

CIRCUIT CELLAR

I N K®

THE COMPUTER APPLICATIONS JOURNAL

August/September, 1992 — Issue #28

SIGNAL PROCESSING

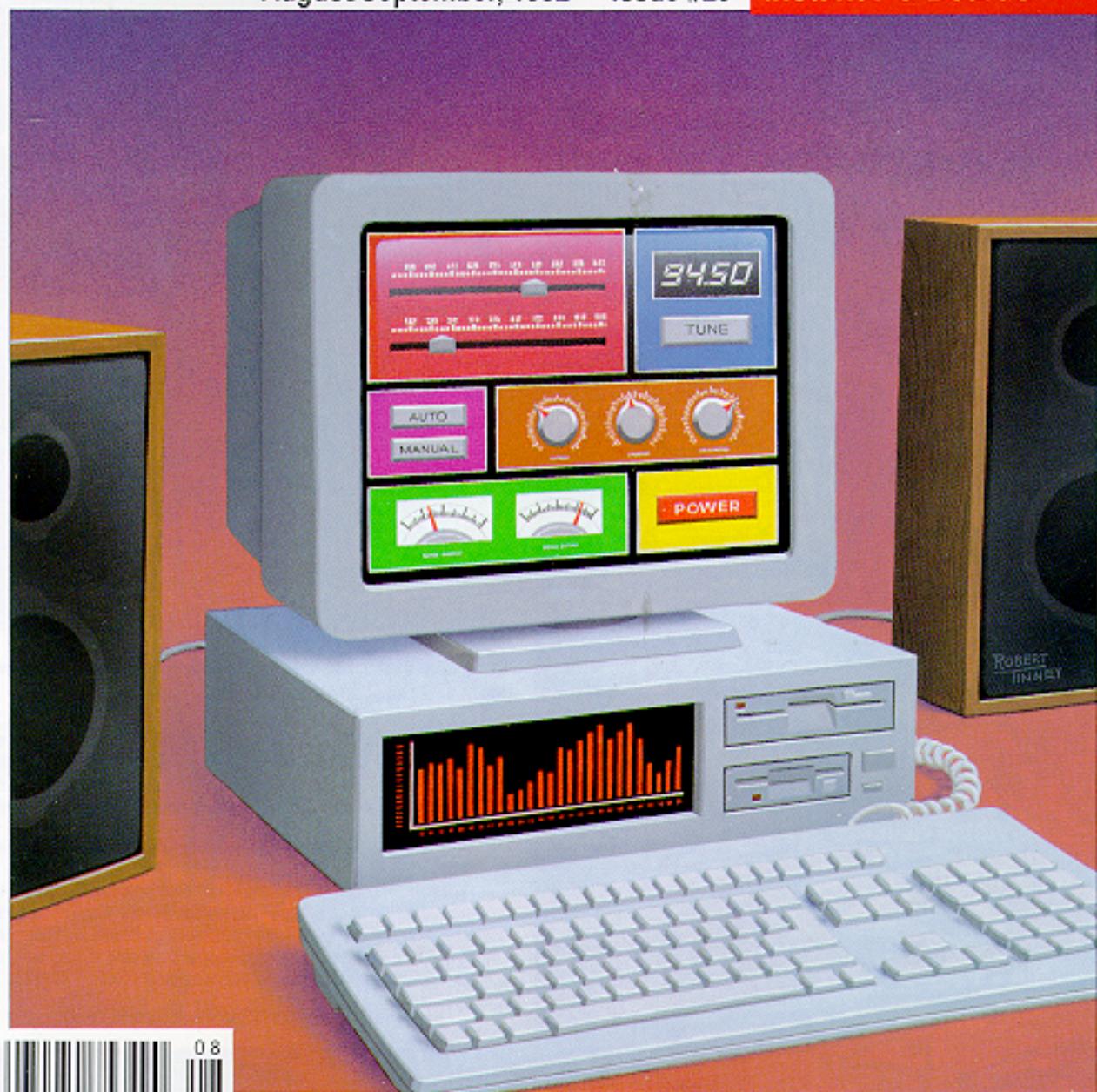
SPECIAL SECTION:

Embedded Interfacing

RISC vs. DSP

The Photonic Transistor

Instant PC Boards



\$3.95 U.S.
\$4.95 Canada

EDITOR'S INK



Process This

While the term "signal processing" can cover a full range of topics, we've decided to concentrate on *digital signal* processing in this issue's theme articles.

DSP continues to be a hot topic, with more and more consumer and industrial devices showing up on the market sporting features and capabilities previously found only on very expensive equipment or not found anywhere at all.

Does digital signal processing necessarily mean the use of a dedicated digital signal processor chip, though? In our first article, we take a look at instances where some of today's fast RISC processors actually look pretty good next to dedicated DSPs in traditional DSP applications.

Our next two articles move away from benchmarks and theory and into the practical uses of DSP chips. Audio spectrum analyzers have traditionally been built using rows of analog filters tuned for specific frequencies. Using DSP, though, our first project fits in the palm of your hand while analyzing the frequency content of a voice input and displaying the results on an oscilloscope screen. Our second project demonstrates the dynamic nature of filters implemented using DSP as the **DSP2000** Audio Waveform Shaper works to extract voice from a signal full of static.

I'm particularly excited about the last article in the feature section this issue. The single most important invention in the development of today's high-speed electronic computer was unquestionably the silicon transistor. However, electron-based computers are quickly approaching theoretical limits when it comes to speed and size. Light is certainly going to play a large role in future computers. We are thrilled to be the first publication to run an article detailing a new invention that could be as important to the computer industry as the silicon transistor: the photonic transistor. Using light and holograms, future computers many orders of magnitude faster than today's best desktop machines will be the norm, all based on the very simple photonic transistor. Be sure to check it out.

In our **special** section, we take a look at how to replace the popular stepper motor with a closed-loop DC motor control system that offers additional benefits over steppers. In the second article, we discuss a topic many readers have written to us about: designing with programmable logic devices. There are some pitfalls you have to avoid if you want a solid design.

Rounding out our issue are our regular columns. Ed wraps up his series describing the various HCS II modules by presenting some tricks he used in the **DIO- and ADIO-Link** modules. He also lets us in on a very useful debugging session that took place in the Circuit Cellar. Jeff shares the results of his experiments using the latest in instant PC board etching systems. Tom reviews a new standard designed to eliminate much of the cable clutter found behind a typical PC. Finally, John explores some options for reducing power consumption in battery-powered embedded systems without the need for complicated power monitoring hardware.

In our next issue, we'll have articles dealing with Measurement 8 Control and Embedded Graphics & Video, so watch for it in the mail.

CIRCUIT CELLAR **INK**

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR

Steve Ciarcia

MANAGING EDITOR

Ken Davidson

ASSOCIATE EDITOR

Lisa Nadile

ENGINEERING STAFF

Jeff Bachiochi & Ed Nisley

CONTRIBUTING EDITORS

Tom Cantrell & John Dybowski

NEW PRODUCTS EDITOR

Harv Weiner

ART DIRECTOR

Lisa Ferry

STAFF RESEARCHERS:

Northeast

John Dybowski

Midwest

Jon Elson & Tim McDonough

West Coast

Frank Kuechmann

PUBLISHER

Daniel Rodrigues

PUBLISHER'S ASSISTANT

Susan McGill

CIRCULATION COORDINATOR

Rose Mansella

CIRCULATION ASSISTANT

Barbara Maleski

CIRCULATION CONSULTANT

Gregory Spitzladen

BUSINESS MANAGER

Jeannette Walters

ADVERTISING COORDINATOR

Dan Gorsky

CIRCUIT CELLAR INK (ISSN 0896-8985) is published bimonthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 8752751. Second-class postage paid at Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions \$1795. Canada/Mexico \$21.95, dloUwcanries \$32.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders to Circuit Cellar INK, Subscriptions, P.O. Box 3050-C, Southeastern, PA 19398 or call (215) 630-1914.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 3050-C, Southeastern, PA 19398.

Cover Illustration by Robert Tinney

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST

Debra Andersen

(617) 769-8950

Fax: (617) 769-8982

MID-ATLANTIC

Barbara Best

(908) 741-7744

Fax: (908) 7416823

SOUTHEAST

Christa Collins

(305) 9668989

Fax: (305) 9858457

MIDWEST

Nanette Traetow

(708) 789-3080

Fax: (708) 789-3082

WEST COAST

Barbara Jones

& Shelley Rainey

(714) 540-3554

Fax: (714) 540-7103

Circuit Cellar BBS—24 Hrs. 300/1200/2400 bps, 8 bits, no parity, 1 stop bit, (203) 871-1988; 9600 bps Courier HST, (203) 871-0549

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

Circuit Cellar INK makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, Circuit Cellar INK disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in Circuit Cellar INK.

Entire contents copyright ©1992 by Circuit Cellar Incorporated. All rights reserved. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

INSIDE ISSUE 28

- 14 To DSP or Not to **DSP**/Will a RISC chip do it Better?
by M. R. Smith
- 26 Analyze Voice in the Palm of Your Hand
by Gerald **McGuire**
- 34 Shaping the World of Sound
by Steven Avritch
- 40 The Dawning of the Light Transistor/
An Optical Computer Method Using Interference Fringe Component Regions
by John N. **Hait**

SPECIAL SECTION Embedded Interfacing

- 50 Closing the Loop on DC Motor Control
by Tom Dahlin & Don Krantz

- 58 Designing with Programmable Logic
by Charles R. Conkling, Jr.

- 2 Editor's **INK**/Ken Davidson
Process This

- 4 Reader's **INK**-Letters to the Editor

- 8 New Product News
edited by Harv Weiner

- 70 Firmware Furnace/Ed Nisley
HCS II War Stories and **I/O** Links

- 80 From the Bench/Jeff Bachiochi
Approaching PCB Nirvana

- 86 Silicon **Update**/Tom Cantrell
The Ultimate Desk **ACCESSory**?

- 92 Practical Algorithms/John Dybowski
Power Code

- 98 **ConnecTime**—Excerpts from the Circuit Cellar BBS
conducted by Ken Davidson

- 112 Steve's Own **INK**/Steve Ciarcia
Cost is in the Eye of the Beholder

- 97 Advertiser's Index

READER'S INK

DS5000 Ideosyncrasy

Having just received issue #27, I was pleased to see another article ("The Elements of a Data Logger" by John Dybowski) that made use of the Dallas DS5000 microcontroller. I have been using this module for a while now, and feel that its advantages in board area, easy downloading, and nonvolatility outweigh its high cost. My only gripe with it is that the versions with more than 32K of memory do not come in the 40-pin DIP form factor.

However, John made a comment in his article that may mislead future users of the device regarding a problem that stumped me for a while. He says, "Also, note that all the pins of port 2 are available for use as general I/O because the RTC is referenced using **MOVX A, @R0**-type instructions that have no effect on the high-order address bus."

What he says is true, and it is also true that explicit references to the embedded RAM of the DS5000 (using **MOVX A, @R0**-type instructions) are performed on a separate internal bus that avoids disturbing P2. However, during the **MOVX A, @R0**-type instructions, certain values that may happen to be on P2 can cause implicit references to the embedded RAM, using a feature known as "page-mode addressing." When this happens, the external memory cycle will not occur, and the embedded RAM will be accessed instead, causing mysterious behavior in the best case, and embedded RAM corruption in the worst.

What this means is that P2 is not completely available for general I/O when you also want to use PO as a multiplexed bus. In my case, I simply avoided using P2.7 and left it set to a "1." This forced all page-mode accesses into the high half of the address space, which is never used for embedded RAM accesses in either the 8K or 32K versions of the DS5000.

The *Computer Applications Journal* is one magazine I really look forward to receiving—I read it cover to cover as soon as it arrives. Thank for doing such a great job.

Dave Tweed, Littleton, MA

Get the Noise Out

(The following exchange took place on the Circuit Cellar BBS:)

In your February/March 1992 issue (#25), Mark Nurczyk went a long way to preprocess the input of an

A/D converter, and although he goes through a number of valid points (like Spice modeling and some other production concerns) in his article, he seems to miss the whole point altogether.

Such preprocessing would be fine if the reason were noise or the like, but as the problem is DC offset and input scaling, there are easier ways to accomplish it. If the ADC is some standard part like the ADC0808 or ADC0809 or almost any other ADC, you can modify the reference inputs to suit your needs.

