. We call out the right formation as each situation develops with the ultimate goal of a winning circuit.

And then, there are those who just like to tinker with the beast, tweaking and tuning until it runs as smoothly as possible.

Lest we forget the SMT surgeons—those with the skill and dexterity necessary to attach or, harder still, replace a vital organ. Microscopic eyesight, a steady hand, and plenty of patience are necessary for this kind of operation.

ALL ROADS LEAD TOWARD ROME

This month, I'll touch on a number of ways you might put your design to the test. These methods include solderless breadboarding, wire wrapping, and point-to-point wiring (which includes channel, weave, 3D, and SMT). Some approaches require special tools and patience beyond belief.

Every design requires thought prior to picking up any pieces. Parts placement can be critical in circuits where high gains and high impedances increase the circuit's sensitivity to noise. The general flow of the circuit (from inputs to outputs or larger parts to smaller) helps indicate the proximity between associated parts.

Although at this point, you may not be considering a manufactured PCB, PCB layout tools can be extremely helpful in determining the best arrangement of parts. Most PCB packages have a rat's-nesting mode which displays the signal connections between part outlines.

As you move the parts [outlines] around onscreen, associated connections stretch and shrink. This feature visually indicates where connections have to be made so you can easily determine optimum placement.

You can stop the PCB design process in the layout stage if you wish. Although not a necessity, the PCB package helps you find good placement without physically moving the parts around. Plus, you can print out the screen placement to help in the prototyping stage.

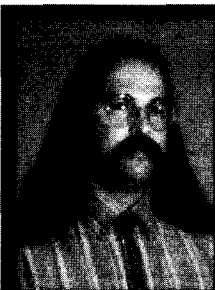
SOLDERLESS BREADBOARDING

For those who wish immediate gratification, solderless breadboarding is the shortest route to applying power. In minutes, you can wire a simple circuit and scope the results.

Solderless breadboards are made of plastic and are covered with a grid of holes on 0.1" centers. Beneath the holes are tiny metal contacts which are usually connected in strips of five.

Although the strips can be laid out in any pattern, the most common is in two parallel rows spaced 0.3" apart. This spacing lets an IC fit between the rows. Photo 1 offers some of the solderless breadboard possibilities.

The legs of an IC or component extend through the holes, making contact with the connectors below. Each lead has four additional contacts connected to it. You can make connections easily between IC pins or components by poking a jumper wire or lead into holes on the contact strip of the original component.



It's back to basics. Jeff walks through

the fundamentals of solderless breadboards, wire wrapping, and point-to-point wiring. There are tips and techniques to every trade, even at the level of fundamentals.

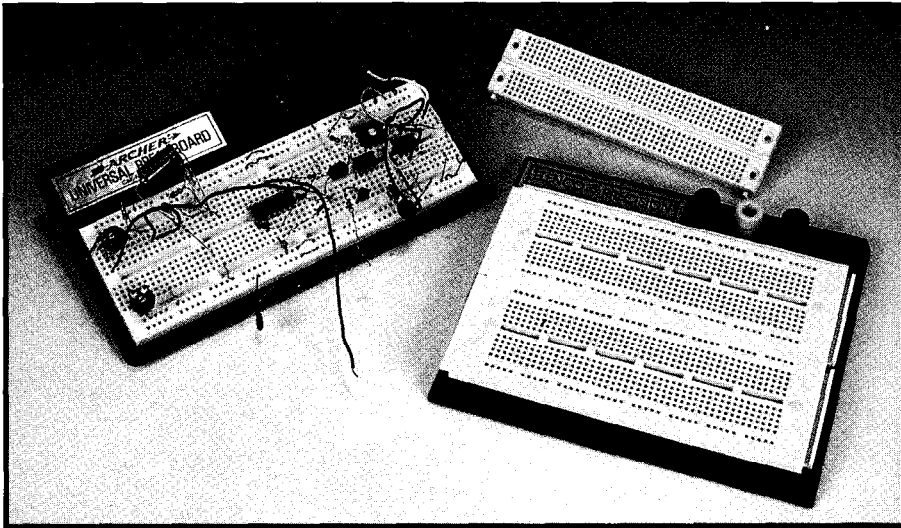


Photo 1—Solderless breadboards come in all sizes. You can piece together your own configuration or purchase a standard product.

The advantage of this system is that no soldering is involved. The disadvantage is the components are not fastened down, so the circuit falls apart if jostled too much.

I frequently use these to test a new circuit. I can quickly exchange parts while tweaking for maximum performance. I can strip off the cover from solid four-conductor phone wire and use the individual pieces for jumping between connections. It's very easy.

Be aware, however, not all solderless breadboards are made alike. Pay close attention to how contacts are formed. For best results, ensure strips are formed using individual contacts instead of a single strip contact. Single strip contacts don't adjust well to leads of different diameters and may make intermittent connection.

Many distributors offer solderless breadboards with a host of other goodies attached. Power supplies are by far the most useful. Additional gadgets include displays, switches, function generators... you name it.

Not everyone needs all this, but it's a great place to start. And if all this isn't enough, you can even get a PC/AT plug-in card covered in solderless breadboarding. Now there's a scary thought!

What's the cost? A solderless breadboard runs \$7.

WIRE-WRAPPING

Although mechanically and electrically equivalent to soldered proto-

boards, this method of interconnecting circuit nodes is also solderless. Every component is either plugged into special wire-wrap IC sockets or is attached to individual wire-wrap pins inserted into a prototyping board.

The board's matrix of holes on 0.1" centers may or may not have solder pads. Individual wire-wrap pins and IC sockets are held in with clips, glued, or soldered. All interconnections use wire-wrapping techniques.

Each wire-wrap pin is square, making the corners very sharp. The soft copper wire-wrap wire winds around the pin so tightly it welds itself to the pin's corners. A specially designed tool ensures each wrap is uniform.

Wire-wrap IC sockets are available in single-, double-, and triple-wrap height. Double height is most common since a connection generally contains two wraps—one for a daisy chain to the pin and a second for the next connection. Photo 2 shows typical wire-wrap circuitry up close.

Each connection usually starts with a piece of wire-wrap wire cut about 4" longer than the path of the connection from one pin to the next. The tough Teflon insulation is removed from each end, with the length of the strip controlling the number of wraps around a pin. Prestripped wires are available in various lengths.

Stripping the insulation should be done only with a device made for wire-wrap wire. Why? Because the wire is extremely thin. Any nick results in a weakness in the conductor. Although it might look fine, it surely will break at the worst possible moment.

A method for wrapping with a continuous unstripped wire is called slit-'n'-wrap. Here, a special tool slits the insulation. The tool places a wrap on the pin, enabling the inner wire to contact the square wire-wrap pins through the slit in the insulation.

In my humble opinion, it's difficult enough to make and inspect a good wrap when you can clearly see it. The slit-'n'-wrap method tends to hide the actual contact area, making it much more difficult to verify a good wrap.

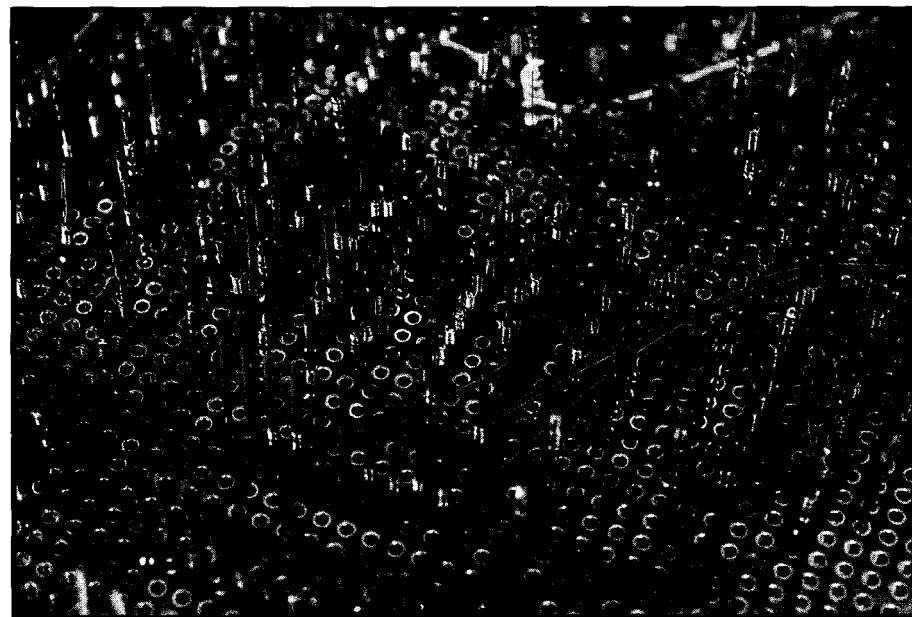
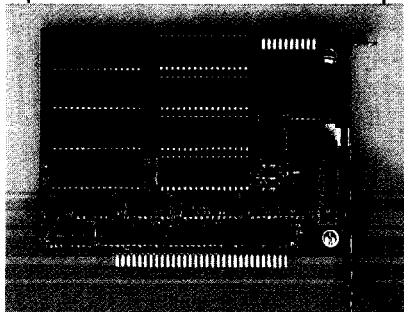


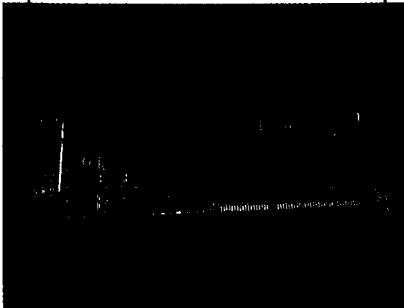
Photo 2—Wire wrapping enables circuit changes without the need for a soldering iron.

VMAX
 QUALITY PRODUCTS
 RESPONSIBLE SERVICE
 RELIABLE DELIVERY



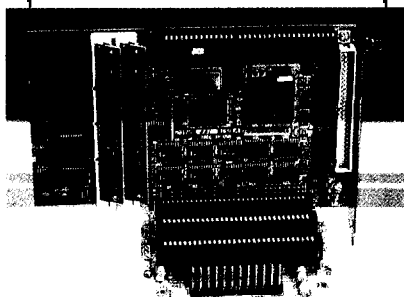
ENHANCED SOLID STATE DRIVE — \$144*

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



486 SLAVE PC — CALL

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/1Par, 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

Fast response! 295 Airport Road
 Naples, FL 33942

#129

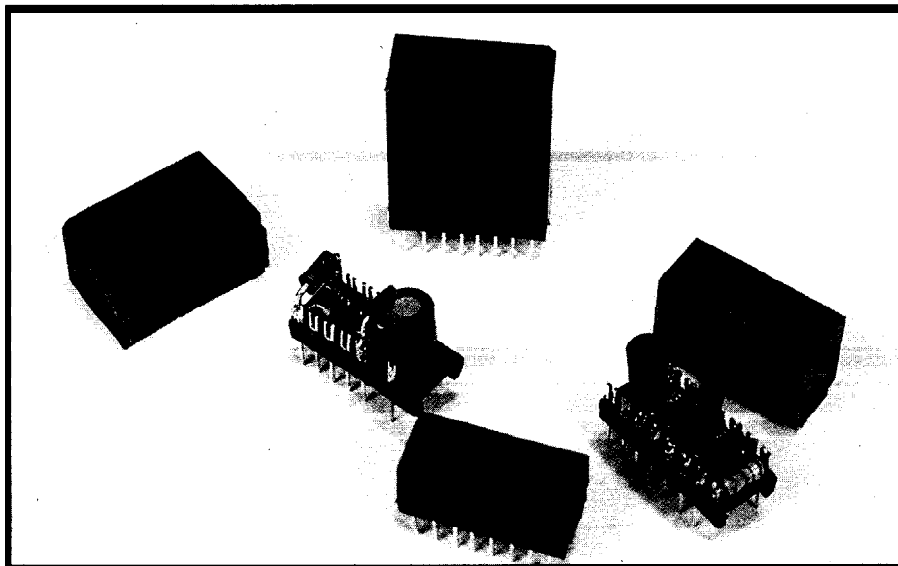


Photo 3—Mounting discrete parts or DIP headers allow them to easily plug into DIP sockets. They can even be covered for a "clean" look.