In Mark's example, by using 2.25 V and 3.9 V in the REF- and REF+ inputs (it can be done with a simple three-resistor divider in the relation 1.3:2:2.7), the thermometer would use 84% of the full range, so the precision goes to 1.2°F per bit.

Another useful alternative would be adding a small noise source (± 1 bit) to the reference inputs (this could be done with the same microcontroller and some passive filter components). This dither noise can be filtered out with a promediation routine in the processor, so you could accomplish any needed precision (with ADC linearity and, maybe, measurement time the only limiting factors).

Edgar Brown, Universidad Simon Bolivar, Caracas, Venezuela

Ed Nisley replies:

Welcome aboard!

I'd like to see a good presentation of using the dithering trick on analog inputs. I've read a little about it, but could use a practical tutorial. Are you up to a brain dump on the subject (if it's too expensive from Caracas, don't do it just for me!).

Edgar Brown replies:

Well, I don't know if I could do a tutorial on this, but here is my best attempt [I've only had 4 hours of sleep in the last 2 days!]:

Dithering is based on promediation of added noise to a signal. Let's say you have a signal that you have to measure. There is too much noise added to it and you know the noise has no DC (i.e., its mean is zero). You could get a good guess by sampling the signal several times and the getting the mean of the samples. $E(S + \text{noise}) = E(S) + E(\text{noise})$, where $E()$ is expected value or mean. It works real well in the analog domain.

In the digital world, you could use the same principle by considering the quantization of a signal as noise added to it. However, the problem is the noise is corre-

lated to the signal. If there is no noise in the original signal, finding the mean of a lot of samples would do no good.

However, if you add some noise to the signal (with zero mean), and this noise is of little amplitude (it makes no sense to hide a good signal with the noise we are adding), you can decorrelate the quantization noise from the signal and make it look like real added noise. Now by finding the mean of the samples, you can filter out the noise.

For this to work, the noise must be at least the size of the quantization noise (i.e., 1/2 an LSB of the quantizer). However, if you wish to account for problems like nonlinearities in the ADC, you can make the noise a little bigger so it spans several bits.

Hope this helps.

Ed Nisley replies:

OK, so there are three principles: the noise must be added in the analog front end, it's got to be relatively small, and you have to know what the characteristics are so you can filter it out digitally once it's been quantized.

I'll add that to my list of Things to Read More About...thanks for the update!

And We Have a Winner

Congratulations to **Curtis Goodson** of Austin Texas. Curtis is the winner of our latest Tech Deck contest sponsored by Contact East and the **Computer Applications Journal**. He will be receiving a Fluke 87 Digital Multimeter.

Be sure to watch your mail for future Tech Decks and more contests!

We Want to Hear from You

Our readers are encouraged to write letters of praise, condemnation, or suggestion to the editors of The Computer Applications Journal. Send them to:

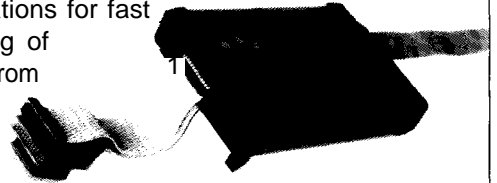
The Computer **Applications** Journal
letters to the Editor

4 Park Street
Vernon, CT 06066

OVER 500
SOLD

320C25 DSP ASAP \$1,995

Drive your embedded 320C25 design to market as fast as possible with our 320C25 in-circuit emulator. The MACROCHIP 'C25 emulator features real time emulation to 50 MHz with no wait states, 64K words of program overlay memory, simple software breakpoint, single step trace, disassembler, and RS-232 communications for fast downloading of programs from your PC COM port.



Call, write or fax for literature:



1301 N. Denton Drive
Carrollton, TX 75006
Tel 1-800-783-9546
Fax 214-245-1005

102

NEW!

INSTANT 320C25 MONITOR \$295

Ever wondered what it would be like if you could communicate with your 'C25 application using your PC COM port? Well, we have. Now we are offering you a C25 chip already programmed with a 4K-word monitor in internal program memory. All you have to do is insert the chip into your target socket and hook up the BIO and XF pins to a MAX 232. With the instant 'C25 monitor, you will be able to download/upload TI-tagged or Intel hex files directly into your target application at speeds to 9600 baud, examine/edit internal/external memory and I/O. We'll give you entry points so you'll be able to call the GET-CHAR, GET_LINE, COUT, PARSE, ASC_TO_HEX, etc. routines already in ROM.



For only \$295 more, we'll even sell you a symbolic debugger you can upload to your target application.

Call, write or fax for literature:



1301 N. Denton Drive
Carrollton, TX 75006
Tel 1-800-783-9546
Fax 214-245-1005

#103

NEW PRODUCT NEWS

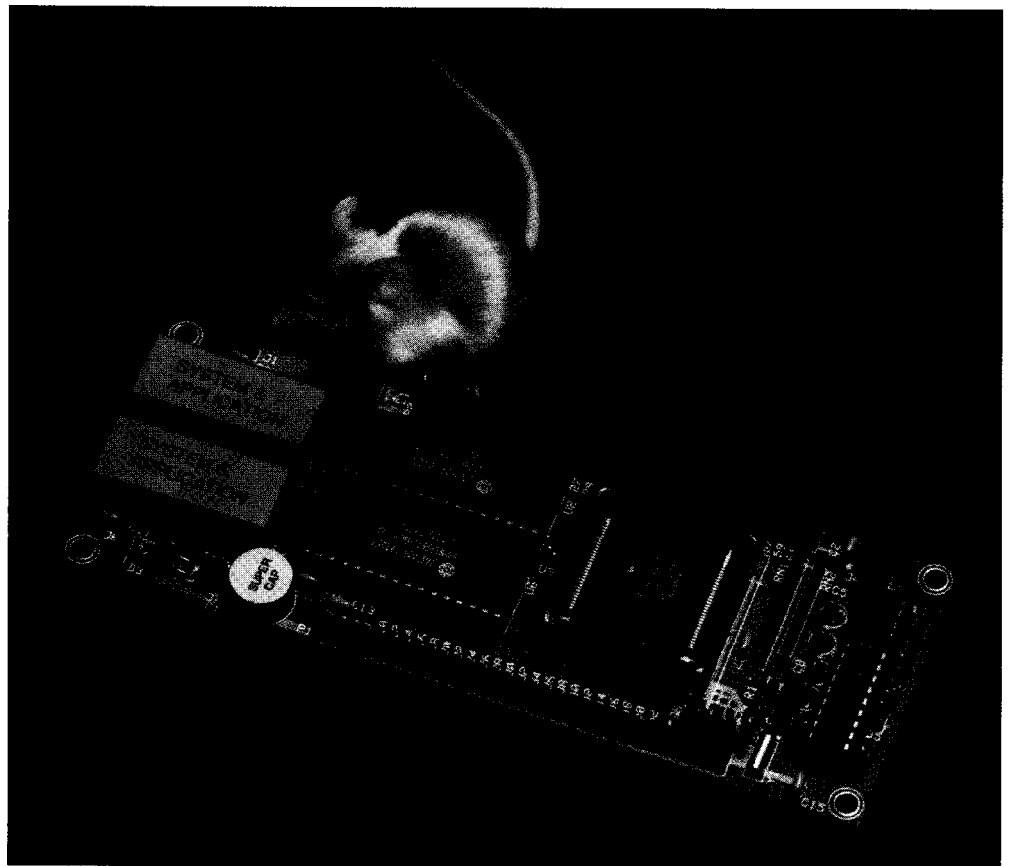
Edited by Harv Weiner

MC68332 SINGLE-BOARD COMPUTER

A low-cost, high-speed, low-power single-board computer based on the Motorola MC68332 32-bit microcontroller has been announced by Vesta Technology. The Vesta SBC332 has features that make it ideal for embedded systems control and development.

Based on the powerful MC68020, the MC68332 has a 32-bit internal data path and 32-bit internal address bus. Externally, it has a 16-bit data bus and a 24-bit address bus. This feature, coupled with a 16.78-MHz clock, makes the SBC332 a potent embedded system "engine." The 68332 microcontroller features on-chip RAM, a Queued Serial Module (QSMI), and an intelligent 16-bit Timed Processor Unit (TPU). These integrated peripherals allow a variety of complex operations with minimal CPU intervention. Low-power operation is fully supported with minimal power consumption during normal operation (110 mA), the capability to reduce clock speed as the application permits, and a standby mode.

The Vesta SBC332 is Motorola Business Card Computer (BCC) compatible so allows the direct use of software systems designed around Motorola's popular



evaluation platform. The Vesta SBC332 features socketed 28- and 32-pin DIP external memory, which includes two 8-bit EPROMs with a capacity up to 1 MB and two 8-bit SRAMs with a capacity up to 1 MB. An external watchdog and supercap-backed standby SRAM are also provided. The SBC332 measures 2.3" x 6.25" and maintains compatibility with the BCC P1 and P2 connectors.

Two peripheral cards, designed to facilitate prototyping, are also available. The MFP332 Multifunction Peripheral adds A/D and D/A conversion to the SBC332. The ADC subsystem will acquire 12 bits of analog

data from any of eight channels at over 80 kHz. Four independent 12-bit DACs provide a 0-IO-volt output range. Both ADC and DAC are connected to the 16-bit-wide bus. Two additional RS-232 serial channels, one of which may be configured for RS-485 operation, are also provided.

The LBK332 LCD, keypad, and beeper peripheral is the operator interface to the SBC332 in embedded applications. It supports alphanumeric displays from 1 x 8 to 4 x 40, providing negative Vss when required. A self-scanning keypad circuit supports up to 8 x 8 keys and will generate an interrupt upon keypress. A multitone audio annuncia-

tor is intended to assist during data entry via keypad.

The Vesta Technology SBC332 single-board computer sells for \$249 and the MFP332 Multifunction Peripheral sells for \$229, both in quantities of 25. The LBK332 is priced at \$79 in single quantities.

Vesta Technology, Inc.
7100 W. 44th Ave., Ste. #101
Wheat Ridge, CO 80033
(303) 422-8088
Fax: (303) 422-9800

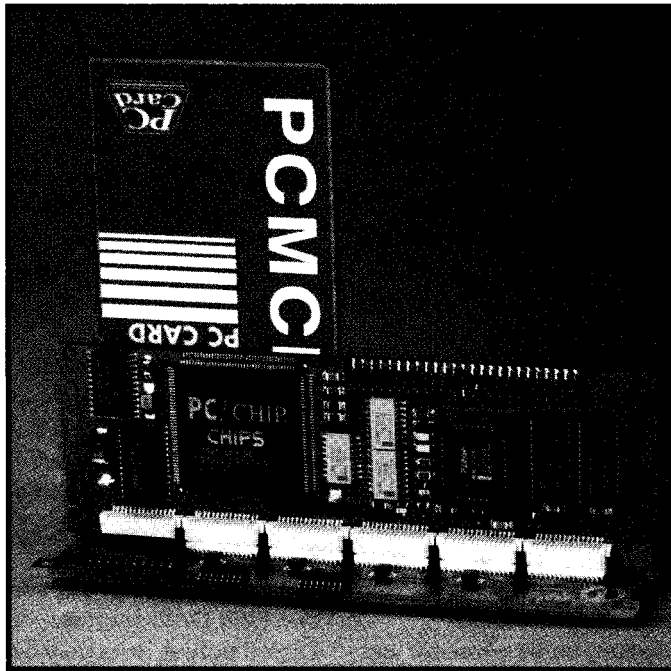
#500

NEW PRODUCT NEWS

MINIATURE XT-BASED COMPUTER

Dover Electronics has announced an innovative miniature computer, the E.S.P. 8680. Based on the Chips and Technologies 8680 single-chip computer, the module measures 1.7" x 5.2" and provides full computing functionality. It has a fully integrated 8086XT with a nonvolatile memory card socket (PCMCIA), CGA graphics, up to 1 MB of RAM, a keyboard interface, one serial port, and expansion connectors.

The E.S.P. 8680 was designed to permit expansion and upgrades. The double-sided surface-mount technology produces a very small, dense module that fits into a OS-inch-on-center backplane in 3-D fashion. In addition to core



processing, add-on E.S.P. modules provide functions such as PCMCIA, modem, fax, packet radio, network interface, SCSI, ADCs and DACs, static memory, and bar-code scanning, all in the same form factor.

Other features internal to the C&T 8680 chip include an intelligent sleep mode for reduced power consumption, a SuperState R mode for a separate operating environment and the enabling of full I/O and

interrupt monitoring without BIOS modification, and a four-stage pipeline and 14-MHz speed for performance comparable to a '286-based system. A 26-bit address bus enables a 64 MB-memory map and directly supports a PCMCIA memory card. A virtual interrupt feature allows interrupts to be monitored, redirected, or both, before any operating system, application program, or TSR sees them.

A development kit with the E.S.P. 8680 is available for \$995.

Dover Electronics
1198 Boston Ave.
Longmont, CO 80502
(303) 772-5933
Fax: (303) 776-1883

#501

64-MHZ ADC BOARD FEATURES 16-MB BUFFER AND DSP LINK

Sonix Inc. has announced the STR*864, a 64-MHz Transient Digitizer board featuring simultaneous sampling at 64 MHz on two channels, a high-speed DSP link, and a flexible memory buffer scheme providing up to 8 MB of memory per channel or 16 MB on one channel. It provides freedom from the static architecture of a stand-alone digital oscilloscope and from the slow data transfer rates between a stand-alone instrument and the computer.

The STR*864's high-speed data acquisition memory is mapped directly into PC memory space. Once the waveform has been captured, the PC can transfer data off the board at 1-MHz (8 bits) or 3-MHz (16 bits).

The STR*864 will not slow down PC operations due to its 25-ns RAM and fast bus interface logic. As the processing speed and power of the PC's microprocessor increases, the STR*864 will continue to provide bus and memory access speed compatibility. A special high-speed data transfer link has been included because the STR*864 is expected to be used with DSP boards.

All board functions are under software control including input impedance selection, AC/DC coupling, input voltage range, sampling rates, ADC output coding, trigger selection, clock control, threshold phase and level, board selection, and interrupt enabling.

The STR*864 is well suited for two-channel, phase coherent acquisitions as in radar applications. Other applications include digitizing waveforms, processing real-time signals, spectrum analysis, capturing transient data, implementing ultrasonic inspection systems, and analyzing optical and laser signals. More than one board may be installed for multichannel applications. Pricing was not available at press time.

Sonix, Inc. . 8700 Morrisette Dr. . Springfield, VA 22152 . (703) 440-0222 . Fax: (703) 440-9512

#502

NEW PRODUCT NEWS

HIGH-LEVEL/LOW-LEVEL DEBUGGER FOR 8051

A high-level debugger for 8051 C compilers that is **keypress** compatible with Borland's Turbo Debugger has been announced by **ChipTools**. **ChipView-51** is available in two versions: a high-performance simulator/debugger or a front end for the Nohau **EMUL5** I-PC emulator.

With **ChipView-1**, the Turbo C programmer can instantly begin debugging code in the embedded-systems environment. **ChipView-1** presents over 14 different views of your program, including all of Turbo Debugger's views. Screen layout is fully user configurable with movable, resizable, and **zoomable** windows for **25-, 43-, and 50-line** EGA/VGA modes. Features include data browsing of C structures and linked lists using point and click, and over **100K** of context-sensitive hypertext help. **ChipView-** can display a C-level call stack, which displays nested function calls along with their arguments—a powerful view absent from other 8051 debuggers.

The emulator/debugger version for Nohau's **EMUL51-PC** is tailored to fully support its emulator and trace boards. Emulation and trace collection can continue to run while you analyze a trace history.

The high-speed simulator version running on a 33-MHz '486 PC achieves the real-time speed of a **12-MHz** 8051. This level of performance makes it ideal for testing entire programs in a safe environment, even before the application hardware is designed. In addition, chip I/O can be simulated for both polled and interrupt-driven systems. Like the emulator version, the simulator can not only collect a trace history for execution analysis, but also reverse-execute, or "undo," back to a previous state whenever a bug is detected, backing into and out of interrupt code if required.

ChipView-1 is the only 8051 debugger to support all popular 8051 C compilers at a high level, and features total compatibility with all object formats. Unlike debuggers that support only simple data types, **ChipView-1** lets you browse through C structures, unions, enumerations, bit-fields, pointers, and arrays.

The simulator version of **ChipView-** sells for \$795, the emulator version for \$595, and a combo package for \$995. System requirements include an IBM XT or compatible with 640K RAM and a hard disk. Additional system RAM as EMS or XMS is recommended for programs with sizable debugging information.

ChipTools

1232 Stavebank Rd. • Mississauga, Ontario • Canada **L5G2V2**
(416) **274-6244** • Fax: (416) 891-2715

#503

PC SERIAL PORT TO DATA ACQUISITION CONTROL MODULE

B&B Electronics is offering a general-purpose control module that is connected to your computer's RS-232 serial port. The **Model SPIO** can be configured and manipulated using a PC or compatible and an easy-to-use command set. The SPIO comes with an instruction manual and a diskette that includes configuration, **demonstration**, and upgrade programs.

The SPIO has eight programmable I/O lines, eight 8-bit ADC inputs, and two 8-bit DAC outputs. Digital output lines may be used as normal outputs or delayed on/off. One digital output is available as a programmable pulse-width output. The digital input lines may be used as normal inputs or latches, and two digital inputs are available as event counters. I/O lines are available through a female DB-25 connector.

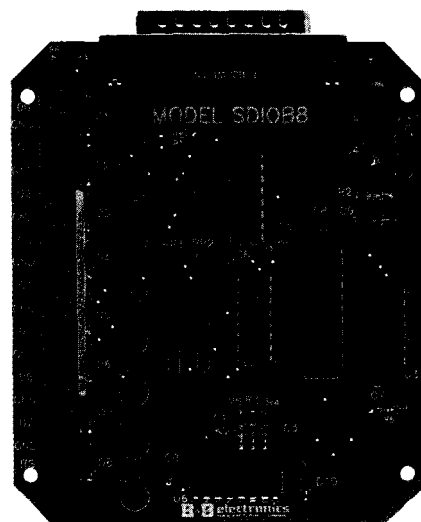
Two compatible interfaces are available for

the SPIO. The **SDIOB8** connects to the SPIO and offers a watchdog circuit for its eight **programmable** I/O lines. Each I/O line has an LED status indicator and can be used to control voltages as high as 50 VDC. The **DAPBI** connects to the SPIO to provide terminal block access to available I/O lines. The **DAPBI** has a prototype area available for custom signal conditioning circuitry.

The SPIO sells for \$119.95. The **SDIOB8** interface module sells for \$69.95 and the **DAPBI** for \$49.95. A separate power supply (**232PS3**) is available for \$14.95.

B&B Electronics
P.O. Box #1040
Ottawa, IL 61350
(815) **434-0846**
Fax: (815) 434-7094

#504



NEW PRODUCT NEWS

HIGH-PERFORMANCE SINGLE-BOARD COMPUTER

Innovative Integration is now shipping the SBC25 high-performance single-board computer.

The SBC25 couples the Texas Instruments TMS320C25 microcontroller and C or FORTH programming language to provide a complete, integrated solution to the

development and implementation of real-time applications.

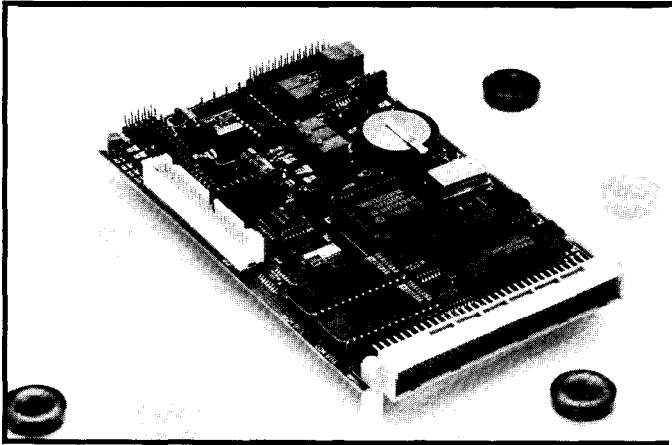
The TMS320C25 architecture is optimized for computationally intensive DSP algorithms and operates at a sustained 12-MIPS throughput. The processor provides one 16-bit counter/timer, three prioritized interrupts, on-chip RAM/ROM, and a 16-bit single-cycle multiplier/accumulator. Additionally, the 160-mm x 100-mm SBC25 features an on-board real-time clock, an eight-channel prioritized interrupt controller, a 2-Mbps DUART,

three 8-bit digital I/O ports, three 16-bit counter/timers, a watchdog timer, a multiplexed 12-bit 330-kHz ADC with a programmable gain amp, and four 12-bit 100-kHz DACs. Up to 16 MB of battery-backed RAM/ROM is supported.

The SBC is priced from \$499 in single quantities. Complete development packages start at \$2995.

Innovative Integration
4086 Little Hollow Pl.
Moorpark, CA 93021
(805) 529-7570

#505



ROMboy

Low Cost, Reliable, High Quality
ROM Emulation from the
Creators of PROMICE

- Emulation of up to 1 Mbit ROMs (4 Mbit-soon)
- Supports any ROM type (DIP or PLCC)
- Supports high speed ROMs
- Supports any word size up to 2048 bits wide
- Host software for DOS, Unix, Mac, VMS
- Battery backed memory
- Intelligent micro-controller based unit
- SMT CMOS layout for robustness
- Tiny size: 2.5" x 3.8" x 0.9"
- Unbelievably low price!

ROMboy...

from the authority in firmware development tools

Orders: (614) 899-7878
West: (415) 750-0219
FAX: (614) 899-7888



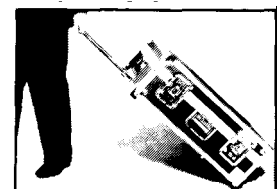
**Grammar
Engine
Inc.**

Take It Easy.

Take it easy on your cargo with a custom Cabbage Case built to the exact dimensions of your equipment.

Take it easy on your back with our extension handle and tilt wheels options.

Take it easy on your wallet. Let Cabbage Cases show you how easy it is to save money on quality, custom-built road cases that make shipping and traveling with your valuable cargo safer and easier. Prices quoted over the phone. Call 800-888-2495 today.



**CABBAGE
CASES, Inc.**

1166-C Steelwood Rd.
Columbus, OH 43212
800/888-2495
614/486-2495
FAX/486-2788

NEW PRODUCT NEWS

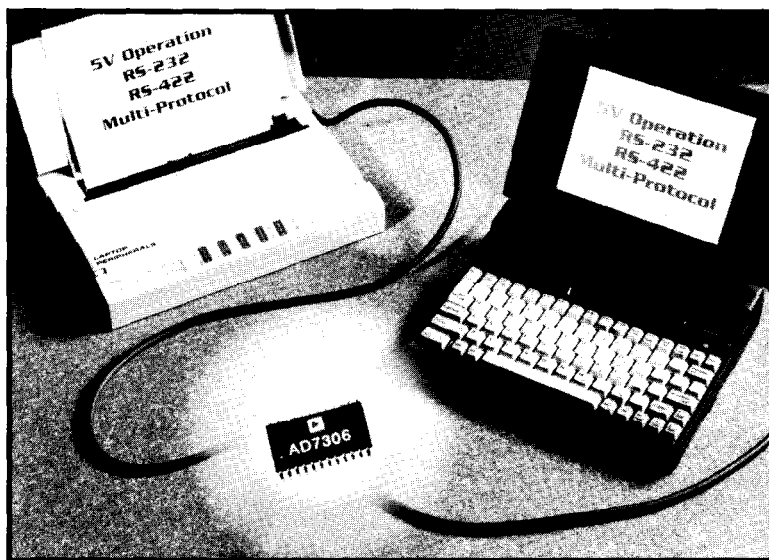
SINGLE-SUPPLY MULTIPROTOCOL TRANSCEIVER

Systems that must operate from a single 5-volt supply and communicate with other equipment using industry-standard RS-232 and RS-422 protocols can benefit from a chip announced by Analog Devices. The AD7306 is a multiprotocol driver/receiver chip that offers two RS-232 drivers, one RS-422 driver, one RS-232 receiver, and a configurable RS-232/RS-422 receiver in a 24-pin SOIC package.

The single-supply AD7306 generates ± 10 volts internally using a charge pump voltage converter. The charge pump allows RS-232 output levels to be developed without the addition of complex

external bipolar power supplies. Unlike designs that require expensive, bulky capacitors of up to 10 μF for operation, the AD7306 charge pump is efficient enough to operate using nonpolarized, miniature 0.1- μF capacitors, considerably saving circuit board space. Additionally, some of the charge pump output is available to power external circuitry requiring dual supplies.