Wire wrapping works well with digital and analog circuits. Since analog circuits usually require many discrete components, connecting them to the pins is a pain. Removing or replacing components once it's wrapped can cause even more problems.

As the component's connection to the pin is (de)soldered, heat travels down the wire-wrap pin and can potentially melt any insulation lying next to the pin. This problem can be avoided by mounting the discrete components on DIP headers and plugging them into wire-wrap IC sockets. Photo 3 demonstrates how discrete parts may be mounted to simplify construction.

A drilled-only 6.5" x 4.5" PCB costs \$4, three 50' spools of 30-AWG wire are \$18, a manual wrap tool is \$19, and a power wrap tool is \$50300.

CHANNEL AND WEAVE POINT TO POINT

All prototyping is essentially point-to-point wiring, but I treat channel and weave separately because of the addition of the all-important solder joint.

Although this method of prototyping can use almost any wire, I find wire-wrap wire best because it's very thin, tough skinned, and comes in a rainbow of colors. Colored wires differentiate between circuit signals.

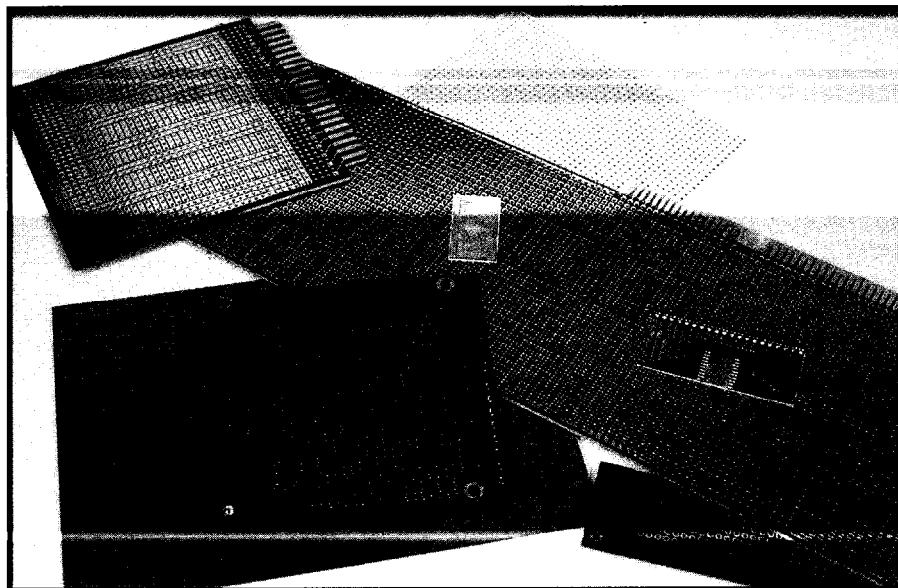


Photo 4—A variety of prototyping boards are available. Substates from phenolic to fiberglass and copper or clad to pad. Double sided and plated holes are optional.



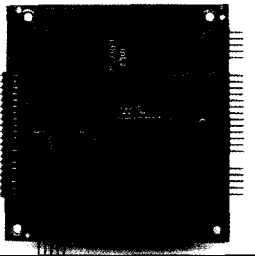
Sets the Pace

in Low Power,
High Performance
PC/104 Technologies



66 MHz
486SXLC66
\$582
100 pcs

- Features:**
486SXLC2
66 MHz
387SX opt.
8K cache
2MB DRAM
3W typical
SSD
EEPROM
WDT
IDE & FDC
2 Serial
1 Parallel
PS/2 mouse
Keyboard
Quick Boot
Virtual devices

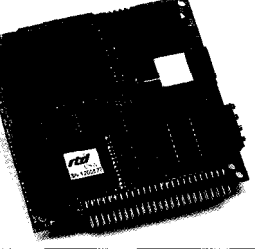


Mi486SXLC66-1 Fully Integrated PC-AT with Virtual Device Support



200 kHz
12-bit DAS
\$498
100 pcs

- Features:**
200 kHz
12-bit A/D
16SE/8DI
Scan
Burst
Multiburst
1K CGT
Triggers
FIFO buffer
Data marker
12-bit D/A
16-bit DIO
+5V operation



DM5408-2 200 kHz Analog I/O Module with Channel-Gain Table

Make your selection from:
9 cpuModules™

SuperXT™, 386SX, 486SXLC66, 486DX2, and 486DX4 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) 2351260 • BBS: 1 (814) 234-9427

RTD Europa RTD Scandinavia

Budapest, Hungary Helsinki, Finland
Tel: (36) 1 325-1130 Tel: (358) 0 346-4538

RTD is a founder of the PC/104 Consortium and the world's leading supplier of PC/104 CPU and DAS modules

While most prototyping boards have a matrix of copper pads with through holes on 0.1" centers, the cheapest proto boards are made from phenolic and simply coat the copper pads to prevent oxidation. More expensive boards are fiberglass and have solder-plated pads and through holes. Photo 4 shows a myriad of prototyping PCBs.

Channel wiring begins with a parts layout similar to wire-wrapping. However, the parts are tack soldered to prevent them from falling out. All discrete part leads are clipped to about 0.1" after tack soldering. Before clipping any discrete leads, bend them toward the part it's going to connect and solder to. This technique reduces the number of connections, but don't create a short by crossing bare wires.

Next, use wire-wrap wire to make the power and ground connections. Route the wiring into channels parallel with the ICs. You can test power and ground at this point without inserting any ICs. The minimum wiring helps to easily locate and fix any problems.

Continue with the rest of the circuit nodes until all connections are made. Once wiring is complete, strategically place wraps of wire-wrap wire around the channel bundles to keep the wire tidy and secure. The board on the left in Photo 5 shows my favorite prototyping method—point-to-point channel wiring.

Weave wiring uses the same supplies as channel wiring. If you have access to a PCB layout package and have used it to provide a parts-placement layout, you might wish to take that a step further and fully route the PCB, even though you may not wish to fabricate a real PCB at this time.

Why? Well, the route patterns produce a true representation of the board using the weave method. Weave wiring closely simulates the final PCB by following the vertical and horizontal patterns of a fully routed PCB.

You have to share feed-throughs since the hole pattern is 0.1", but the look is there. Use open-frame IC sockets and keep discrete components off the proto board's surface, so there's easy access beneath all components.

A single-sided pattern PCB (6.5" x 4.5") costs \$15 while a plated-through-hole board of the same size goes for \$18. Nonadjustable 30-W soldering iron costs \$4 while the adjustable 60-W soldering station sells for \$120. Solder (0.031") is \$10 for a 1-lb. roll.

3D CIRCUITS

This method of prototyping has roots dating back prior to solid state. Those of you who built a Heathkit know what I'm talking about. It was common practice to tie components together by connecting them to one another while suspended in the air.

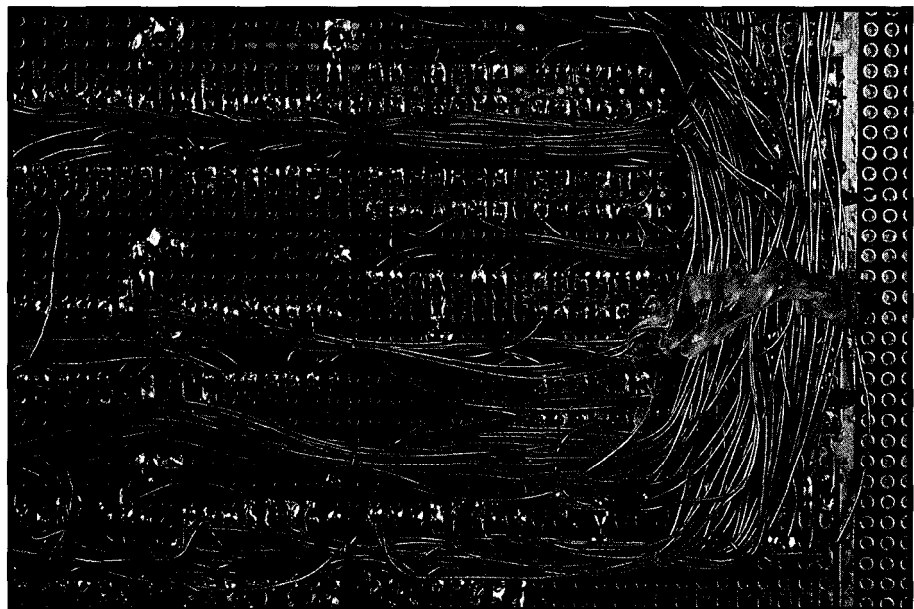


Photo 5—Point-to-point wiring uses wire-wrap wire for its size and semitough skin. It's important to leave the soldering connections uncovered.

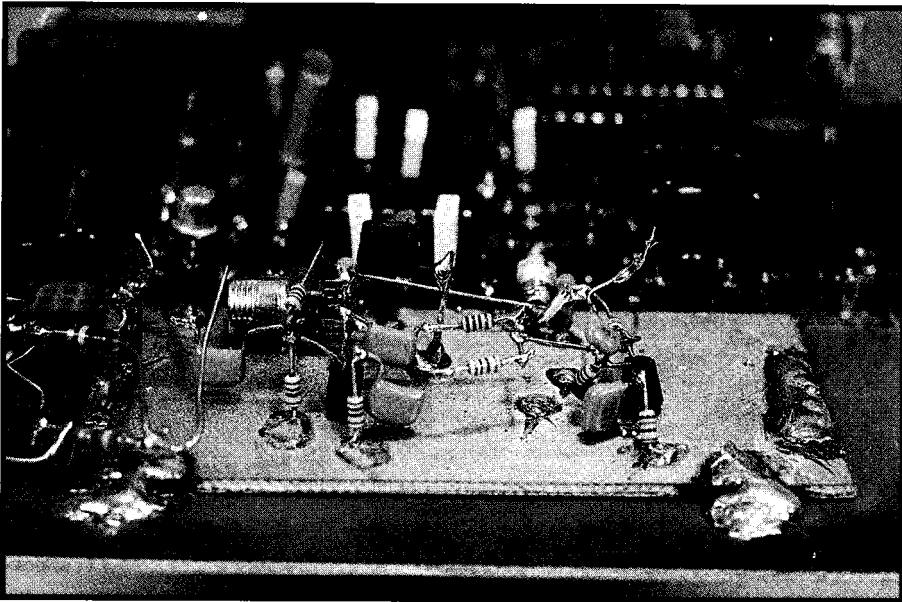


Photo 6-A ground plane forms the base for this analog sculpture

Most circuits contain components which mount on some kind of chassis. Potentiometers, I/O jacks, switches, and such become mechanically stable tie points for other components.

And, should one end of a component not have a mechanically sound connection, terminal strips created an anchor.

With the introduction of integrated circuits, the PCB became a necessity and has all but eliminated this form of prototyping. However, now and again, you might find a use for 3D circuitry.

Photo 6 illustrates how 3D fabricating serves up some analog circuitry.

The costs of 3D circuits are the same as those for channel and weave.

SMT CIRCUITS

Ever wanted to try your [steady] hand and (eagle) eye at surface mount?

Most electronic supply houses carry a large selection of SMT parts. Before you buy, though, you'd better outfit yourself with some special tools.

Start off with a soldering iron with tips down to 0.005". You'll get by with a tip of 0.05" for the discrete parts. However, for fine-pitch ICs, you won't stand a chance without smaller tips.

Although some solder manufactures offer solder down to 34 gauge, you'll probably need a cream solder or flux combination for the fine-pitch flat-pack ICs. A good set of tweezers aids in placing the tiny parts.

Don't forget a magnifying glass. Fluorescent light magnifiers improve the view, but a jeweler's loupe looks after in-your-face inspections.

Surface mount components are available through almost every distributor. And, prototyping with them is easy thanks to a line of specially designed prototyping boards. Conventional prototyping boards are replaced by single-sided patterns which easily accommodate SMT parts.

Two- and three-terminal devices are connected between isolated copper lands. SIP pins, available on many boards, allow the finished circuits to be inserted into solderless breadboards for quick testing. These pins are great for forming your own resistor, capacitor, diode, or transistor arrays.

Other surface models include SOIC layouts, so you can prototype a complete circuit including an IC and discretes. To prevent using too many wire jumpers, the surfboard land patterns require some circuit-layout thought. A few minutes planning the parts placement helps make the finished prototype a whole lot neater. Photo 4 shows some SMT prototyping boards.

Many surface mount parts don't have room for identification information. It's extremely important to label all the parts' containers as they're received to ensure that you can identify them a month from now. If they come packaged on tape-and-reel strips,

don't remove them from their packaging until you're ready to solder them.

When a new part comes in, I like to look it up in a databook and attach a copy of the pinout to the part's packaging. I can get right down to work when building a prototype and not have to waste time looking up part's. Beware: the same part may have different pinouts for each package style.

A 1" x 2", single-sided pattern PCB costs \$7. You need both 0.031" and 0.020" solder. The 0.020" solder sells for \$15 and solder cream (25-grain syringe) is \$22.

NOW IT'S YOUR TURN

In some instances, the investment in prototyping tools is small. But, if you choose the SMT venue, more sophisticated tools are necessary, especially as you venture toward the finer-pitched components.

So, pull out those dinner napkins and get your design ideas transformed into working models. 📄

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.

SOURCES

Digi-Key Corp.
701 Brooks Ave. South
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

Jameco Electronic Components
1355 Shoreway Rd.
Belmont, CA 94002-4100
(415) 592-8097
Fax: (415) 592-2503

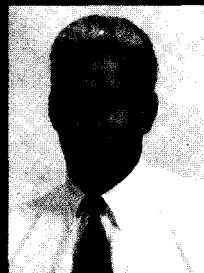
Mouser Electronics
11433 Woodside Ave.
Santee, CA 92071
(800) 346-6873
Fax: (619) 449-6041

I R S

419 Very Useful
420 Moderately Useful
421 Not Useful

Fuzzy PID-Pong

The Final Chapter



In *INK 67*,
Tom told
us the
latest on
Adaptive

Logic's AL220, a fuzzy-logic chip. Last month, he put it through its paces with the PID-Pong machine. This month, he reviews its development tools, ruleware, and performance.

SILICON UPDATE

Tom Cantrell



Silicon Update (*INK 67*) introduced the AL220 fuzzy logic chip. *INK 69*

continued with a description of the PID-Pong machine and an AL220-based controller (Figure 1). And, this month, I'll finish by looking at the AL220 development tools and procedure, describing the PID-Pong control ruleware, and assessing its performance.

I've devoted an unprecedented three articles to the subject in order to satisfy my own curiosity (and hopefully yours, too) about whether fuzzy is finally ready for prime time or yet another Tomorrow's Technology of Tomorrow.

I also wanted to devote enough space to completely dissect a real application and avoid the handwaving that characterizes the all-too-typical smoke-and-mirrors articles on fuzzy logic. Based on the results, you can draw your own conclusions, but nobody can assume, hype, or hope away the facts of the matter.

Before the PID-Pong balls start flying, let's take a look at the tools and techniques for developing an AL220-based application.

INSIGHT PITCH

The Insight IIe package (Photo 1) bundles all the hardware and software you need to take an AL220 from shipping tube to socket. Considering the package includes the equivalent of a compiler, simulator, full-speed emulator, and device programmer, the \$595 tag seems eminently reasonable.

The process of developing "ruleware" for the AL220 consists of four basic steps:

- assign the I/O
- define the fuzzy variables
- specify the membership functions
- enter the rules

Let's step through the sequence screen by screen, using snippets of the PID-Pong program to elaborate.

Photo 2a shows the I/O screen which defines the name of each I/O pin. Inputs and outputs are simply assigned to pins in order of entry—the first input becomes AINO, the third output AOUT2, and so on. Changing assignment (e.g., to optimize PCB layout) is simply a matter of using the Move option to shuffle the order.

Remembering that a fuzzy variable relates an input to a membership function, Photo 2b shows how each input is characterized in terms of multiple membership functions and how the membership functions themselves (i.e., center, width, shape) are defined. This particular screen shows that the controller's goal is to position the ball within ± 7 (i.e., $1\frac{1}{2}\% \pm 3\%$ accuracy) of the setpoint.

Once the I/Os are assigned and the variables [input and membership function pairs) are defined, the rules that control the behavior of the system [i.e., output action) are typed in as shown in Photo 2c. The highlighted rule says to decrease the fan power by 4 (i.e., $\frac{4}{256}$ 1.5%) if the ball position is $V(\text{ery})\text{HIGH}$.

After the rules are entered, you're left with a screen (Photo 2d) which shows the pinout and lists the rules. Notice that the signals shown as inputs to the chip include the outputs, which are in parentheses to indicate they are fed back internally.

From this point, development flow continues much along the lines of a micro—the application can be simulated on the PC or downloaded to the emulator (via printer port) for testing.

One welcome difference is that there aren't any slow compile or download cycles. Simply saving a program automatically produces the emulator-digestable **.HEX** file which is only a few hundred bytes long and downloads in a couple of seconds.

When simulating a design, a key question is: Where do the inputs come from?

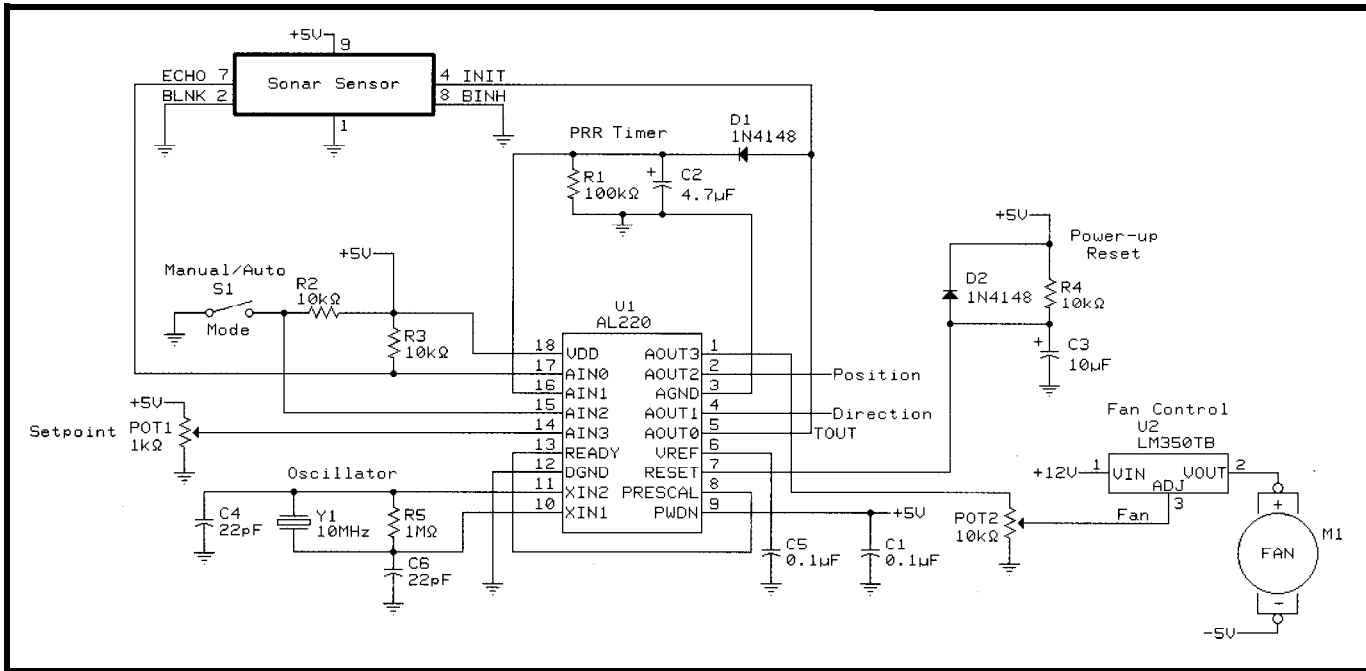


Figure 1-As we saw last month (INK 69), the AL220 was a snug fit, but it proved up to the PID-Pong challenge.

Insight offers two options for describing an input sequence. First, a text file can be prepared that simply lists a stream of input values to apply frame by frame (remember, the AL220 processes all inputs, rules, and outputs in 1024 clock chunks).

Alternatively (the approach used for the PID-Pong simulation), an equation editor (Photo 3) enables you to model the inputs as a function of time. You can model the outputs using a complete selection of numeric operators including trig, logarithm, exponentiation, square root, and so on.

Of course, it's unlikely a process model will be totally accurate (if it is, it may be amenable to conventional control), so the goal of simulation is mainly to catch major gotchas. Indeed, it may be helpful to simulate portions of the code piecemeal in order to zero in on a particular bug.

Simulation (and emulation as well) can proceed rule by rule, frame by frame, or continuously. The simulation results can be viewed immediately or, since that slows everything down to the PC screen update rate, captured for later viewing.

The captures are in a variety of formats, the most useful being a line graph showing the state of all the various inputs and outputs over time (Photo 4).

Once simulation shows you're in the ballpark, the code can be downloaded to the emulator for in-system exercise. The emulator offers a variety of setup options (Photo 5) including internal or external clocking, start and stop modes, and selective I/O logging.

The latter option can be used to limit what's viewed (using the same facilities as the simulator) to I/Os of

interest, thereby reducing the data transfer overhead between the PC and emulator.

THE GAME PLAN

Without further ado, Listing 1 shows the AL220 PID-Pong control program, comprising the aforementioned I/O assignments, fuzzy variable definitions, and rules.

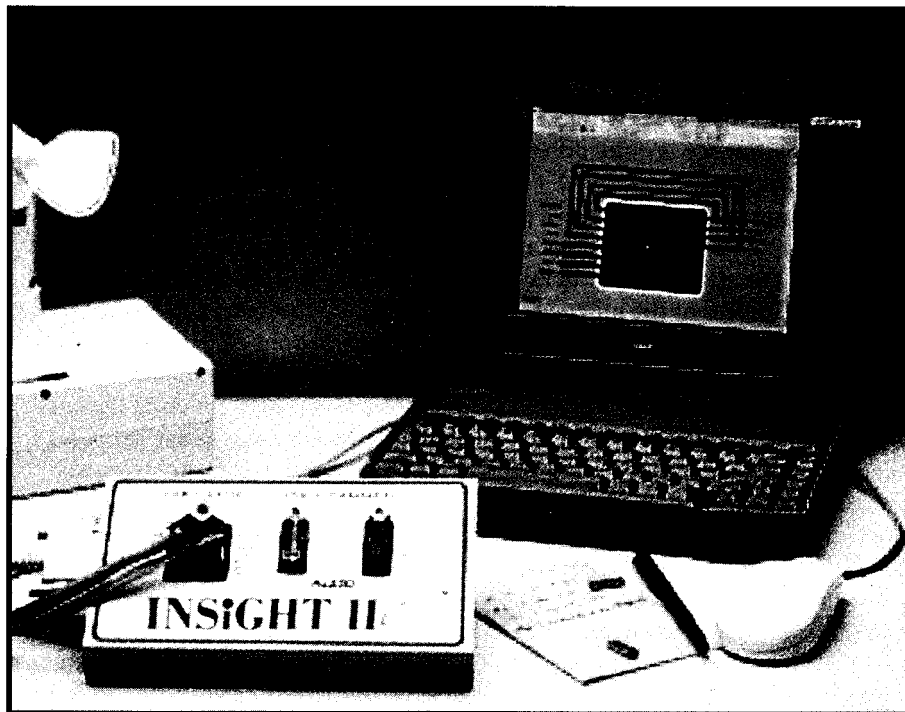


Photo 1—The Insight II development package includes Windows-based development software and a combined emulator and programmer that connects to a PC printer port.