The AD7306 transceiver provides an interface between TTL/CMOS signal levels and dual standard EIA RS-232/RS-422 signal levels.



The RS-232 channels communicate at rates up to 100 kHz, and the RS-422 channels are suitable for high-speed communications up to 5 MHz. Timing skew (T_{sk}) for RS-422 communication is typically only 2 ns. No-load power consumption is typically 50 mW with 100 mW maximum.

The AD7306 sells for \$3.75 in quantities of 1000.

Analog Devices
181 Ballardvale St.
Wilmington, MA 01887
(508) 658-9400

#506

DIGITAL SIGNAL PROCESSING BOARD

A digital signal processing board with analog and digital I/O has been announced by Dalanco Spry. The Model 500 is designed for the PC/AT- and ISA (Industry Standard Architecture)-compatible microcomputers. Applications include data acquisition, instrumentation and control, speech and audio, as well as general-purpose DSP software development.

The Model 500 is based around the Texas Instruments TMS320C5120-MIPS DSP. The unit provides data acquisition for eight channels at 12-bit resolution and a maximum 225-kHz sampling rate. Two 12-bit analog output channels are also provided, along with a buffered digital I/O connector for user expansion and the two serial interfaces of the TI DSP.

The board is populated with 64K words of program RAM and 128K words of dual-ported data RAM. The dual-ported architecture enables the creation of applications requiring simultaneous mathematical calculations coupled with concurrent analog and disk I/O.

The Model 500 features high throughput to the host PC's memory (up to 3 MB per second) and disk. Multiple boards may be used within a single system. Depending on the host system, data may be written to or read from disk at the maximum sampling rate of 225 kHz.

Software included with the Model 500 consists of a TMS320C51 assembler and debugger, FFT software, real-time signal and spectrum display, concurrent record and playback to or from disk, digital filter examples and FIR filter code generator, and a waveform editor.

The Model 500 with TMS320C51 sells for \$1600.

Dalanco Spry
89 Westland Ave.
Rochester, NY 14618
(716) 473-3610
Fax: (716) 473-3610

#507

FEAT'URES

14

To DSP or Not to DSP

26

Analyze Voice in the Palm of Your Hand

34

Shaping the World of Sound

40

The Dawning of the Light Transistor

To DSP or Not to DSP

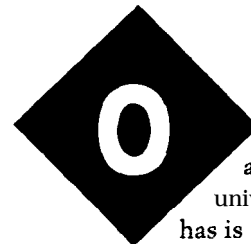
Will a RISC chip do it better?

Everyone usually assumes that to do true digital signal processing, a dedicated DSP chip is necessary.

Depending on the application, today's high-speed RISC processors may do just as good a job.

FEATURE ARTICLE

M. R. Smith



One of the few advantages a university professor has is combining research and teaching. I am able to expand my own knowledge in my chosen field as well as impart my experience to others with similar interests. The work I share with you here is a rewarding combination of results from both these vocations.

As a research project, my students and I designed a number of high-speed coprocessors for use in modeling to improve the quality of magnetic resonance images and for other digital signal processing (DSP) algorithms. The fastest system we developed was based around an Advanced Micro Devices microprogrammable DSP byte-slice chip family, which is now obsolete. Their disappearance is no tragedy, because although the project was successful, the custom micro-programmed system's high pin count made it unreliable. Recently, we focused toward implementing our algorithms on the new single-chip DSPs coming on the market.

Teaching classes in computer architecture that involve the comparative analysis of high-speed processors has allowed me, with my students, to compare the architectural features of the Motorola DSP56001, DSP56200, and DSP96002, NEC μ PD77230, and Texas Instruments TMS320 family DSP products. More recently, we have examined the Advanced Micro Devices Am29050, Intel i860, Motorola MC88100, and the SPARC family of RISC chips.

We became curious about just how well the non-DSP, but high-speed,

RISC chips would stack up against the more dedicated DSP chips and where the design limitations existed. The RISC instructions expose much of the processor's internal architecture, which allows tailoring of the timing of an algorithm's operation to optimize the use of the highly pipelined RISC resources. The dedicated DSP chips are more like complex instruction (CISC) processors where timing optimization is not as available. However, the DSP's specialized hardware makes each instruction very efficient.

The RISC chips are good general-purpose processors with many practical applications. Two high-speed RISC chips proved very amenable for DSP applications, but for very different architectural reasons. The Am29050 has a large register window and additional high-speed registers, all with direct access to a fast arithmetic processor unit (APU). By contrast, the i860 gets its DSP capability through the use of a wide (64-bit) instruction cache and an extremely wide (128-bit) data cache.

In this application tutorial, I discuss the implementation of a Finite Impulse Response (FIR) digital filter using these processors. Although aimed specifically at the Am29050 and i860 processors, many of the techniques I describe can be used to your advantage in other RISC applications. I will show you how to overcome the problems experienced because of memory access difficulties and the typically long RISC floating-point unit

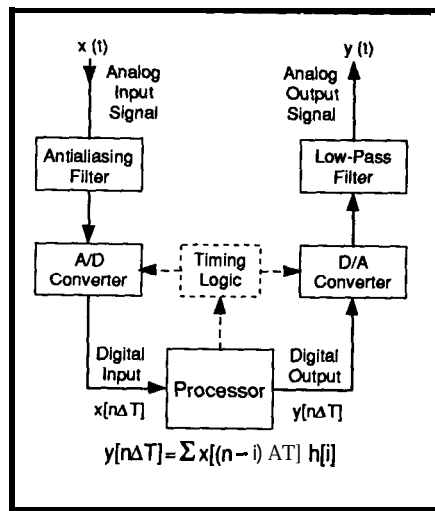


Figure 1-A FIR digital filter can be constructed with a processor and memory-mapped ADC and DAC devices. Conventional filters are used to band limit the input and output.

(FPU) pipeline. I'll discuss the limitations of the Am29050 and i860 as generalized DSP chips, and I'll suggest possible architectural modifications.

FIR FILTER THEORY

Figure 1 shows the typical schematic for a dedicated real-time FIR digital filter system. The ADC, used to sample the input signal $x(t)$, and the DAC, used to produce the filtered output signal $y(t)$, are assumed to be memory mapped into the data memory space of the processor. This location means that simple reads and writes can be used to access their values.

The pseudocode for the FIR filter program is shown in Listing 1 using a C-like syntax. The analog input signal is first filtered to reduce the signal and

noise bandwidth below the Nyquist rate to avoid signal contamination (aliasing). The digitizing occurs at intervals of ΔT seconds under the control of the timing logic. The new sampled input value, $x[n\Delta T]$, and the old digitized values, $x[(n-1)\Delta T]$, $x[(n-2)\Delta T]$, ..., are stored in a buffer. The digitized filter output of a linear phase FIR filter with **LENGTH** taps, $y[n\Delta T]$, is determined from the simple summation equation

$$y[n\Delta T] = \sum_{i=0}^{\text{LENGTH}-1} x[(n-i)\Delta T] \times h[i]$$

where the digital filter coefficients, $h[i]$, are designed to meet the required filtering response. For example, the output of a simple linear phase S-tap FIR filter is calculated from

$$y[n\Delta T] = \{x[n\Delta T] + x[(n-4)\Delta T]\}h[0] + \{x[(n-1)\Delta T] + x[(n-3)\Delta T]\}h[1] + x[(n-2)\Delta T]h[2]$$

The final analog signal is obtained by low-pass filtering the DAC output to remove the high-frequency components introduced during conversion. Before starting the next filtering stage, the circular buffer used to store the earlier input values must be updated. For the 5-tap filter above, the output is the same as

$$x[(n-4+i)\Delta T] = x[(n-3+i)\Delta T]; \quad 0 \leq i < 4$$

For long-length filters, this update would require considerable, time-consuming data movement or pointer manipulation.

IMPLEMENTATION OF A FIR FILTER-ATTEMPT I

Implementing a FIR filter on a processor does have its difficulties. Filter coefficients must be expressed with sufficient accuracy to model the required filter response. You must also ensure there are sufficient bits in the APU to prevent either underflow or overflow during the summation, $y[n\Delta T]$. Although the number of bits required for this accuracy depends on the actual filter, a reasonable number of guard bits for the sum would be about 12 at either end of the digitized

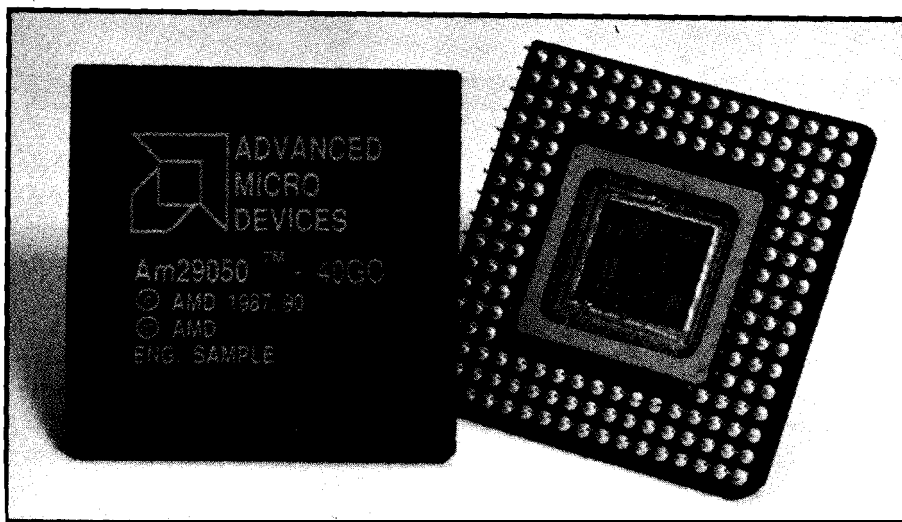


Photo 1-The Am29050 RISC chip has a lot in its favor in certain DSP-oriented applications.

input signal. For a 12- to 16-bit input signal, an APU with a width of 36 or more bits is required. For example, the Motorola DSP56001 chip maintains a 56-bit accumulator accuracy-much higher than the input signal's 24-bit width.

The FIR filter pseudocode (refer to Listing 1) indicates that considerable overhead will occur at each step of the filter to make sure the program does not write outside the circular buffer required to store the previous input values, $x[n \text{ AT}]$. To cut this overhead, maintain the buffer using the specialized hardware you find in dedicated DSP chips (e.g., TMS320 and DSP56001).

Many DSP algorithms involve extensive loops. In a processor with a long instruction pipeline, these loops can cause time loss by stalling after a branch as new instructions are fetched. Some DSP chips get around this problem by allowing the repeat of a given instruction many times in a single instruction loop. Finally, sufficient additional registers for pointers and temporary variables are needed.

RISC chips often have a register window with a format very similar to a circular buffer. However, unless this window has easy access (no penalty) to a hardware multiplier, the implementation of the FIR equation will be greatly slowed. The scalar Am29050 has a large [128] register window with direct access to the high-speed FPU. The SPARC chip has a register window, but it does not have this direct access. The superscalar i860 does not have the register window; instead, it uses a dual-instruction capability to refill its smaller register bank in parallel with floating-point operations.

The filter coefficients also need to be stored where they can be accessed without penalty. In addition to the register window on the Am29050, there is a large additional register bank also with direct access to the FPU. Again, the dual instruction of the i860 can be used to create a similar capability. Some other RISC chips are not as fortunate, so their speed is considerably slower.

Listing 1-A direct implementation of a FIR digital filter can be described in C-like pseudocode.

```
/*Requires software implementation of a circular buffer */

int ReadA2D(void);
void SetUpCoeffs(float *,int);
void WriteD2A(int);

#define LENGTH 95
#define NUMCOEFFS 48

main0 { /* data and filter coeffs buffers */
    float x[LENGTH], coeffs[NUMCOEFFS];
    /* control of the buffers */
    float *buffstart, *buffend, *nextempty;
    float *xrecent, *xancient, *coeffcurrent;
    float sum /* filter sum */
    int xn, yn; /* ADC and DAC integer values */
    int count;

    SetUpCoeffs(coeffs, NUMCOEFFS); /* start up */
    buffend = nextempty = x + LENGTH 1;
    buffstart = x;

    for (;;) { /* main loop */
        xn = ReadA2D(); /* get new value */
        *nextempty = (float) xn; /* convert and store */

        xrecent = nextempty; /* adjust pointers to buffer */
        coeffcurrent = coeffs;
        xancient = xrecent + 1; /* since circular */
        if (xancient > buffend) xancient = buffstart; /* wrapped? */

        /* perform convolution, check buffer */
        sum = 0.0;
        for (count = 0; count < NUMCOEFFS - 1; count++) {
            sum += *xrecent * *coeffcurrent;
            if (--xrecent < buffstart) xrecent = buffend;

            sum += *xancient * *coeffcurrent;
            if (++xancient > buffend) xancient = buffstart;
            coeffcurrent++;
        }
        sum += *xancient * *coeffcurrent; /* center tap */

        yn = (int) sum /* convert and write out */
        WriteD2A(yn);
        /* update start of future circular buffer */
        if (++nextempty > buffend) nextempty = buffstart;
    }
}
```

The i860 and Am29050 meet the accuracy requirement because they can maintain a 64-bit internal representation in the FPU accumulators without any time penalties.