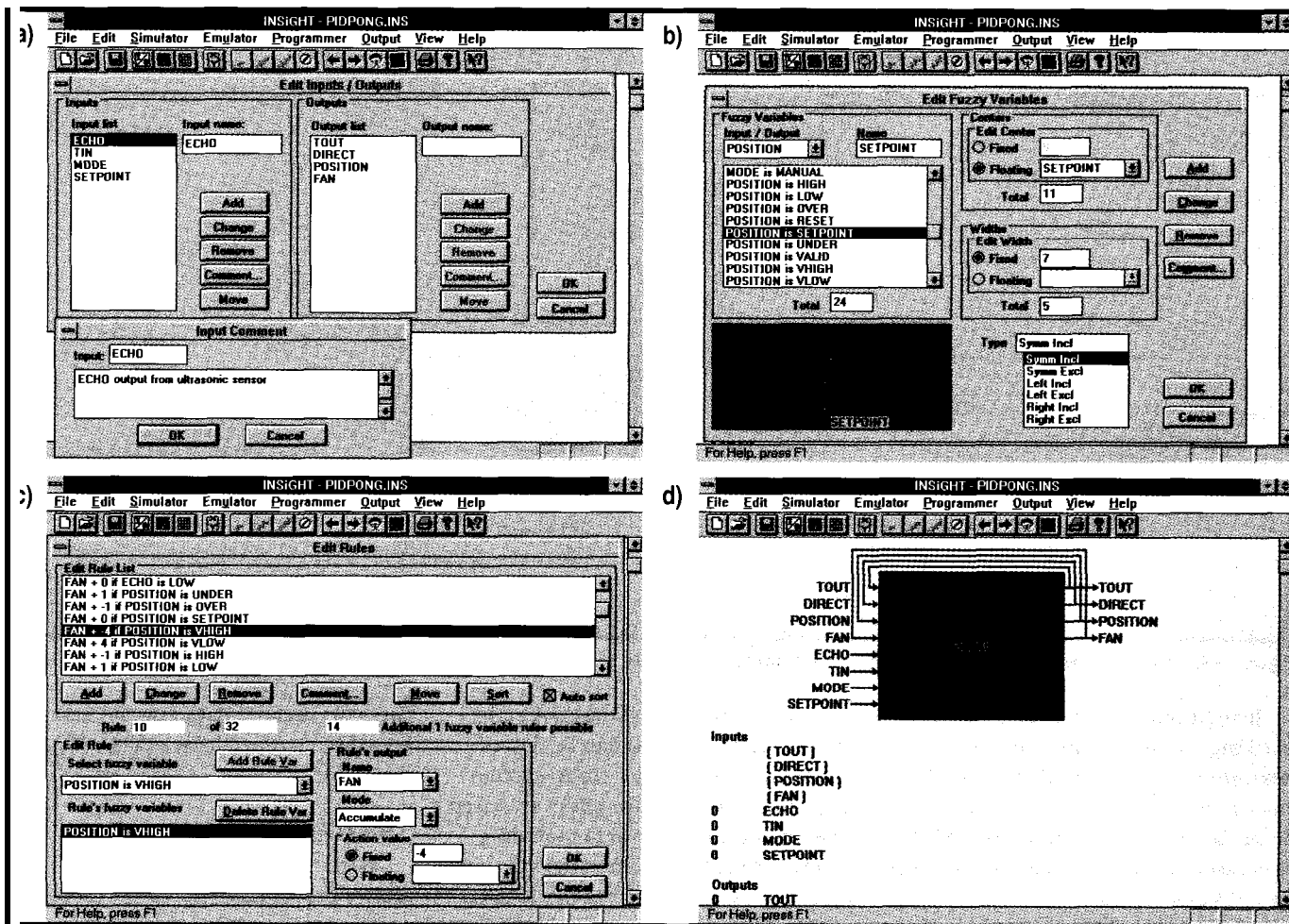


Photo 2—Sample screen shots show the major steps in AL220 software development: a) I/O assignment, b) fuzzy variable definition, c) rule entry, and d) ready for simulation or emulation.

Rather than give a rule-by-rule account, it's a little easier to explain the big picture, calling out segments of the program as needed. Remember, the AL220 (executing frame-by-frame) concept of sequence is quite different than that of a micro.

The grouping of rules affects a given output more than the ordering of rules within the group. Thus, we can examine bits and pieces of the program somewhat independent of their position in the file.

With that in mind, you can see from the listing that the rules affecting the fan output are split into two groups: 3-12 and 16-29. This means two rules, one from each group, update the fan output in a single frame. As you'll see, these rules kind of correspond to the proportional and derivative terms of a conventional control strategy.

The basic control rules are of the archetypical fuzzy variety. Rules 9-12

apply a first order of control simply based on the position (**VH I GH**, **H I GH**, **LOW**, **VLOW**) of the ball relative to the setpoint.

Along the same lines, rules 18 and 20 apply coarse adjustment anytime the ball is moving fast (i.e., **FDOWN** and **F U P**), a situation that should be dealt with rather quickly. Finally, rules 19-29 (minus the previously mentioned rule 20) all fine tune the control based on combinations of the position, speed (i.e., derivative), and direction.

The other rules affecting the fan output handle various special cases such as initialization and manual mode (i.e., the potentiometer doesn't adjust the setpoint, but simply controls the fan voltage directly).

Notice how rules 3 and 16 duplicate each other. No, it isn't a typo. Rather, the duplication ensures that when manual mode is in effect, it overrides both groups of basic control rules. The same logic applies to duplicated rules 5

and 17 which suppress fan control between sensor updates.

The remaining rule groups (rules 0-2, 13-15, and 30-31) deal with the TI01 interface and the PRR (process repetition rate) timer based on the RC (R1 and C2 on the schematic) delay. The AL220 kicks things off by driving **TOUT** high (rule 13) which, with **TOUT** connected to the TI01 **I N I T** input, issues a ping and starts charging the RC.

Having cleared it with rule 1, the AL220 starts incrementing the **POS I - T I O N** counter (rule 2) until the **T I N** input (connected to the RC driven by **TOUT**) reaches a value (**TMAX**) corresponding to the TI01 blanking interval (rule 1).

At this point, the **P O S I T I O N** counter is reset again (rule 0). Notice how this time, it is reset to 2 rather than 0. This is a trick (we'll see it used again shortly) to enable a single variable to differentiate between two states (i.e.,

the blanking and active intervals). The goal of all this is to efficiently apply the AL220's 8 bits of resolution to measure only the active (i.e., beyond the blanking interval) region of the ultrasonic sensor.

At this point, rule 14 kicks in to change TOUT from 255 (TMAX) to 250 (TH I). As in the previous example, the multistate trick is used again to get the most bang out of the fewest bits. The TI01, being a lowly digital device, won't even notice that the I N I T input has gone from 5 V (TMAX, 255) to 4.9 V (TH I, 250), but the AL220 takes note of the fact to dismiss rules that look for TOUT = TMAX.

Eventually, RC gets fully charged (i.e., TIN = TMAX) and the ultrasonic sensor replies (ECHO = HIGH), which causes rule 15 to fire. This step completes the sensor cycle by driving TOUT (and thus the sensor I N I T line) low, which also starts discharging the RC. Since TOUT is now 0, rather than TH I, the P O S I T I O N counter update via rule 2 stops, latching the ball position.

That leaves only rules 30 and 31, which deal with the D I R E C T variable that is used to determine both the direction and speed (i.e., derivative) of the ball. Here's how it works.

So far, we've seen rules that look for the RC to be fully charged (i.e., T I N is

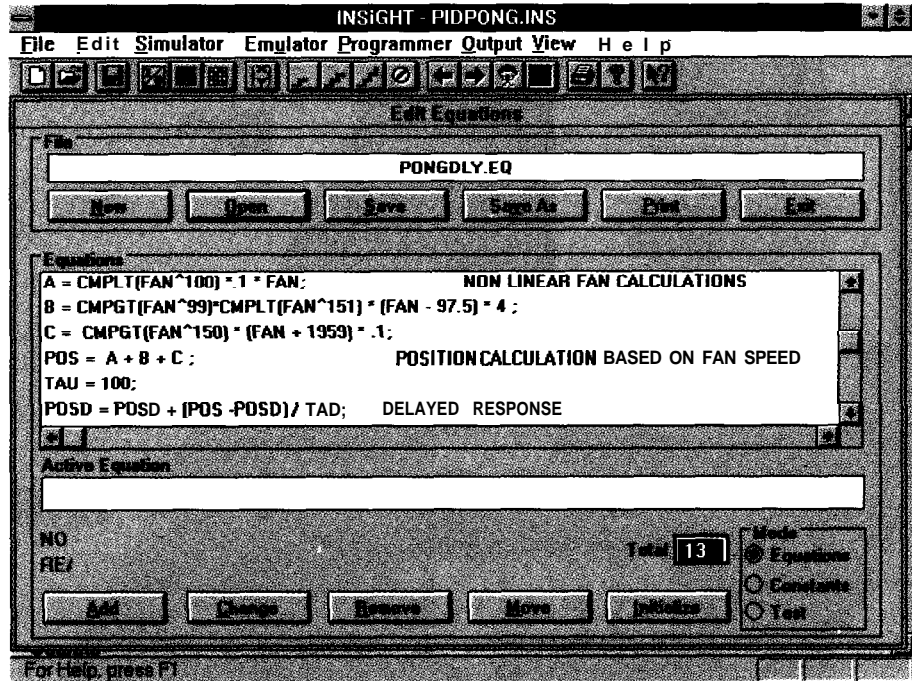


Photo 3—For simulation purposes, inputs can come from a file or be derived from equations.

TMAX) and fully discharged (i.e., T I N is TMIN). Relying once again on the analog capabilities of the AL220, a third intermediate level is defined as S A M P L E (see fuzzy variable definition 1).

When the RC on T I N discharges to the SAMPLE level, rule 30 stores the value of POSITION in DIRECT. However, at this point the AL220 has already used P O S I T I O N and the previous

value of DIRECT. Thus, DIRECT can be considered the previous P O S I T I O N for purposes of calculating the direction and derivative. The calculation is embodied in the fuzzy variable definitions 15-20 which use the floating-membership feature to compare D I R E C T (i.e., the previous P O S I T I O N) with the current POSITION.

Meanwhile, rule 31 simply initializes the previous position to a known value (SETPOINT) the first time around.

YOUR SERVE

Enough with the preliminaries. Let's see if the AL220 is up to a little friendly game of PID-Pong.

Well, maybe not so friendly. Some of you may still be thinking the PID-Pong machine is some kind of fall guy, little more than a warmup for a big-league control challenge.

I've tried previously to convey in words how difficult the control problem is but at this point, a picture (or, rather, a graph) will do a much better job. Take a look at Figure 2, which shows the ball position at a constant fan setting.