The branching in a heavily pipelined RISC has to be handled carefully. You can avoid stalls in the instruction pipeline by having an on-board cache to hold the most recently used branch instructions. Stalls in the ALU pipeline have to be handled in a different way, as I demonstrate later.

A number of RISC chips (including the Am29050 and the i860) have the ability to perform a simultaneous floating-point multiply-and-accumulate operation (**FMAC**) without the necessity of fetching and initiating individual multiply and add instructions—a great timesaver. Most of the newer RISC chips have a floating-point capability that can simplify the design of DSP algorithms.

With these resources, I will show that implementing a 95-tap linear

phase FIR digital filter with a $AT \approx 3.0$ μs sample time using the Am29050 and i860 RISC processors is possible.

The architectural features of the RISC chips play an important role in DSP applications. Of the two chips, the Am29050 has the simpler (more intuitively obvious) assembler instruction set, so the FIR filter will be discussed in terms of this chip.

Listing 2 shows the start-up Am29050 assembler code for the filter. Various mnemonics are established for the Am29050's 64 general-purpose registers (**g r**), used for the filter coefficients (H_n), and the 128 local window registers (**l r**), used as a circular buffer for the data (X_n). Constants (such as addresses) needed inside the loop are placed in additional general registers for faster operation.

The first attempt at an actual 95-tap FIR filter code is shown in Listing 3 and is an almost direct implementation of the pseudocode from Listing 1. The integer input value from the ADC is converted to a floating-point number before being stored in local register X 0. This step avoids the overhead of continually reconverting the previous digitized and stored values into floats during the time-sensitive filter loop. The **FMAC** instruction is used to maximize the speed of the FIR equation. After the filter loop, the sum is converted back to an integer before being sent to the DAC. Finally, the circular buffer is updated by adjusting the register window using the register stack pointer (**g r 1**). In a single cycle, this instruction obtains what would otherwise require 94 register moves.

Generally, this code implies that an N-tap linear phase FIR filter can be implemented in $N + 8$ instructions. At first glance, this assumption would indicate that a 95-tap filter with 192 floating-point operations can be performed in 103 cycles (2.56 μs at 40 MHz). However, this result is not the case because it takes $4N + 18$ cycles (9.95 μs) when run. The problem is the code does not take into account the RISC FPU's highly pipelined architecture used for the **FMAC** instruction.

The other RISC chips fair even worse, with the i860 and MC88 100 requiring a time in the order of $6N-9N$

REALIZE THE POWER OF PARADIGM

...ced staff
...able to answer ques-
... and provide application
... assistance. Reliable product, qual-
... ity people realize the power of
... the when for your next embedded
... system design.

Paradigm Productivity

EMUL6: Harness Turbo Debugger's power for debugging embedded applications using popular Intel 80x86 and NEC V-Series microprocessors.

PARADIGM IDE: The only low-level software to fully support all Borland and Microsoft compilers. HIRENA: Design your embedded system state from the comfort of your PC.

Call for information!

Toll-Free 1-800-537-5043



PARADIGM

The Model for Programming Productivity

3301 Country Club Road, Suite 2214 • Endwell, NY 13760 • (607) 748-5966 •
FAX: (607) 748-5968

See us at the Embedded Systems Conference—Booth#305

The Computer Applications Journal Issue #28 August/September, 1992

17

cycles ($\approx 15\text{--}23\mu\text{s}$). Not only is there the deep FPU pipeline, but these chips only have 32 registers attached to the FPU. These registers must be continually reloaded with data and coefficient values from the memory. In addition, the circular buffer operation must be handled in software, whereas on the Am29050 it was handled at low cost via the register window. Some current SPARC chips also have a problem because these data fetches must compete with instruction fetches on a single input bus. (Not all SPARC chips have exactly the same architecture; e.g., instruction caches.)

Details of the problems from the long FPU pipeline found in RISC chips are explained in terms of the Am29050 pipeline. Table 1 shows the effect of the pipeline operation for the consecutive **FMAC** instructions needed to implement the start-up of the FIR operation

```
X0 = (float) Xn      ; CONVERT
sum = 0              ; FMAC
sum = X0xH0+sum      ; FMAC
sum = X1xH1+sum      ; FMAC
sum = X2xH2+sum      ; FMAC
```

where X_0 , X_1 , and X_2 are the respective current, last, and next-to-last inputs, and H_0 , H_1 , and H_2 , the corresponding filter coefficients. The pipeline stalls after the first **FMAC** instruction as the processor waits for the conversion of X_n to complete. A second stall immediately occurs because the deep pipelined **FMAC** instruction means that **sum** is not available for use by the second **FMAC** instruction until six cycles later.

The most obvious result of these pipeline problems is the occurrence of three **STALLs** as each **FMAC** instruction waits for the last to complete. Also, a sequence of eleven **STA L Ls** occurs after the last **FMAC** instruction as the pipeline flushes. These stalls are completely transparent to the programmer, but they do not make the algorithm go any faster. Therefore, you must make use of the possible parallel operations on the Am29050 to fill in these transparent stall cycles.

On the other RISC chips, the FPU instructions must also be properly

Listing 2—The common start-up code for the direct (slow) and custom (fast) implementation of an Am29050-based floating-point FIR digital filter.

```
.cputype 29050

.equ AB_PLUS_0, 4 ; for FMAC
.equ AB_PLUS_ACC, 0 ; for FMAC
.equ SUM 0 ; fp accumulator 0

; FORMAT and ROUNDING control
.equ SING_FP, 1 ; is single precision fp
.equ DOUBLE_FP, 2 ; is double precision fp
.equ INT, 0 ; is integer
.equ NEAREST, 0 ; round to nearest. integer
.equ SIGNED_INT, 0 ; is signed integer

; GENERAL REGISTER ALLOCATION
.set A20, gr127 ; store ADC address location
.set D2A, gr126 ; store DAC address location
.set ZERO, gr125 ; store ZERO constant
.set Yn, gr124 ; final value
.set Xn, lr94 ; temporary storage of ADC input

.set H0, gr64 ; also H94 FIR coefficients
.set H1, gr65 ; also H93
.set H2, gr66 ; also H92
.set H3, gr67 ; also H91

; .....
.set H45, gr110 ; also H49
.set H46, gr111 ; also H48
.set H47, gr112 ; center tap

.set X94, lr0 ; input delayed by 94 sample periods
.set X93, lr1 ; stored in CIRCULAR buffer
.set X92, lr2 ; local registers
.set X91, lr3

; .....
.set X48, lr46
.set X47, lr47
.set X46, lr48

; .....
.set X3, lr91 ; input delayed by 3 sample periods
.set X2, lr92 ; input. delayed by 2 sample periods
.set X1, lr93 ; input delayed by 1 sample periods
.set X0, lr94 ; current input.

; ADC and DAC locations and fp filter coefficients locations
.equ AtoAddress, 0x80000000
.equ DtoAddress, 0x80000004
.equ FILTERCOEFF, 0x80000100

.text
.global start
start:
; Set FP Acc do d.p. mode for 64-bit accumulators
; Ignore all FP Exceptions
ntsrirm fpe. 0x43F ; FP environment

const gr127, FILTERCOEFF ; read in filter coefficients
consth gr127, FILTERCOEFF ; store as floats
ntsrirm cr. 47 ; bring in 48 coefficients
loadm 0, 0, H0, gr127

const A20, AtoAddress ; establish constants
consth A20, AtoAddress
const D2A, DtoAddress
consth D2A, DtoAddress
const ZERO, $float(0.0)
consth ZERO, $float(0.0)
```

Listing 3 The FIR pseudocode from Listing 1 can be implemented directly in Am29050 assembler code.

```
.equ    LENGTH, 95
.equ    SYMMETRICPART. (LENGTH >> 1)

LOOP:
load    0, 0, Xn. A2D      ; read ADC and convert
CONVERT X0, Xn. SIGNED_INT, NEAREST, SING_FP, INT

FMAC    AB_PLUS_0, SUM ZERO, ZERO ; sum = 0;
                                     ; calculate sum
FMAC    AB_PLUS_ACC, SUM X0, H0    ; sum += x0 * h0
FMAC    AB_PLUS_ACC, SUM X94, H0   ; sum += x94 * h0
FMAC    AB_PLUS_ACC, SUM X1, H1    ; sum += x1 * h1
FMAC    AB_PLUS_ACC, SUM X93, H1   ; sum += x93 * h1

.set    N, 1
.equ    COEFF. 64
.equ    DATA. &X0

.rep    (SYMMETRICPART - 2)
.set    N, N + 1
                                     ; sum += xn * hn
FMAC    AB_PLUS_ACC, SUM   %% (DATA-N),   %% (COEFF+N)
                                     ; sum += x(LENGTH-n-1) * hn
FMAC    AB_PLUS_ACC, SUM   %% (DATA-LENGTH+N+1),   %% (COEFF+N)
.endr

FMAC    AB_PLUS_ACC, SUM X47, H47 ; center tap

MFACC   Yn. SING_FP, SUM ; yn = (int) sum
CONVERT Yn. Yn. SIGNED_INT, NEAREST, INT, SING_FP

store   0, 0, Yn. D2A      ; store at DAC
add     gr1, gr1, 4        ; adjust circular buffer
jmp     LOOP
NOP
```

ordered to keep the pipeline full. In addition, the memory accesses for data and filter coefficients and circular buffer operations must be more efficiently manipulated for a better performance.

IMPLEMENTATION OF A FIR FILTER-ATTEMPT II

The major problem with the direct implementation of the FIR filter is the code did not take into account the internal construction of the pipelined FPU. The code must be adjusted to keep this pipeline full. This change is made by rewriting the algorithm into a form that allows consecutive **FMAC** instructions on the same floating-point accumulator every *n*th cycle, at which time the values will be available. (This adjustment requires four and three partial sums for the Am29050 and the i860, respectively.) The pseudocode for the new loop part of the high-speed FIR filter implementation is shown in

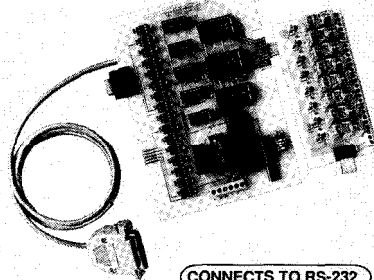
Listing 4. In this version, the calculation of the filter output, *y[n AT]*, is performed using four partial sums added together to get the final result. This approach (schematically shown in Table 2) allows the floating-point pipeline to be kept full.

The actual Am29050 code requires (*N* + 20) instructions with only ten cycles remaining where you must wait for an operation to complete or to flush the pipeline. Thus, the 95-tap filter completes in 125 cycles (3.125 μ s at 40 MHz)-a 350% time improvement over the direct implementation. Because this system is dedicated, you can further customize the code through some minor reordering, and fill in the transparent **STALL** cycles with useful operations by overlapping two filter cycles.