Wow, the ball seems to cha-cha-cha all over the place with a mind of its own! Beyond what looks to be about 25% or so noise in the response, check

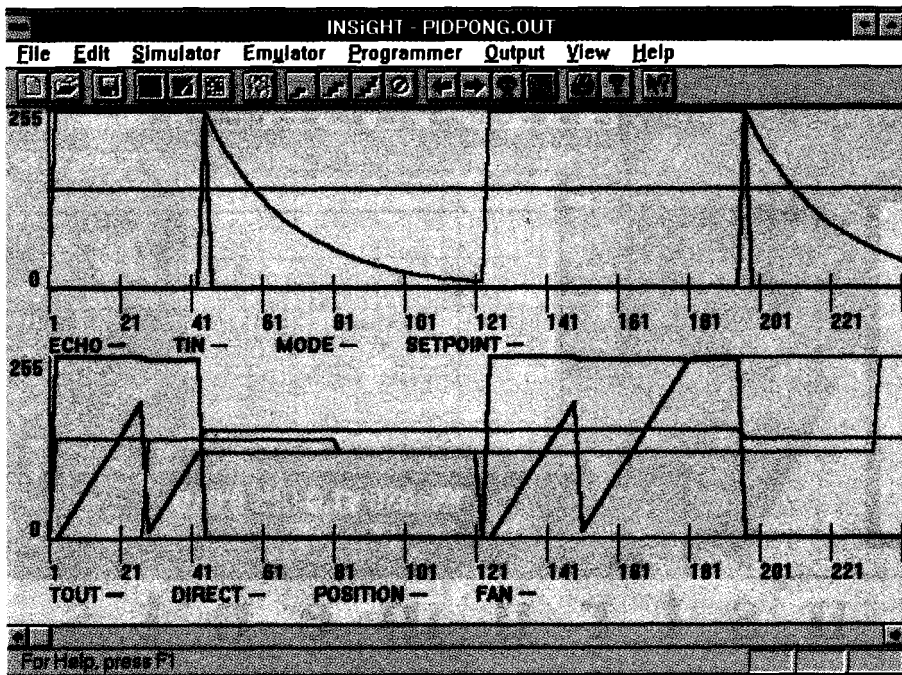
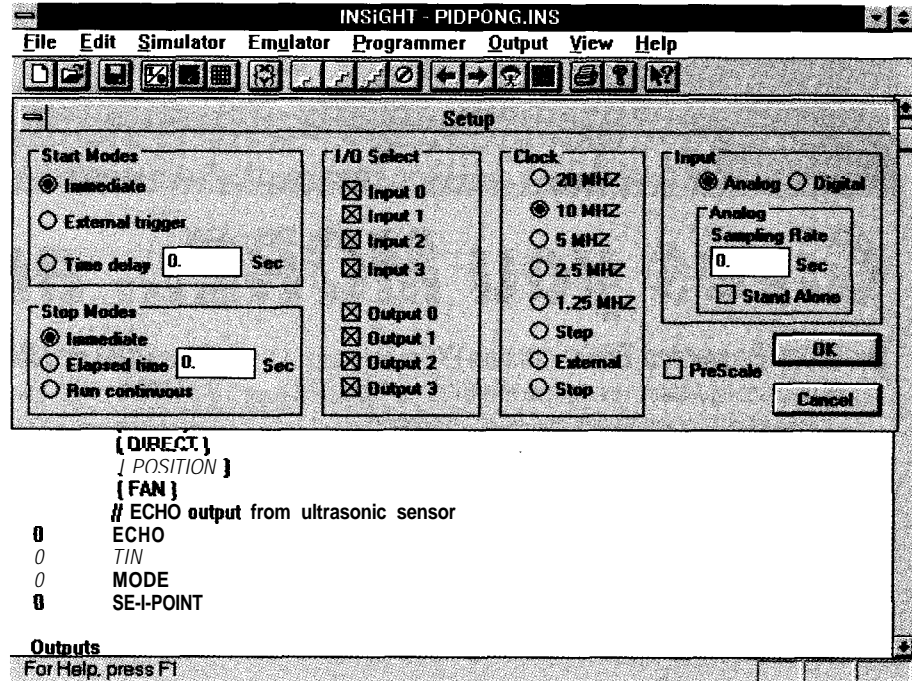


Photo 4—Output from the simulation can be viewed in a number of formats including line graphs.

Photo 5—The emulator features a variety of trigger, capture, and clock options.



out the radical glitches. That sound you hear is all those folks who thought the PID-Pong machine would be a pushover, scrambling to make an emergency call to their bookie.

With Figure 2 in mind, now take a look at Figure 3, which shows the step response of the AL220 PID-Pong machine combo as it volleys the ball back and forth between high and low setpoints. Not bad, eh?

The setup achieves pretty good stability and accuracy, say $\pm 5\%$. Notice the derivative action at work, as evidenced by the fan-power slope transitions leading the setpoint/position equilibrium. For instance, at the very left edge of the curve, notice how the power peaks and starts to back off well before the ball reaches the setpoint.

Indeed, though not shown in Figure 3, the controller exhibits a pseudo-integral action as well. For example, you can lay a pencil over the top of the tube to represent an unexpected load

shift and, assuming you don't overdo it (i.e., there's fan power in reserve), the AL220 deals with it by increasing the necessary power to achieve the goal.

By contrast, a conventional P or PD controller leaves the ball hanging a bit (determined by relative magnitude of the unexpected load) short of the setpoint. For instance, an equation of the form:

$$\text{Fan Power} = (\text{P_gain} \times \text{Error}) + (\text{D_gain} \times \text{dERROR/dt})$$

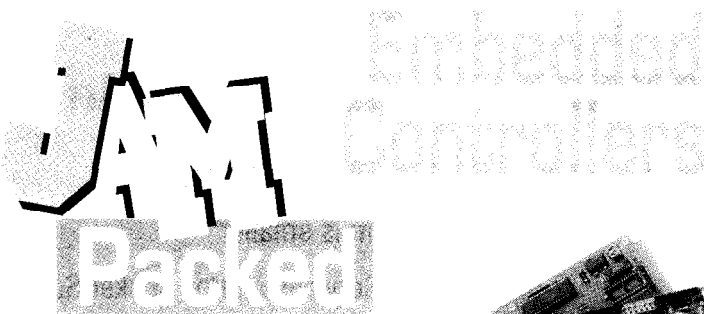
won't change the fan power in a situation where the ball is resting [i.e., $\text{dERROR/dt} = 0$] below the setpoint.

POSTGAME REPORT

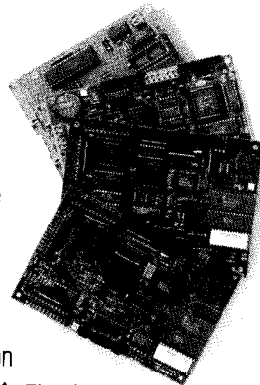
Let me sum up this saga with my own conclusions. Just remember, thanks to the fact that everything is laid bare, you can (and should) make your own judgments without the usual dose of blind faith.

As for the basic control of the PID-Pong machine, the AL220 does a good job, though admittedly it doesn't match the performance (speed, stability, accuracy) of the original floating-point PID controller (INK 50).

Of course, this is an apples-and-oranges comparison in a number of ways. Notably, the AL220 does the job with several hundred thousand fewer



- ◆ A/D inputs, 12-bit accuracy
- ◆ Analog outputs
- ◆ Relay control
- ◆ Counter/Quadrature encoder inputs
- ◆ Buffered RS-232/485 serial ports
- ◆ Operator interface via keypad and LCD display
- ◆ Program using a PC
- ◆ 512K program, 512K data memory
- ◆ 5V only operation
- ◆ Built-in BASIC supports all on-card hardware
- ◆ Floating point math
- ◆ From \$195 in 1's



REMOTE™
PROCESSING

The embedded control company

Call for more information and FREE
Catalog of embedded controllers:

Running under Windows?

Using a standard data acquisition board is like wing an old typewriter. They both get the job done, but ... there is a better way.

Standard data acquisition boards can unknowingly sabotage your data. Ensure the integrity of your results.

ADAC's Windows Optimized 5800 Series gives you the resources you need: FIFOs, Channel Gain RAM, Dual DMA, aggressive prices, and some of the best noise performance in the industry!

5801MF: 16 channel 12-bit A/D, 333KHz, 2 16-bit D/A, 40 digital I/O

5803HR: 16 channel 16-bit A/D, 100KHz, 2 16-bit D/A, 40 digital I/O

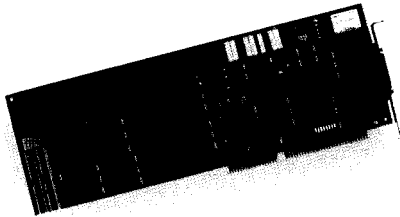
learn more -

voice **800-648-6589**

fax **617-938-6553**

web **<http://www.adac.com>**

email **info@adac.com**



ADAC

American Data Acquisition Corporation
70 Tower Office Park, Woburn, MA 01801 USA

Listing 1—The AL220 PID-Pong control program

I N P U T S

(TOUT)
(DIRECT)
(POSITION)
(FAN)
ECHO
TIN
MODE
SETPOINT

O U T P U T S

TOUT
DIRECT
POSITION
FAN

FUZZY VARIABLES

FuzzyVar No. 0
MODE is MANUAL (10, 0, Left Inclusive)
IF MODE IS LOW MANUAL MODE (FAN = SETPOINT)

FuzzyVar No. 1
TIN is SAMPLE (50, 5, Symmetric Inclusive)
SAMPLE DIRECT TIME

FuzzyVar No. 2
TIN is TMN (10, 0, Left Inclusive)
PRR TIMER CAP DISCHARGED

FuzzyVar No. 3
TIN is TMAX (200, 0, Right Inclusive)
PRR TIMER CAP FULLY CHARGED

FuzzyVar No. 4
ECHO is LOW (100, 0, Left Inclusive)

FuzzyVar No. 5
ECHO is HIGH (100, 0, Right Inclusive)
SONAR ECHO HIGH OR LOW

FuzzyVar No. 6
POSITION is RESET (2, 0, Symmetric Inclusive)

FuzzyVar No. 7
POSITION is VALID (192, 0, Right Inclusive)

FuzzyVar No. 8
POSITION is UNDER (5, 0, Left Inclusive)
POSITION OUT OF LINEAR SONAR REGION

FuzzyVar No. 9
POSITION is OVER (210, 0, Right Inclusive)
POSITION OUT OF LINEAR SONAR REGION

FuzzyVar No. 10
POSITION is VLOW (SETPOINT, 20, Right Exclusive)

FuzzyVar No. 11
POSITION is LOW (SETPOINT, 7, Right Exclusive)

FuzzyVar No. 12
POSITION is SETPOINT (SETPOINT, 7, Symmetric Inclusive)

FuzzyVar No. 13
POSITION is HIGH (SETPOINT, 7, Left Exclusive)

FuzzyVar No. 14
POSITION is VHIGH (SETPOINT, 20, Left Exclusive)
DIFFERENCES BETWEEN POSITION AND SETPOINT

FuzzyVar No. 15
DIRECT is ZERO (0, 0, Symmetric Inclusive)
DIRECT = 0, INITIAL CONDITION

FuzzyVar No. 16
DIRECT is FDOWN (POSITION, 12, Left Exclusive)

FuzzyVar No. 17
DIRECT is DOWN (POSITION, 7, Left Exclusive)

FuzzyVar No. 18
DIRECT is STOPPED (POSITION, 7, Symmetric Inclusive)

FuzzyVar No. 19
DIRECT is UP (POSITION, 7, Right Exclusive)

FuzzyVar No. 20
DIRECT is FUP (POSITION, 12, Right Exclusive)
DIRECTION AND SPEED OF DERIVATIVE F = FAST

FuzzyVar No. 21
TOUT is TMN (10, 0, Left Inclusive)
PRR TIMER LOW

(continued)

FuzzyVar No. 22

TOUT is THI (250, 5, Symmetric Inclusive)
 PRR TIMER HIGH

FuzzyVar No. 23

TOUT is TMAX (255, 0, Right Inclusive)
 PRR TIMER AT MAX

R U L E S

Rule No. 0

If ECHO is LOW and TOUT is TMAX and POSITION is VALID then
 POSITION = 2
 RESET POSITION AT START OF ACTIVE SONAR REGION

Rule No. 1

If ECHO is LOW and TIN is TMIN then POSITION = 0
 RESET POSITION AT START SONAR PULSE

Rule No. 2

If ECHO is LOW and TOUT is THI then POSITION + 8
 COUNT POSITION DURING ACTIVE SONAR REGION

Rule No. 3

If MODE is MANUAL then FAN = SETPOINT
 SETS FAN SPEED TO SETPOINT IN MANUAL MODE

Rule No. 4

If DIRECT is ZERO and TOUT is TMAX then FAN = SETPOINT
 SETS FAN SPEED TO SETPOINT INITIAL CONDITION

Rule No. 5

If ECHO is LOW then FAN + 0
 KEEPS FAN FROM UPDATING UNLESS ECHO IS HIGH

Rule No. 6

If POSITION is UNDER then FAN + 1
 INCREASE FAN SPEED IF POSITION IS BELOW ACTIVE SONAR REGION

Rule No. 7

If POSITION is OVER then FAN + -1
 DECREASE FAN SPEED IF POSITION IS ABOVE ACTIVE SONAR REGION

Rule No. 8

If POSITION is SETPOINT then FAN + 0
 CHANGE FAN SPEED IN RESPONSE TO POSITION

Rule No. 9

If POSITION is VHIGH then FAN + -4

Rule No. 10

If POSITION is VLOW then FAN + 4

Rule No. 11

If POSITION is HIGH then FAN + -1

Rule No. 12

If POSITION is LOW then FAN + 1

Rule No. 13

If TIN is TMIN then TOUT = 255
 START OF SONAR CYCLE

Rule No. 14

If TOUT is TMAX and POSITION is RESET then TOUT = 250
 DENOTES START OF ACTIVE SONAR CYCLE

Rule No. 15

If TIN is TMAX and ECHO is HIGH then TOUT = 0
 DENOTES END OF ACTIVE SONAR CYCLE

Rule No. 16

If MODE is MANUAL then FAN = SETPOINT
 SETS FAN = SETPOINT IN MANUAL MODE

Rule No. 17

If ECHO is LOW then FAN + 0
 KEEPS FAN FROM UPDATING UNLESS ECHO IS HIGH

Rule No. 18

If DIRECT is FDOWN then FAN + 5
 CHANGE FAN SPEED IN RESPONSE TO POSITION AND DERIVATIVE

Rule No. 19

If POSITION is VHIGH and DIRECT is DOWN then FAN + 2

Rule No. 20

If DIRECT is FUP then FAN + -5

Rule No. 21

If POSITION is VLOW and DIRECT is UP then FAN + -2

Rule No. 22

If POSITION is VHIGH and DIRECT is UP then FAN + -2

Rule No. 23

(continued)

IMMEDIATE RESPONSE!