In Listing 5, the code moved into the **STALL** cycles is indicated by stars (* ●) in the comment field. Moving instructions for maximum speed is not

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS



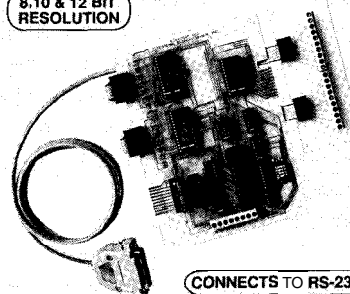
CONNECTS TO RS-232

AR-16 RELAY INTERFACE.....\$ 89.95

Two 8 channel relay output ports are provided for control of up to 16 relays (expandable to 128 relays using EX-16 expansion cards). Each relay output port connects to a relay card or terminal block. A variety of relay cards and relays are stocked. Call for more info. RS-422 available (distances to 4,000 feet). PS-4 port selector may be used to control satellite AR-16 interfaces. (up to 16,384 relays) RD-8 REED RELAY CARD (8 relays, 10 VA).....\$ 49.95 RH-8 RELAY CARD (10 amp SPDT 277 VAC).....\$ 69.95 EX-16 RELAY EXPANSION CARD (16 channel)....\$ 59.95

ANALOG TO DIGITAL

8.10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER (16 channel, 8 bit).....\$ 99.95

Input temperature, voltage, amperage, pressure, energy usage, energy demand, light levels, joystick movement and a wide variety of other types of analog signals. Inputs may be expanded to 32 analog or 128 status inputs using the AD-16 or ST-32 expansion cards. 112 relays may be controlled using EX-16 expansion cards. Analog inputs may be configured for temperature input using the TE-8 temperature input conversion. RS-422 available. PS-4 port selector may be used to connect satellite ADC-16 interfaces (up to 4,096 analog inputs/16,384 status inputs and 14,336 relays). Call for info on 10 & 12 bit converters. (terminal block and cable sold separately)

ST-32 STATUS EXPANSION CARD.....\$ 79.95

Input on/off status of relays, switches, HVAC equipment, thermostats, security devices, smoke detectors and other devices including keypads and binary coded outputs. Provides 32 status inputs (opto isolators sold separately).

TE-8 TEMPERATURE INPUT CONVERSION.....\$ 49.95

Includes 8 temperature sensors & terminal block. Temperature range is min: 40 to 145 degrees F.

PS-4 PORT SELECTOR (4 channels RS-422).....\$ 79.95

Converts on RS-232 port into 4 selectable RS-422 ports.

TOUCH TONE DECODER and other serial interfacing products available. Call for free information packet.

• **FULL TECHNICAL SUPPORT**..Provided over the telephone by our staff. EACH ORDER INCLUDES A FREE DISK WITH PROGRAMMING EXAMPLES IN BASIC, C AND ASSEMBLY LANGUAGE. A detailed technical reference manual is also included.

• **HIGH RELIABILITY** ..engineered for continuous 24 hour industrial applications. All ICs socketed.

• **Use with IBM and compatibles**, Tandy, Apple, Mac and most other computers with RS-232 or RS-422 ports. All standard baud rates and protocols may be used (50 to 19,200 baud)

Use our 800 number to order FREE INFORMATION PACKET. Technical Information (614) 464-4470.

24 HOUR ORDER LINE (800) 842-7714

Visa-Mastercard-American Express-COD

International & Domestic FAX (614) 464-9656

Use for information, technical support & orders

ELECTRONIC ENERGY CONTROL, INC.

380 South Fifth Street, Suite 604

Columbus, Ohio 43215

	<u>Buses</u>		<u>Multiplier</u>		<u>Adder</u>				<u>Registers</u>	
<u>Instr.</u>	<u>A</u>	<u>B</u>	<u>MT</u>	<u>PS</u>	<u>DN</u>	<u>AD</u>	<u>RN</u>	<u>RU</u>	<u>ACC0</u>	<u>LR94</u>
CONV		Xn								
FMAC	0	0			Xn					
FMAC	-s-	-s-	0 x 0			Xn				
-s-	-s-	-s-	-s-	0 x 0			Xn			
-s-	HD	X0	-s-	-s-	0 x 0			Xn		
FMAC	HD	x94	HOX0	-s-	-s-	0 x 0				X0
FMAC	H1	X1	HOX94	HOX0	-s-	-s-	0 x 0			
-s-	-s-	-s-	-s-	-s-	-s-	-s-	-s-	0 x 0		
FMAC	H1	x93	H1X1	HOX94	HOX0+S	-s-	-s-	-s-	S	
-s-	-s-	-s-	-s-	-s-	-s-	HOX0+S	-s-	-s-		
-s-	-s-	-s-	-s-	-s-	-s-	-s-	HOX0+S	-s-		
-s-	-s-	-s-	-s-	-s-	-s-	-s-	-s-	HOX0+S		
FMAC	H2	x2	H1X93	H1X1	H1X1+S1	-s-	-s-	-s-	S1	
-s-	-s-	-s-	-s-	-s-	-s-	H1X1+S1	-s-	-s-		

Table 1--The FPU's pipeline blocks(-s-) during consecutive FMAC operations using a single FPU accumulator.

straightforward because of the difficulty when six-deep pipeline FMAC instructions are mixed with three- or four-deep pipeline instructions (DADD, CONVERT), so the instructions may block each other. Memory is accessed when using the LOAD and STORE instructions to read and load the ADC and the DAC. If the external memory is slow, these instructions may need to be moved elsewhere in the loop so memory values are available at the right moment.

The importance of an intelligent DSP compiler becomes obvious. Combining the accumulator clearing with the first four multiplicative operations further reduces the number of cycles, but you must reorder the first eight multiplicative operations. This step ensures the CONVERT operation on the ADC input value does not stall the pipeline due to a result being unavailable for the X0 x HO operation. In order to keep up the

accuracy of the summation, the new program brings out the FPU accumulators into twin global registers and uses DADD to add up the partial sums. This move does not cause any time penalties when doing a single-precision add because of the Am29050 FPU has 64-bit buses to and from the main register bank.

The code in Listing 5 was finalized in conjunction with an Am29050 simulator and actually operates in N + 13 instructions with only three STALL cycles remaining. For filters with a slow digitization rate, the processor would be put into a tight infinite loop at the start of the FIR calculation. An interrupt signal from the timing logic would then initiate another calculation cycle. In this situation, the remaining STALLs are unimportant.

The 95-tap Am29050 FIR digital filter implementation completes 192 floating-point operations in 111 cycles (2.78 μ s at 40 MHz, or nearly 68

megaFLOPS)-not bad for a nonspecialized chip.

Fixing the FPU pipeline on the MC88100 and the i860 greatly speeds them. However, they remain 2.5 times slower than the Am29050 because they have only a small 32-floating-point register bank attached to the FPU compared to the 192 registers of the Am29050. The MC88100 and i860 registers must be continually reloaded, taking additional cycles for every FPU operation.

However, the i860 RISC has a couple of aces up its DSP sleeve. The on-board instruction cache can be switched to a dual-instruction capability that allows the registers to be sent to the FPU while simultaneously permitting other registers to be reloaded from the data cache. This feature would improve things, but it still would not allow the superscalar i860 to be as fast as the scalar Am29050. The circular buffer over-

	<u>Buses</u>		<u>Multiplier</u>		<u>Adder</u>				<u>Registers</u>				
<u>Instr.</u>	<u>A</u>	<u>B</u>	<u>MT</u>	<u>PS</u>	<u>DN</u>	<u>AD</u>	<u>RN</u>	<u>RU</u>	<u>ACC0</u>	<u>ACC1</u>	<u>ACC2</u>	<u>ACC3</u>	<u>LR94</u>
CONV	-	Xn											
FMAC	0	0			Xn								
FMAC	0	0	0 x 0			Xn							
FMAC	0	0	0 x 0	0 x 0			Xn						
FMAC	0	0	0 x 0	0 x 0	0x0			Xn					
FMAC	HD	X0	0 x 0	0 x 0	0x0	0x0							X0
FMAC	HD	X94	HOX0	0x0	0x0	0x0	0x0						
FMAC	H1	X1	H1X94	HOX0	0x0	0x0	0x0	0 x 0					
FMAC	H1	X93	H1X1	HOX94	HOX0+S	0x0	0x0	0 x 0	S				
FMAC	H2	X2	H1X93	H1X1	HOX94+T	HOX0+S	0x0	0 x 0		T			
FMAC	H2	X92	H2X2	H1X93	H1X1+U	HOX94+T	HOX0+S	0 x 0			U		
FMAC	H3	X3	H2X92	H2X2	H1X93+V	H1X1+U	HOX94+T	HOX0+S				V	
FMAC	H3	X91	H3X3	H2X92	H2X2+S1	H1X93+V	H1X1+U	HOX94+T	S1				
FMAC	H4	X4	H3X91	H3X3	H2X92+T1	H2X2+S1	H1X93+V	H1X1+U		T1			

Table 2--The Am29050 FPU pipeline can be kept full by splitting the FIR calculation into four partial sums.

Listing 4-The pseudocode for an FIR implementation using partial sums to overcome the FPU stalls.

```

/* Assume automatic circular buffer wrap-around handling */

float sum[4];          /* filter partial sums */
int count, whichsum

/* New loop to perform the convolution using partial sums */

for (count = 0; count < 4; count++)    sum[count] = 0.0;

for (count = 0, whichsum = 0; count < NUMCOEFFS 1; count++) {
    sum[(whichsum++) % 4] += *xrecent-- * *coeffcurrent;
    sum[(whichsum++) % 4] += *xancient++ * *coeffcurrent++;
}
sum[0] += *xancient++ * *coeffcurrent++;    /* center tap */

/* Convert and write out */
yn = (int) (sum[0] + sum[1] + sum[2] + sum[3]);

```

head still has to be handled, and too many registers need to be reloaded in the cycles available. Therefore, the i860 changes gear and switches into a 128-bit quadruple fetch mode from the data cache to allow the loading of four registers simultaneously. A couple of cycles of the dual-instruction i860 FIR code is shown in Listing 6 for an example. Because of the structure of the i860 FPU pipeline and assembler code, the filter sum is implicitly broken up into three sections rather than explicitly into four as in the Am29050 implementation.

Note that, unlike the scalar Am29050, the superscalar i860 has some fancy addressing modes more reminiscent of a CISC rather than a RISC chip. The data {X 1-X 8} and coefficients {H 1-H 8} are fetched into the FPU registers using pointers (X p and H p), with the circular data buffer starting at memory location Xba se.

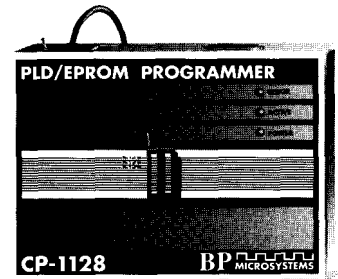
Note also that the **d . f** nop stalls the FPU once every eight cycles in order to steal the time to update the circular buffer using the integer APU. (The second circular buffer update is performed in parallel with an existing FPU operation and causes no overhead.) Although not obvious in this code, the i860 **FMC** instruction, **ml 2 a pm sd**, takes three operands, exposing the architecture of the FPU in an unusual way. In any one cycle, these operands correspond to those two just about to enter the FPU pipeline and the value being stored

from the **FMC** instruction that started six cycles back.

The structure of the six-deep i860 FPU pipeline means that filling and flushing the pipeline at the start and end of each summation requires a total of six (nonproductive) cycles. This requirement is not important for the long 95-tap FIR filter, but would be significant for shorter DSP loops. Like the Am29050, the i860 will also stall when adding together partial sums. The conversion of the integer ADC value into floats, and vice-versa for the DAC value, also causes some additional overhead because of the difficulty of overlapping these operations with other FPU operations. (The i860 does not have an explicit **CONVERT** instruction.) These factors mean that an N-tap FIR digital filter will take $\sim 1.13N + 24$ cycles.

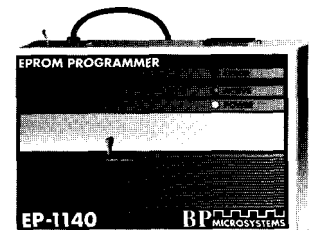
The 95-tap i860 FIR digital filter implementation completes 192 floating-point operations in 132 cycles (3.3 μ s at 40 MHz, or 58 mega-FLOPS)- also no slouch as a DSP processor.

As I mentioned earlier, I teach comparative computer architecture. Under the AMD University Support Program, I was able to obtain some Am29000 evaluation boards, which I later upgraded with the pin-compatible Am29050 chips. I have yet to approach Intel to see if they would also provide me with a classroom full of i860 boards to allow a similar extensive DSP experimentation. Therefore, I am



CP-1128
Combination Programmer
\$1295.00

- ✓ Supports AMD's MACH110/210/120/130/230 EPPLDs, Altera's 900, 1800-series and MAX EPPLDs, Cypress' CY7C36 1, Lattice's LSI1032 & pLSI1032, National Semiconductor's -5D & -7D devices and MAPL devices.
- ✓ Qualified and recommended by AMD, Lattice, National Semiconductor, Signetics and others.
- ✓ Utilizes only manufacturer approved programming algorithms.
- ✓ Supports upto 28-pin E/EPROMs and bipolar PROMs including the microwire²C devices.
- ✓ Supports Dallas Semiconductor NVRAMs and TI DSP320, Microchip PIC microcontrollers
- ✓ Lifetime FREE software updates available via BBS and US Mail
- ✓ Call for a DEMO disk and literature pack.
- ✓ Made in the USA



EP-1140
E/EPROM & μ controller
Programmer
\$895.00

- ✓ The EP-1140 supports NEC's 27C8001, 8-Mbit EPROM, all 27C2404-Mbit 16 bit EPROMs, FLASH EPROMs, NVRAMs and all microwire EPROMs.
- ✓ All Intel, AMD, and Signetics 40-pin μ controllers supported directly.
- ✓ Qualified and recommended by Intel, Signetics, National Semiconductor, and others
- ✓ Lifetime FREE software updates available via BBS and US Mail
- ✓ Risk-free thirty-day money-back guarantee
- ✓ Made in the USA

With lifetime FREE software updates, you can't go wrong and BP Microsystems offers a thirtyday money-back guarantee to ensure product satisfaction. Remember, BP Microsystems is...

The Engineer's **Programmer**TM

BP MICROSYSTEMS

10681 Haddington • Houston, TX 77043-3239
(713) 461-9430 • FAX (713) 461-7413

© 1991 BP Microsystems, Inc.

#109

not as familiar with the i860 as with the Am29050 chips, so a more experienced programmer might tighten the i860 code by a few cycles.

EVALUATION OF RISC CHIPS FOR DSP PURPOSES

As I have shown, some RISC chips have many of the features useful for DSP applications. Most importantly, they are fast (30-50 MHz). An on-board register stack or a quadruple-ported data cache can store data and coefficients without having to access slow external memory. Some RISC chips have a separate instruction bus to keep data and instruction fetches from competing for resources. Another advantage the RISC chips have over dedicated DSP chips is an internal architecture more exposed to the programmer/compiler, giving full control of the FPU pipeline to maximize the filter throughput.

The Motorola DSP56200 is an interesting DSP chip to compare to the RISC FIR performance. The DSP56200 was designed for FIR operations and has some fancy features. It has a 16-bit input with a 40-bit integer data path for the accumulator operation compared to the 64-bit floating-point path on the Am29050. What is neat about the DSP56200, in a comparative architectural sense, is it was specifically designed (c. 1985) with simple interchip serial connections to allow multiple DSP56200s to operate in parallel if a single chip does not perform quickly enough. I adapted the timings shown in Table 3 from data sheet results for DSP and RISC chips.

For nonspecialized DSP chips, RISCs do not perform too badly. The Am29050 and i860 RISC outperform the other RISCs because of their register- and memory-handling capabilities. For other DSP operations that make less use of a large number of registers, the timings are closer (e.g., for the shorter-length Infinite Impulse Response [IIR] digital filter). The availability of an **FMAC** instruction is also very significant in this DSP application.

The Am29050 and i860 RISCs seem to be handling themselves as reasonable (if not perfect) general-

Listing 5—Two overlapped filter cycles are used for maximum efficiency in the final partial sum version of the FIR filter code. Instructions for the second filter cycle are moved into the stall positions (**) of the initial filter cycle.

```

: changes to general register definitions
.set      TEMPO, gr122 ; external copies of FP ACCUMULATORS
.set      TEMP1, gr120 ; using twin registers for double
.set      TEMP2, gr118 ; precision storage
.set      TEMP3, gr116
.set      Yn, TEMPO

: changes to FP accumulator names
.equ      SUM0, 0
.equ      SUM1, 1
.equ      SUM2, 2
.equ      SUM3, 3

load      0, 0, Xn, A2D, .x0 = (float) ReadA2D()
CONVERT   X0, Xn, SIGNED_INT, NEAREST, SING_FP, INT

FMAC      AB_PLUS_0, SUM0, X1, H1 ; ** sum[0] = x1 * h1
FMAC      AB_PLUS_0, SUM1, X93, H1 ; ** sum[1] = x93 * h1
FMAC      AB_PLUS_0, SUM2, X3, H3 ; ** sum[2] = x3 * h3
FMAC      AB_PLUS_0, SUM3, X91, H3 ; ** sum[3] = x91 * h3

FMAC      AB_PLUS_ACC, SUM0, X2, H2 ; ** sum[0] += x2 * h2
FMAC      AB_PLUS_ACC, SUM1, X92, H2 ; ** sum[1] += x92 * h2
LOOP:
FMAC      AB_PLUS_ACC, SUM2, X94, H0 ; ** sum[2] += x94 * h0
FMAC      AB_PLUS_ACC, SUM3, X0, H0 ; ** sum[3] += x0 * h0

.equ      LENGTH, 95
.equ      SYMMETRICPART, (LENGTH >> 2)
.set      DATA, &X0
.set      COEFF, 64
.set      N, 2 ; **
.set      (SYMMETRICPART - 2) ; **
.rep      N, N + 2
FMAC      AB_PLUS_ACC, SUM0, %(DATA-N), %(COEFF+N)
FMAC      AB_PLUS_ACC, SUM1, %(DATA-LENGTH+N+1), %(COEFF+N)
FMAC      AB_PLUS_ACC, SUM2, %(DATA-N-1), %(COEFF+N+1)
FMAC      AB_PLUS_ACC, SUM3, %(DATA-LENGTH+N+2), %(COEFF+N+1)
.endr

FMAC      AB_PLUS_ACC, SUM0, X48, H46 ; finish off summation
FMAC      AB_PLUS_ACC, SUM1, X46, H46
FMAC      AB_PLUS_ACC, SUM0, X47, H47 ; center tap
add        gr1, gr1, 4 ; ** Adjust circular buffer
store      0, 0, Yn, D2A ; ** WriteD2A(yn) for previous cycle

MFACC      TEMP2, DOUBLE_FP, SUM3 ; temp2 = sum3
MFACC      TEMP3, DOUBLE_FP, SUM0
MFACC      TEMP0, DOUBLE_FP, SUM1
MFACC      TEMP1, DOUBLE_FP, SUM2

FMAC      AB_PLUS_0, SUM2, X3, H3 ; **
DADD      TEMP2, TEMP2, TEMP3 ; temp2 += temp3
FMAC      AB_PLUS_0, SUM3, X90, H3 ; **
DADD      TEMP0, TEMP0, TEMP1 ; temp0 += temp1
FMAC      AB_PLUS_0, SUM0, X2, H2 ; **
FMAC      AB_PLUS_0, SUM1, X91, H2 ; **
DADD      TEMP0, TEMP0, TEMP2 ; temp0 = temp0 + temp2

FMAC      AB_PLUS_ACC, SUM0, X1, H1 ; ** sum[0] += x1 * h1
FMAC      AB_PLUS_ACC, SUM1, X92, H1 ; ** sum[1] += x92 * h1
load      0, 0, Xn, A2D ; **
CONVERT   X0, Xn, SIGNED_INT, NEAREST, SING_FP, INT ; **
jmp        LOOP

; yn = (int) temp0
CONVERT   Yn, TEMP0, SIGNED_INT, NEAREST, INT, DOUBLE_FP

```

DSP chips

single DSP56200	10 MHz	10.4 μ s
dual DSP56200	10 MHz	3.25 μ s
quadruple DSP 56200	10 MHz	2.71 μ s
single DSP56001	33 MHz	5.64 μ s
single DSP96002 (fp)	40 MHz	4.96 μ s
single TMS32025	50 MHz	7.84 μ s
single TMS32030 (fp)	33 MHz	4.75 μ s

RISC chips

single Am29050 (fp)	40 MHz	2.78 μ s
single i860 (fp)	40 MHz	3.3 μ s
single MC881 00 (fp)	33 MHz	9.75 μ s
single SPARC (fp)	33 MHz	>9.75 μ s

Table 3—In the case of a 95-tap FIR filter, some processors do better than others due to differences in architectures. The Am29050, in fact, beats almost all of the DSP chips in an application traditionally reserved strict/y for such chips.

purpose DSP-capable chips. What is missing?

- The Am29050 could do with a *floating-point accumulator-to-accumulator add* operation similar to that found on the Am29027 coprocessor used in conjunction with the Am29000 integer RISC chip. Examining the code for the final FIR filter indicates the FPU accumulators are transferred to local registers simply to bring them back into the FPU so an accumulator-to-accumulator add can occur. This time does not incur a substantial overhead for a long (95-tap) FIR filter because you can fill the **STALL** cycles associated with pipeline filling and flushing with useful instructions. However, it could be critical for the shorter loops often present in other DSP applications.

- *A problem with the i860 is also associated with the way its FPU pipeline is filled and flushed. Compared to the Am29050, the i860 has an additional 6-cycle overhead, critical for shorter DSP loops. It also suffers from **STA L L** problems when adding together partial sums even with the capability of the direct floating-point accumulator-to-accumulator add lacking in the Am29050.

- *For applications requiring a very large number of continual memory accesses [e.g., a 150-tap FIR filter], what would be useful is if the Am29050 had the superscalar i860 capability of being able to reload one bank of floating-point registers while using another bank. The scalar

Am29050 already has a *load multiple* instruction, **LOADM**, and appears to have sufficient internal buses to allow this operation to occur in conjunction with FPU operations. However, because of the complex data dependencies that might occur, this parallelism is currently blocked by the Am29050 internal logic. However, I would not want to go to the other extreme with no checks on conflicts, which makes programming difficult. With the large number of available registers on the Am29050,

handling the logic associated with reloading one "bank" (say a group of 8 or 16) while using another should be possible.

CONCLUSION

In this application, I started off with an academic exercise to evaluate the RISC chips in a simple DSP application—a FIR digital filter. I ended up with some interesting conclusions about the similarity between the RISC and specialized DSP chips. In particular, the Advanced Micro Devices Am29050 and the Intel i860 RISC chips were shown to have many DSP capabilities.

- *These chips are fast (40 MHz); have a fast, high-precision, on-board FPU; and are capable of performing many simultaneous floating-point and integer operations simultaneously.

- *The RISC instruction set exposes the chip's internal operation to the programmer/compiler, permitting maximum pipeline efficiency at all points in a DSP algorithm.

- *Although the data coming into the FPU is 32 bits wide, the internal representation of the FPU accumulators allows 64-bit double-precision floating-point storage of intermediate results with no time penalty. This feature means that sufficient guard bits are available to ensure against underflow and overflow of DSP algorithms involving extensive summation or recursive operations.

- *Typical DSP applications involve extensive looping [branching]. For a

REALIZE THE POWER OF...

PARADIGM
LOCATE

PARADIGM

The Model for
Programming Productivity

3301 Country Club Road
Suite 2214
Endwell, NY 13760
(607) 748-5966
FAX: (607) 748-5968

All trademarks are property of their respective holders.

See us at the Embedded Systems Conference—Booth#305

Listing 6—The i860 FPU pipeline has a dual-instruction mode that can be used in FIR calculations. The filter sum is implicitly broken up into three sections rather than explicitly into four as in the Am29050.

d.m12apm.sd	X5. H5.	T1	fld.q	0(Hp), H1	; sum1 += X5 * H5, fetch 4 coefficients
d.m12apm.sd	X6. H6.	T1	add.s	16. Xp. Xp	; sum2 += X6 * H6. update data pointer
d. fnop			and	#\$1FF, Xp. Xp	; FPU stall while adjust circular buffer
d.m12apm.sd	X7. H7.	T1	add.s	32. Hp. Hp	; sum3 += X7 * H7. update coeff pointer
d.m12apm.sd	X8. H8.	T1	fld.q	16(Bp), 135	; sum1 += X8 * H8, fetch 4 coefficients
d.m12apm.sd	X1.	H1, T1	fld.q	Xp(Xbase), X5	; sum2 += X1 * H1, fetch 4 data values
					; sum1 += X5 * H6, now complete
d.m12apm.sd	X2. H2.	T1	add.s	16. Xp. Xp	; sum3 += X2 * H2. update data pointer
					; sum2 += X6 * H6, now complete
d.m12apm.sd	X3. H3.	T1	and	#\$1FF, Xp. Xp	; sum1 += X3 * H3. adjust buffer
					; sum3 += X7 * H7, now complete
d.m12apm.sd	X4. H4.	T1	fld.q	Xp(Xbase), X1	; sum2 += X4 * H4. fetch 4 data values

heavily pipelined system, this aspect could involve frequent inefficient pipeline stalls (flushes) while the new instructions are being fetched. The use of the **delayed jump** instruction overcomes much of this problem. The availability of an on-board Branch Target Cache (Am29050) or an instruction cache (i860) overcomes problems with slow external instruction memory.