When you've got to get that project done

NOW!!

Our off-the-shelf SBCs help you ship **FAST**

552SBC

- 80C552, a '51 Compatible Micro
- 40 Bits of Digital I/O
- 8 Channels of 10 Bit A/D
- 3 Serial Ports; 232 or 422/485
- 2 PWM channels
- 6 Capture/ Compare Inputs
- Real Time Clock
- 64KB SRAM; 1 UVPRM Socket
- 512B Serial EEPROM
- Watchdog, PFI, Regulation
- Expansion bus for more circuits
- Development ROM Available

Our popular 552SBC-40 starts at just \$299, quantity 1. And we can create a version just for your needs, then pass the savings on!

C/C++ w/188SBC

- 80C188XL: x86 Family Micro
- 16 channel, 12-bit A/D
- 8 Channel, 12-bit D/A
- 24 Opto Rack Channels
- 2 Serial Ports; 232 or 422/485
- Real Time Clock, Watchdog
- 512KB Static, batt-backed RAM
- 2 UVPRM or Flash Sockets
- PFI, On-Board Power supply
- PC/1048-bit Expansion bus
- FPGA Socket and more!

Use our development tools with your MS & Borland compiler. Quantity 1 pricing from \$299 for the 188SBC-10 to \$749 for the -50!

Other products are available. Call Today!

We specialize in custom work too; as few as 25 units. **CALL NOW!**



HiTech Equipment Corp.
 9400 Activity Road
 San Diego, CA 92126
 (Fax: (619) 530-1458)

Since 1983

(6 19) 566-1892



E-mail: info@hte.com - [Ftp: ftp.hte.com](ftp://ftp.hte.com)
 Web: <http://www.hte.com>

transistors than the micro (H8 with 32-KB ROM and 1-KB RAM) used in the previous article. On the other hand, the micro is able to do other stuff too, such as reporting machine status via a built-in UART.

This price/performance tradeoff highlights the fact that you've got to carefully decide just what kind of control is acceptable in your application. I suspect there's a tendency to over-specify, as in using a full-blown floating-point PID to control a hair dryer.

As an aside, noticing (in Figure 3) that less than half the fan dynamic range is exploited, I suspect the AL220 setup is unduly burdened by an overly macho fan. The result is the machine process rate (i.e., PRR timer) has to be limited to avoid wind up and oscillation. A smaller fan might do a better job by enabling the process rate to be sped up and reducing lag, to boot.

Anyway, don't forget there's more to the story than just the results. A complete and fair comparison must also consider how hard it is to achieve them. While it's said that the journey is the reward, remember that you may not get a paycheck until you reach the destination.

Consider that with 32 rules and 24 variable definitions, the AL220 program comprises about a page of code. By contrast, the earlier PID program required a few hundred lines of C and assembly to get the job done. Yes, that's apples and oranges, but even the simplest micro setup would likely call for more code than the fuzzy version,

Listing 1 - continued

```

      If POSITION is VLOW and DIRECT is DOWN then FAN + 2
Rule No. 24
      If POSITION is HIGH and DIRECT is DOWN then FAN + 1
Rule No. 25
      If POSITION is HIGH and DIRECT is UP then FAN + -1
Rule No. 26
      If POSITION is LOW and DIRECT is DOWN then FAN + 1
Rule No. 27
      If POSITION is LOW and DIRECT is UP then FAN + -1
Rule No. 28
      If POSITION is SETPOINT and DIRECT is DOWN then FAN + 1
Rule No. 29
      If POSITION is SETPOINT and DIRECT is UP then FAN + -1
Rule No. 30
      If TIN is SAMPLE and TOUT is TMN then DIRECT = POSITION
      STORES OLD POSITION IN DIRECT, CALCULATES DERIVATIVE
Rule No. 31
      If DIRECT is ZERO and POSITION is RESET then DIRECT = SETPOINT

```

if only to initialize the micro after reset.

Yes, the fuzzy code may look a little weird at first, but let's get real. Only the most die-hard bit-banger could argue that it's less readable than C or assembly. Any thoughts along those lines are surely not due to the fact the fuzzy trick is overly tricky, but rather that the programming dogs are real old.

Indeed, the toughest parts of the fuzzy code are those that force fit the analog AL220 into digital (the ultrasonic sensor) duty. By contrast, the basic control rules are quite straightforward. Compare these to the PID code, which clearly called for quite a bit of scientific and programming know-how.

Don't forget the effort associated with tuning the machine operation. Many hours were spent tweaking the coefficients of the earlier PID loop, only to find that a change in goal (i.e., setpoints), machine response (i.e., fan output increases after it warms up), or environment [air conditioning kicks in, etc.] would goof things up.

To tell the truth, I'm still not sure there aren't pathological cases associated with the original PID code. By contrast, along the lines of the tortoise and hare, the fuzzy version isn't as snappy but seems more robust.

At times, I've heard complaints by some fuzzy-skeptics that they worry about not knowing what's going on inside the chip. Surely, I hope this example helps lay that myth to rest.

First of all, once you get past the AL220 analog facade, everything devolves to good old 1s and 0s. I mean, you can step through a simulation rule by rule and see each result in all its digital glory. It's not like the numbers on the screen are blurry or something.

By contrast, there are plenty of examples of digital systems that compute the wrong answer with great precision! Indeed, I feel I know a lot more about what's going on inside the AL220 than what's going on inside a C compiler.

To sum it all up, I feel fuzzy—at least as embodied in the AL220—surely deserves serious consideration rather than just lip service. These days,

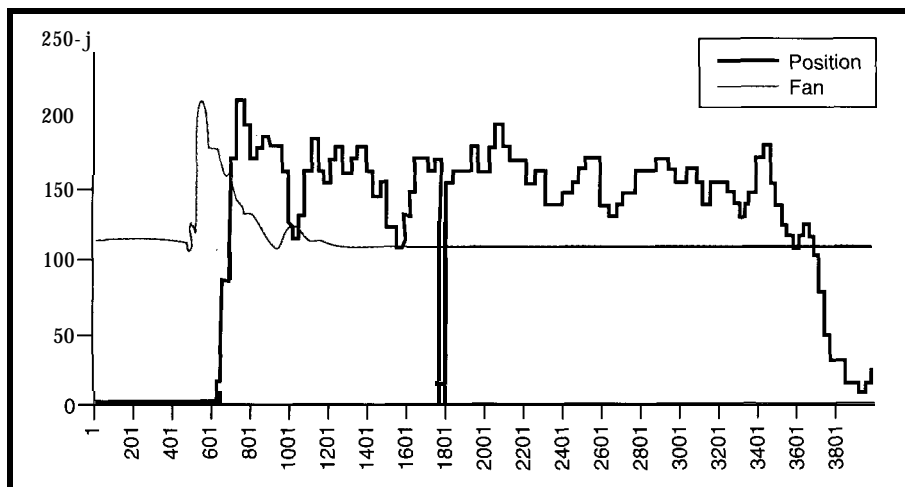


Figure 2—The PID-Pong machine presents a challenging control problem. Even when the fan is at a steady state, the ball's position jumps all over.

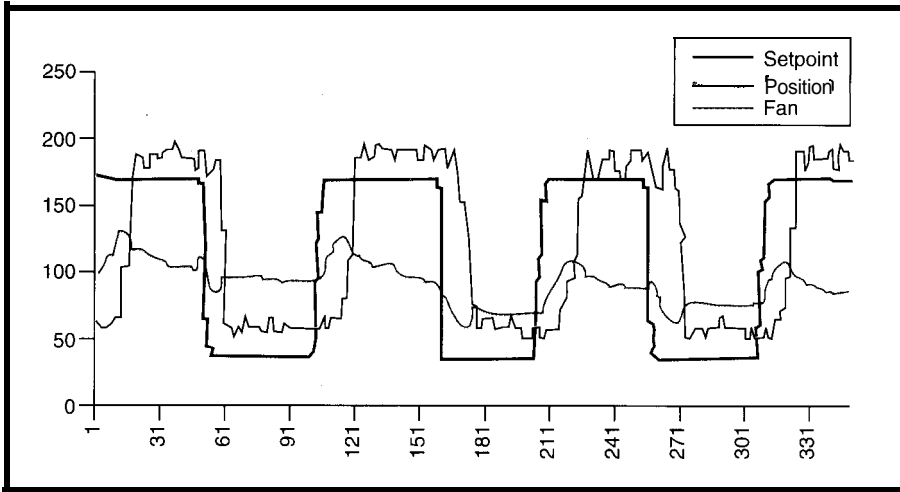


Figure 3—As the ball setpoint keeps changing up and down, the AL220 does a pretty good job of keeping the ball in play.

everybody knows one size doesn't fit all—different problems are best served by different technologies.

My guess is that, if an AL220 can do the job, it can likely do the job easier and cheaper than many of the alternatives. This is especially true for those problems that reside completely in the analog domain, though we've seen the AL220 can play the digital game too.

In fact, a micro-and-AL220 combination might be just the ticket for mixed-signal applications.

Is there any doubt that fuzzy logic and the AL220 should be added to your quiver of tools? I invite naysayers to meet me on the PID-Pong court. Bring along your favorite micro, DSP, PLD, op-amp, or whatever, and may the best chip win. ☐

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 by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264 or by fax at (510) 657-5441.

CONTACT

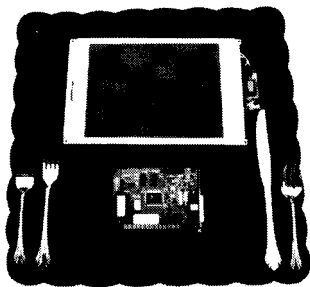
AL220
Adaptive Logic, Inc.
800 Charcot Ave., Ste. 112
San Jose, CA 95131
(408) 383-7200
Fax: (408) 383-7201
<http://www.adaptivelogic.com/>

TI01 Ultrasonic Transducer
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

Flat Panels Served Here



House Specialty - Monochrome LCD Kit \$199

NEW LOWER PRICES!

VGA LCD Controllers for PC ISA Bus

Earth LCD/M Monochrome LCD Controller \$99*
EarthVision/ISA Color LCD Controller \$249*
*When purchased with LCD

Display Kits Include LCD, Controller, & Backlite

Monochrome 9.4" \$199
Color Single Scan 8.2" \$399
Color Dual Scan 9.4" \$649
Color 9.4 Active Matrix \$095

Displays Only Prices start at:

Mono VGA LCD \$49
Color VGA Single Scan \$99
Color VGA Dual Scan \$299
Color VGA Active Matrix \$599



"The Flat Panel Solutions Company"

P.O. Box 7089 - Laguna Niguel - California - 92607
h: (714) 448-9368 - Fax: (714) 448-9316 - <http://www.flat-panel.com>

LCD
RELAYS
ADC
DAC
RS232
RS485
SRAM
EPROM
FLASH
EEPROM
RTC
TIMERS
BATTERY
WATCHDOG
ENCLOSURES
SOFTWARE
PIO
CORES
DMA

Control Practically Anything. from \$79.

Little Star, \$195
30 I/O Lines

Our miniature controllers are ideal for machine control and data acquisition. Easy to program with our Dynamic C. Start saving time and money. Call for your free demo disk today!

1724 Picasso Ave.
Davis, CA 95616
916.757.3737
916.753.5141 FAX
<http://www.zworld.com/>

For immediate information, use our 24-Hour AutoFAX. Call 916.753.0618 from your FAX. Request catalog #18.

CONNECT TIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) 871-1988—Four incoming lines
Internet E-mail: sysop@circellar.com

I've selected two threads this month that show just how helpful our BBS users can be when someone has a troublesome problem.

The first deals with figuring out what to do with a dead NiCd pack, Is it worth frying to bring back to life? Read on to find out.

In the other thread, a caller relates some irritating problems he's having with a sensor circuit. Noise in the circuit disrupts things just enough to cause unreliable operation. Other callers offered suggestions until the solution was finally found.

NiCd Rehab?

Msg#: 3887

From: Jefferson Jacobs To: All Users

I have come into possession of an about-one-year-old battery from a Mac Powerbook, discarded by the owner because it would not run his laptop for more than a few minutes.

It occurred to me that it might still be useful to power a small project for an extended period. Markings on the case designate it as model number M5417 and KR18/51NiCd (also a stern warning that it should only be charged with a model M3050 [Mac] charger). My thought was to drain it down completely with a small resistance, and then bring it back up with a constant current.

The only text I have on the NiCds suggests a charge of amp-hour capacity divided by 10 for 14 hours, or a continual trickle at AhC/30–50. Unfortunately, I don't know what the amp-hour capacity is. I am guessing the voltage to be a nominal 6 V (it was reading about 5.5 V when I got it).

Anyone have any insights or suggestions to offer?

Msg#: 3918

From: Ken Simmons To: Jefferson Jacobs

Forget about trying to rejuvenate the pack. If it won't hold a charge for more than a few minutes, it's safe to say one (or more) of the internal cells is shot.

Do this: if you have a Dremel (or similar tool), carefully open the plastic case. You should see a number of individual cells, probably of the sub-C size, probably 4 or 6 cells. Remove these cells, carefully noting where the positive and negative terminals connected to the pack's contacts (apply a piece of masking tape to the positive contact or otherwise denote it for later use).

Go to Radio Shack and get a set of their sub-C high-capacity cells. When you get home, carefully solder the cells together (you may have to rip off the metal straps on the old cells if plain wire won't fit in the pack) and attach the positive and negative pack terminals to the appropriate cells. Close up the pack and voilà—you have a reconditioned Powerbook battery pack!

As for charging the darling, apply a voltage equal to $(1.5 \text{ V}) \times (\# \text{ cells})$. For example, if you have 4 cells, apply 6 volts. Current-limit the supply using a power resistor. Calculate the charging current as $0.1 \times$ amp-hour capacity of the cells (e.g., a 1000-mAh cell needs 100 mA charging current) and use the voltage difference between the supply voltage and the cells' final voltage [e.g., a 4.8-V NiCd pack needs 6 V to charge, the limit resistor will have 1.2 V across it $[6 \text{ V} - 4.8 \text{ V}]$ and the charging current to calculate the resistor's value and power rating.

For what it's worth, I always include a series diode (1N400x) between the supply and the cells for reverse-voltage protection as well as a method of charge stopping when the cells reach full charge (no forward bias of the diode results in essentially no more current flow and no further charging).

I hope this helps.

Msg#: 4146

From: Pellervo Kaskinen To: Jefferson Jacobs

Looks like this topic comes up every few weeks!

1. **You** just cannot expect to restore the NiCd as an assembled package. You can try your luck (short term) with the individual cells inside the pack.

2. The restoration process you can try is intended for burning away some whiskers inside the cells that are shorted. For that, you provide a jolt of 50-200 A for 1-5 ms to a single cell. May take a couple of jolts or more, depending on the size of the cell and other conditions.

3. The whiskers are going to grow back. They will grow faster if the assembled pack is ever loaded so that the repaired cell becomes empty before the others and is driven to reverse polarity by the good cells.

4. At the very best you can benefit from renewed life for a few weeks.

As to the charging, there are dangers as well. The 1.5 times the cell voltage that Ken Simmons suggested is prob-

CONNECT TIME

ably too much. The nominal voltage for a NiCd cell is 1.20–1.25 V and the charging voltage *at the terminals* is 1.35–1.40 V. If the cell capacity is 1 Ah, then the normal charging current is no more than 100 mA. The trickle charge, after the voltage has risen to 1.35 or 1.40 V (varies by brand and size) is 2030 mA. It can be even lower if battery cooling is not good.

Let's make some worst-case calculations: 100 mA at 1.35 V and dropping down to 20 mA at 1.40 V, without any resistance switching scheme, leads to $V_0 = 1.4125$ V and $R_s = 0.625 \Omega$.

The more practical scheme is that the charger has a switching point where it changes over to the trickle current. Before that it should provide the appropriate charging *current*. The switching point to the trickle *current* level is determined either from the temperature of the cell or from the first derivative of the voltage rise versus time. When the first derivative turns negative, it's time to reduce the current.

The commercial charger control chips operate on the derivative principle. Moreover, they are trying to provide constant current rather than constant voltage. With a 1.5 times nominal voltage and a relatively high value resistance, you can approximate the constant current well enough if the battery is close to okay.

By the way, NiMH cells cannot be charged reliably based on the first-derivative polarity reversal. They need a polarity-reversal determining point of the *second derivative*.

Msg#: 4190

From: Ken Simmons To: Pellervo Kaskinen

Have you ever directly measured the terminal voltage of a charging NiCd pack?

I gave the figure of 1.5 V charging voltage per cell because all the 1.2-V NiCds I've dealt with show 1.5 V when charging. Remove the charging current, and the terminal voltage drops back to 1.2 V/cell.

Lead-acids undergo similar treatment—something like 2 V/cell for a 1.5-V/cell lead-acid pack.

Therefore, you need to design your charger to supply the desired charging current (Ah capacity/10 typically) with the 1.5-V/cell charging voltage.

Works every time for me.. .

Msg#: 7615

From: Pellervo Kaskinen To: Ken Simmons

Yes, I have directly measured the cell voltages. Because I have lost my confidence in doing things other than a single cell at a time, I have come up with the numbers I presented. But I still have to emphasize that there are possible differences between brands and sizes of the batteries that I cannot cover.

During charging, the lowest voltage is directly at the terminals. If you measure at the connecting wires, you already get a drop in the contact, although you may be able to ignore the resistive drop in the wire at low currents. Whatever wire there is inside the cell may change the characteristics and cause additional voltage, but few battery makers would deliberately reduce their efficiency with inadequate wire sizes.

If your voltages work for you, fine. I have had a couple of cases that seem to support you, but the majority of the batteries I have dealt with are definitely overcharged by the time you reach 1.5 V. All they do is heat up.

There have been any number of articles in the trade magazines during the past decades telling the story. Some of them are from the battery manufacturers, some are from the semiconductor companies that sell the charger control chips. The second kind are typically available either as reprints or application notes from the same companies.

Msg#: 8166

From: Ken Simmons To: Pellervo Kaskinen

I've done research and lots of reading (*Art of Electronics* being a prime source) and all the sources I've read agree you need a higher potential applied to the cell to adequately charge it (at the proper current, of course!). This level is to guarantee the "stuffing" of electrons into the cell.

For what it's worth, I've directly measured the voltage of the cells while charging as well as after charging (removal from the charger). The NiCds ultimately achieve a 1.5-V terminal potential while charging (starting from roughly 0.9 V or so from the dead condition) and drop to the requisite 1.2 V when you remove them from the charger. Yes, the cells do get warm regardless of the charging current applied (higher applied current = warmer cell, natch!).

In fact, as part of my research, the charging current, as measured on my DMM, was well within the calculated current range of my charging setup even though the terminal voltage was 1.5 V, not 1.2 V as you'd expect.

I know lead-acid batteries [i.e., car batteries] operate similarly, requiring a higher applied potential than the rating of the cell or battery (roughly 0.5 V/cell higher, given the 2-V/cell potential of car batteries. Your mileage may vary.).

I've got some Benchmarq info on their charging-control chips. As I remember, they don't explicitly mention anything about terminal voltages.

Msg#: 8053

From: Larry G Nelson Sr To: Pellervo Kaskinen

I just finished a charger design using a Benchmarq 2003 chip. This works very well and can be very flexible. The chip does all the work and with a little thought you can

CONNECTIME

charge most NiCd and NiMH batteries with a few option jumpers or plug-in headers having various resistors for voltage and current selections.

Motor noise problem

Msg#: 5777

From: Jim Oslislo To: All Users

I am trying to control the motion of a machine powered by 90-VDC motors, but am being thwarted by motor noise getting into the feedback circuitry (I think).

The speed of the motors is controlled by turning an SCR on or off through an optoisolator attached to a microcontroller. The microcontroller is interrupted at line frequency and pulses the SCR a little after the AC line crosses zero.

Feedback from the motor is accomplished with a pair of slotted optical switches monitoring a hole in a shaft-mounted disc on the back of the motor.

The speed-varying circuitry works fine, but there are major inaccuracies in the positioning area. When I use a

scope to look at the output of the optical switches (which are connected to 7414 Schmitt triggers), there is a sawtooth wave of a couple of volts imposed on top of the high part of the square wave. I don't know if this is brush noise from the motors or from the switching of the SCR.

My efforts to eliminate the noise (an EMI/RFI filter on the AC power, MOVs across the motor, and MOVs from the AC lines to ground] have been in vain. I have also tried a snubber across the SCR which also didn't help. I think the solution lies in eliminating the noise, but I wouldn't be surprised if I am barking up the wrong tree. Any ideas?

Msg#: 7200

From: Ken Simmons To: Jim Oslislo

What kind of power supply bypassing do you have! For something like what you're using, I'd suggest hefty filter caps (5000+ μ F) paralleled with 0.01- μ F polyesters. The small caps will act as high-frequency shunts, effectively eliminating any feedback hash on the power rail.

Also, every chip in your circuit should have a 0.01- μ F bypass cap between the V_{cc}/V_{dd} and Gnd/ V_{ss} pins-again for hash removal.

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.

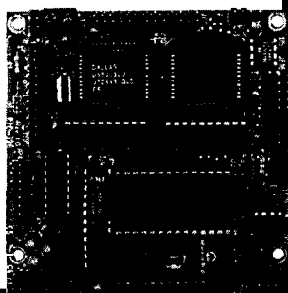
Occupying the same small 3.5"x3.5" RTC footprint and using S-V-only power, the RTC320 uses the new Dallas Semiconductor 80C320, which is 8031 code compatible and 3-5 times faster. At 33 MHz, the RTC320 is an 8-MIPS controller! Along with the new powerful processor, the RTC320 board accommodates up to 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 tiddle 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)



CALL 1-800-635-3355 TO ORDER
MICROMINT, INC.

4 PARK STREET • VERNON, CT 06066 • (860) 871-6170 • FAX (860) 872-2204

#119

Products, Pricing and Service
That's Out of This World...

JAMECO DELIVERS!

Your favorite
Shopping destination

Call for our free catalog

1-800-833-3333

FAX 1-800-833-3333

VIP# 003

#141

CONNECTIME

Msg#: 7541

From: Jim Oslislo To: Ken Simmons

The 7414 off of the slotted switch has a 0.1- μ F cap as do the ICs on the microcontroller board, which is a commercial item. The power supply is just a dinky 9-V wall-plug-in adapter hooked to a 7805 regulator, but there doesn't seem to be much in the way of ripple on the 5-V supply.

The thing I don't understand is how the AC noise is being coupled to the DC circuitry. I've tried to be pretty conscientious about isolating the two, but the spikes on the encoder outputs look like they have a 60-cycle aspect to them.

Msg#: 7566

From: Ken Simmons To: Jim Oslislo

Try putting a 0.01- μ F cap directly across the 7805 output. You might be getting intermittent oscillation out of it.

Another thing to try: put a choke between your motor feeds and the controller board. Maybe that'll dampen things a bit. Do you, perchance, use optoisolators to couple the board to the motor? Maybe you need some direct isolation like that as well as a totally separate supply for the decoder?

Msg#: 7553

From: Brad Sanders To: Jim Oslislo

What is the typical output frequency of the encoder? Is this slotted switch actuated once, twice, or more per revolution?

If the pulses are just sloping (not spikes, but a slope), then I wouldn't sweat it, since 7414s tend to have this as a "feature." They are Schmitt triggers, and 7414s don't have CMOS outputs. As long as they stay at valid logic levels, I wouldn't sweat it. If the outputs are "swinging" at 60 Hz, or going below and above valid logic levels, perhaps you've gone a bit too far in isolating things and have capacitively coupled things, but have no DC ground path?

Now, if they are spikes at 60 Hz, perhaps it's coming from somewhere else? If the problem is caused by an AC motor with brushes, the spikes should be appearing at a much higher frequency than 60 Hz.

Msg#: 7631

From: Jim Oslislo To: Brad Sanders

Here is a little more info about my setup. There is one slotted hole in the disc on the motor with two optical

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 8051/8031 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

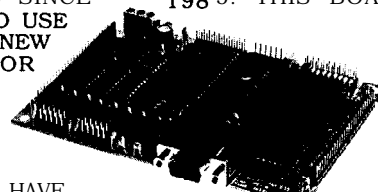


Binary Technology, Inc.
P.O. Box 541 • Carlisle, MA 01741



SINGLE BOARD COMPUTER

THAT'S RIGHT! \$129.95 FOR A FULL FEATURED SINGLE BOARD COMPUTER FROM THE COMPANY THAT'S BEEN WILDING SBC'S SINCE 1985. THIS BOARD COMES READY TO USE FEATURING THE NEW 80535 PROCESSOR WHICH IS 8051 CODE COMPATIBLE.



ADD A KEYPAD AND AN LCD DISPLAY AND YOU HAVE A STAND ALONE CONTROLLER WITH ANALOG AND DIGITAL I/O. OTHER FEATURES INCLUDE:

- UP TO 24 PROGRAMMABLE DIGITAL I/O LINES
- 8 CHANNELS OF FAST 8/ 10 BIT A/D
- OPTIONAL 4 CHANNEL, 8 BIT D/A
- UP TO 4.16 BIT TIMER/COUNTERS WITH PWM
- UP TO 3 RS232/485 SERIAL PORTS
- BACKLIT CAPABLE LCD INTERFACE
- OPTIONAL 16 KEY KEYPAD & INTERFACE
- 192K OF MEMORY SPACE, 64K INCLUDED
- 8051 ASSEMBLER & MONITOR INCL., BASIC OPT.



ENAC, inc.

61 E-529-4525 Fax 457-0110 BBS 529-5708
P.O. BOX 2042, CARBONDALE, IL 62902

CONNECTIME

switches. When going in one direction, only one switch is on; in the other direction, both.

As a test, I turned on the motor in one direction and watched for a transition on one of the switches and then checked the other. In the case where the level was low on the other switch, the counting was correct, (i.e., the level was low in all cases, no erroneous high levels were detected). However, going the other way, when a high level was expected on the other switch, maybe 1 out of 20 levels were low; not the kind of error rate I had in mind.

On the scope, the switches' TTL outputs are solid and stable, but the noise, which looks like three small triangles about 1 or 2 V on top of the 5 V of the high level, walk across the high level of the signal. The low level shows no sign of the spikes.

I don't remember if I stated previously that the SCR was driven through a Darlingon optoisolator, but I don't think the noise is coming through there.

Msg#: 8208

From: Brad Sanders To: Jim Oslislo

Check your optoisolators. Have you tried replacing

them? It sounds like one is either faulty, or just improperly biased.

Msg#: 9246

From: Jim Oslislo To: Brad Sanders

I was happy to hear this suggestion because I hadn't really considered this. To check out the optical switch circuit, I spun the encoder by hooking a battery-powered drill to the motor's armature. Unfortunately (or fortunately), the Schmitt trigger output was rock solid on the scope, and the computer tracked changes in direction without any miscounts. Nuts.

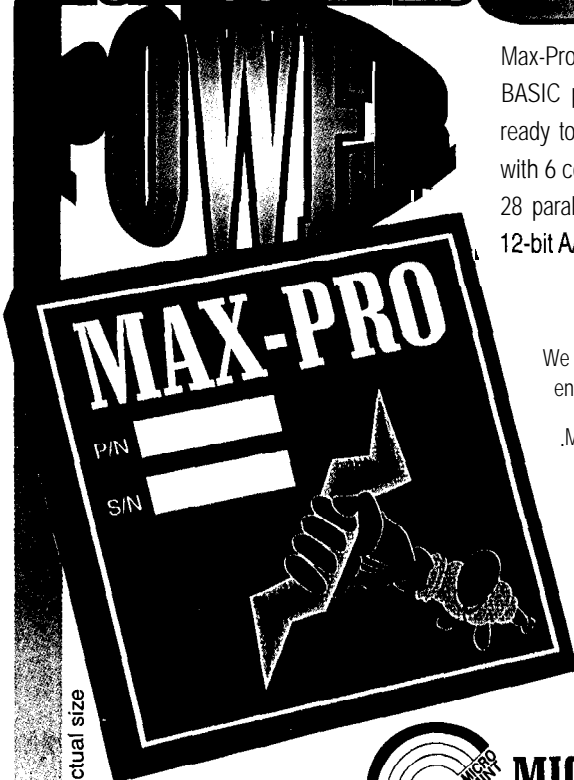
It looks like my original theory of noise from the AC circuitry gumming up the works may be right. The encoder is right behind the motor's brushes and power leads. Do you think the noise is being transmitted because of the close proximity?

Msg#: 9996

From: Brad Sanders To: Jim Oslislo

Could be. Have you tried a grounded foil shield?

ENCAPSULATED Programmable Multitasking BASIC



actual size

MAX-PRO

P/N

S/N

MICROMINT

POWER

Max-Pro is Micromint's latest miniature encapsulated C-language, assembly, or BASIC programmable controller. Max-Pro comes as a no-option, full-featured module ready to assume the toughest embedded tasks. Packed inside is a 9-MHz processor with 6 counter/timers, 3 serial ports (two RS-232, one RS-485), 128K bytes SRAM, 28 parallel I/O lines, hardware real-time clock, and an 8-channel, multiranging 12-bit A/D. Simply plug in a development system EPROM or compiled code and go.

ALL THIS FOR ONLY \$259 COMPLETE!

We offer two PC-based software development packages to make your software environment easier:


- Multitasking BASIC compiler: super fast, 32 simultaneous task compiler that creates code like an interpreter but executes it with the speed of a compiler.
Max-Pro BASIC \$149.00
- C-language compiler: provides a convenient code development environment that is the next best thing to straight assembly code.
Max-Pro C call for price

ORDER CALL 1-800-636-6363

MICROMINT, INC.

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

MAX-PRO



CONNECTIME

Msg#:10239

From: Jim Oslislo To: Brad Sanders

I have thought about shielding, if that's what you mean. The trouble is, I wasn't sure exactly what to shield. Is it the motor leads, the signals from the optical switches, the switches themselves, or a box around the motor switching circuitry? Should the microcontroller be on a separate board than the switching circuitry? This is making me crazy. Nothing I do seems to make any change in the noise.

Msg#:10726

From: Brad Sanders To: Jim Oslislo

That's your mission. Relocate the circuitry with a shaft. See if the noise is being conducted electrostatically, electromagnetically, or directly through some wiring.

Have you thought about hitting the library? A course-book on controls wiring would probably help you, as would the Intel microcontroller handbooks (which have app notes on noise and shielding techniques).

If it's always three triangular waves on the crests of the encoder outputs, it's probably electrostatic conduction. Think about it: when the switch is closed, the noise is

shunted; when the switch is open, the "one" isn't being properly pulled up. Have you tried decreasing the impedance of the pullup? Adding a higher value cap across the switch?

You really should track down the books.. ..

Msg#:10751

From: Ken Simmons To: Jim Oslislo

Signals from the encoder outputs should be transmitted via shielded twisted-pair for maximum noise immunity. Note: Do *not* allow the AC ground and your system's ground to come in direct contact or you'll have more noise than you can possibly deal with!

Msg#:11300

From: Jim Oslislo To: Ken Simmons

Shielding the encoder outputs did the trick. I *thought* the solution to the problem was going to be something simple, but I was so concerned with eliminating the noise from the generation end that I neglected the receiving end. Thanks for the advice.

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..

**C
D
W**

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

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 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) 8752199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

425 Very Useful

426 Moderately Useful

427 Not Useful

PRIORITY INTERRUPT

An Inventing Experience



t

his past weekend, I had a very positive experience with a dedicated group of science-minded students. Thanks to Jake Mendelsohn, I had an opportunity to judge the Connecticut Invention Convention. It took being involved with this independent nonprofit organization whose goal is promoting critical and creative thinking to redeem my faith in the educational experience. Despite the mind-numbing political correctness and lowest common denominator approach of public education, perhaps the brightest can still succeed.

The Convention promotes invention contests among the participating schools and has a final state-wide contest to select grand winners. Because a school's participation is voluntary, however, some of the students who might benefit most were not included. For whatever reason, virtually none of Connecticut's large city school systems participated.

On this occasion, 600 young inventors from grade school through high school presented their projects to a collection of patent attorney, engineering, and technical career judges. The inventions had to be real, solve a problem, and serve a genuine purpose. Little consideration was given to cost or patentability. With such a broad propose, you can guess the variety of entries.

The single most popular subject dealt with mailboxes. It seems kids will do anything to avoid a safari to the roadside mailbox. There were automatic mail-delivery announcers and flashers, pneumatic tube systems, rope-and-pulley mail boxes, swiveling mailboxes, and practically everything short of a rocket-delivery system.

Among the other entries were a Diet-Drink Detector, Robot Dog Feeder, Super-duper Tissue Loader, **2-Minute** Toothbrushing Timer, Under-the-Counter Cereal Dispenser, Automatic Tea-Bag Squeezer, Biodegradable Golf Tee, Squirrel-proof Bird Feeder, Pool Scrubber, Fish Alarm, E-Z Sock Sorter, Lighted Door Knob, Automatic Music-Page Turner, Easter-Egg-Painting Lathe, Hamster Treadmill, and Heated Placemats. One of the inventions I liked was a Christmas tree stand with a water-level indicator and **4-foot** filling tube complete with funnel.

Because there is so little exposure to electronics in our school systems (I don't count using the school's "computer" lab as learning electronics), very few of the inventions involved technology beyond electrical or electromechanical. However, the more precocious older students, some of whose inventions had an energy appetite requiring perpetual motion to be remotely cost effective, exhibited some truly masterful footwork when discussing the practicality of their entry with a knowledgeable judge.

One such invention was a driveway deicing/snow-removal system. It incorporated a grid of resistance heating wires in the pavement. Since a real model was impossible, a battery-powered miniature version was submitted.

When quizzed about whether he had calculated how many hundreds of kilowatt hours or the cost of melting a foot of snow on a real driveway, his answer was reminiscent of my experiences before judging the CIC. He said he didn't care how much it was going to cost. His technique would work and that was the objective of the contest. I chuckled to myself and thought, "Here's a kid who's already learned everything he needs for a career at the Pentagon."

steve.ciarcia@circellar.com