- A register window connected to the FPU or the ability to reload a small floating-point register bank in parallel with floating-point operations is essential for high-speed DSP operation.

- A useful DSP feature apparently missing from the Am29050 is a floating-point accumulator-to-accumulator add implemented directly in the FPU. However, the presence of this instruction on the i860 did not provide additional speed because of the structure of its FPU pipeline. This feature can currently be implemented on the Am29050 using two existing instructions that expose the accumulator-to-accumulator add pipeline to the programmer. Such an approach may actually better control keeping the FPU pipeline full. Whether the lack of this instruction is a bug or a feature in DSP applications is a question answered only by further evaluation.

*Data access can compete for resources with the instruction fetches

in DSP loops. The i860 and the Am29050 avoid this problem by having separate instruction and data buses and an instruction cache, features not available on all RISC chips. However, it would be useful if the current Am29050 instruction prefetch buffer was actually a cache for DSP applications. As shown on the

of the tight DSP loops is done efficiently on RISC machines.

*By examining a real-time FIR digital filter application, I have hinted at possible RISC DSP applications in the telecommunications area. Economy of scale would bring the RISC prices down. Low-power standby modes and other useful DSP features could easily be added to RISC chips if market forces come to bear.

*For the 95-tap FIR digital filter discussed, the Am29050 and the i860 RISC outperformed many of the specialized DSP processors currently on the market.

All things considered, the Am29050 and i860 RISC appear to be fairly powerful, general, floating-point DSP processors. The optimum RISC DSP chip would appear to require the i860's dual-instruction and quadruple data fetch capability combined with the Am29050 register window and a combination of the two FPUs. □

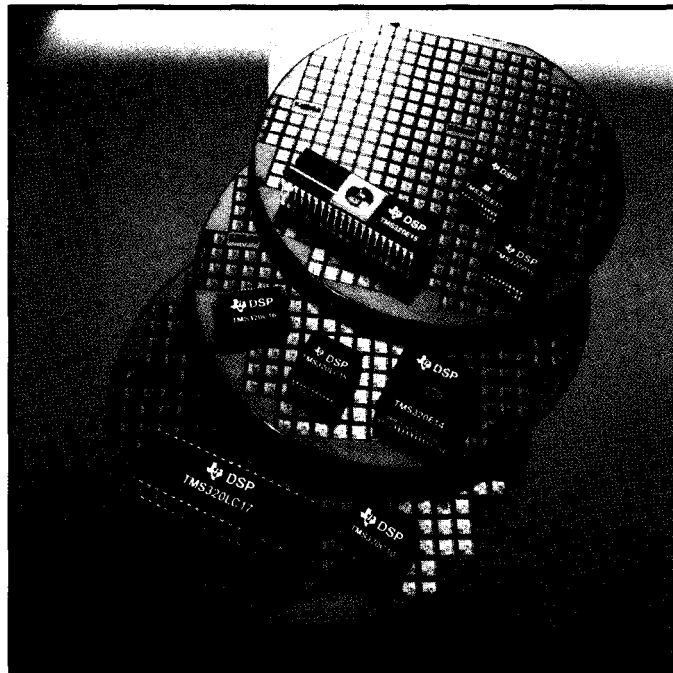


Photo 2—The TMS320 is a popular dedicated DSP chip that, in some applications, may be passed over for capable RISC chips.

i860, the ability to access twin instructions and quadruple data fetches in a single cycle from a reconfigured cache considerably improves speed.

- Clumsy FPU pipeline filling and flushing can produce heavy overhead on short DSP loops.

- A DSP-intelligent compiler is needed to make sure the optimization

Useful discussions on this application note were held with Dr. S. T. Nichols, Dr. L. E. Turner, and S. Worthington of the University of Calgary, Canada, and with Pat Eichenseer of the 29k support group in Austin, Texas. Am29050 RISC architecture evaluation boards were made available under the AMD University Support Program.

M. R. Smith holds a Ph.D. in Physics and is with the Department of Electrical and Computer Engineering at the University of Calgary, Canada. He teaches and does research into hardware and software for high-speed image processing.

FURTHER READING

T. J. Terrell, *Introduction to Digital Filters*, MacMillan Press, 1983.

Motorola, *DSP56200 Cascadable Adaptive Finite Impulse Response Digital Filter-Technical Data*, Motorola, 1986.

Texas Instruments Signal Processing Guides, *Digital Signal Processing Applications with the TMS320 family*, vol. I-IV, 1986-1991.

M. R. Smith, T. J. Smit, S. W. Nichols, S. T. Nichols, H. Orbay, and K. Campbell, "A Hardware Implementation of an Auto-regressive Algorithm," *Measurement in Science and Technology*, 1, 1000, 1991.

J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms and Application*, 2nd Ed., MacMillan Publishing Company, New York, 1992.

Advanced Micro Devices, *Am29050 32-Bit Streamlined Instruction Processor, User's Manual*, 1991.

Cypress Semiconductors, *SPARC RISC User's Guide*, 1990.

Intel, *i860 Microprocessor Family Programmer's Reference Manual*, 1991.

LSI Logic, *SPARC Architecture Manual*, 1990.

Motorola, *M88100 RISC Microprocessor User's Manual*, 2nd ed., 1990.

IRS

- 401 Very Useful
- 402 Moderately Useful
- 403 Not Useful



MultiTask!™ Execs and GOFAST™ Math Speed Time-to-Market

Zap your application with peak performance using tested code and expert support. Control real-time scheduling with MultiTask! source code executives, or ROM-able, re-entrant GOFAST floating point:

- *Replace 80x87 MATH COPROCESSORS;
- *Drop-in IEEE SOURCE LIBRARIES;
- *Link-and-Go with C compilers: Intel®, Microsoft®, Borland®, WATCOM®, Zortech, Metaware®, and more...

Solutions for 80386/486 PROTECTED-MODE, 80x86 & V-Series, Z80/180/64180, 8085, 68xxx, 68HC16, 68HC 11, 6801, 6809, 8051, 80196, i960, R3000, SPARC® and more. Call for free information diskettes today. PHONE 503-641-8446; FAX 503-644-2413; USA TOLL FREE 800-356-7097.



14215 NW Science Park Drive
Portland, OR 97229

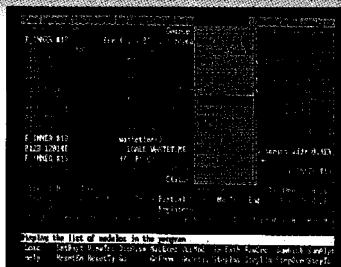
U S SOFTWARE®

© 1992 US Software Corporation. GOFAST and MultiTask! are trademarks of US Software Corporation. All other trademarks belong to their respective owners.

#110

See us at the Embedded Systems Conference—Booth#738

8051 68HC11 COP8 68HC05



iceMASTER™ Improved User Interface Features

Rental And 10-Day Trials Available

- **iceMASTER delivers productivity:** easy to learn, easy to use and fast!
- **Hyperlinked On-line help** guides you through the emulation process.
- **iceMASTER is FAST!** The 115.2K baud serial link keeps typical download times to under 3 seconds using a standard COMM port!
- **Broad support of derivative devices.**
- **Flexible user interface:** you can completely configure the windows for size, content, location and color.
- **iceMASTER is convenient!** It connects easily to your PC, requires no disassembly, nor does it take up any expansion slots. It works on any PC (DOS or OS/2), Micro Channel or EISA. Even Laptops!
- **Supports source level debug** (C and PL/M) and source level trace. 4K trace buffer with advanced searching and filtering capabilities.
- **Now quick-breaks, virtual memory and mouse support.**

**NEW
68HC05!**

Call today for **FREE DEMO DISK!**
Call today to ask about **FREE 8051 Macro Assembler!**
(800) METAICE (800) 638-2423

MetaLink®
Corporation

MetaLink Corporation P.O. Box 1329 Chandler, AZ 85244-1329 Phone: (602) 926-0797 FAX: (602) 926-1198 TELEX: 4998050MTLNK

#111

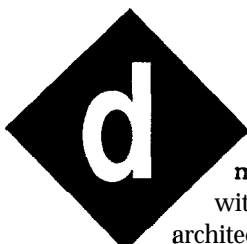
See us at the Embedded Systems Conference—Booth#604

Analyze Voice in the Palm of Your Hand

A spectrum analyzer is normally made with banks of analog filters. Using a popular DSP, this project will display the frequency content of a voice on an oscilloscope screen using just four chips and some glue.

FEATURE ARTICLE

Gerald McGuire

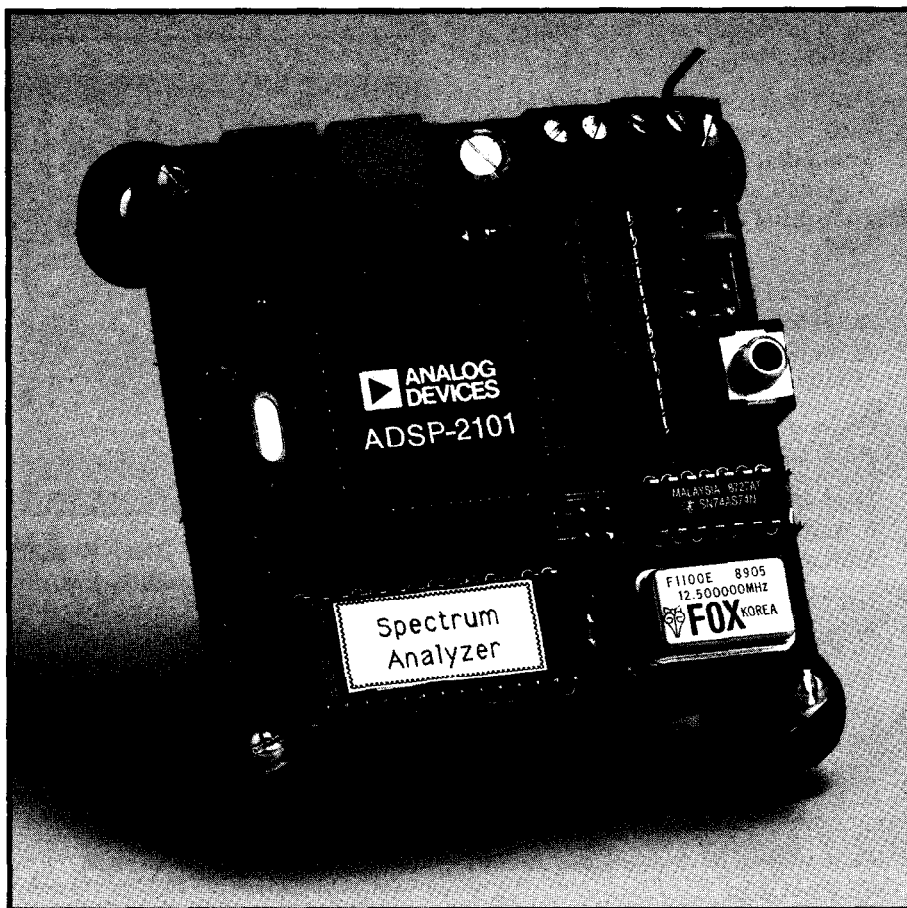


DSP chips are microprocessors with an internal architecture and structure optimized for calculation, especially where calculations involve the analysis of real-world-based signals. In many practical applications, the frequency-domain analysis of a time-domain signal is necessary. I will focus on the very common Fast Fourier Transform (FFT) and its implementation in real time on a digital signal

processor, the Analog Devices ADSP-2101. By using a DSP-based FFT analyzer interposed between a microphone and an oscilloscope, you can display the spectrum of the voice-band signal (0 to 4 kHz) in real time on the scope screen.

This analyzer runs in real time, digitizing samples from a voice-band ADC at a rate of 8 kilosamples per second. Frequency domain values are calculated and transferred via the DAC portion of the analyzer for display on an oscilloscope. Conversions are done using the ADSP-28MSPO2, a voice-band sampling ADC and DAC in a single package. A block diagram overview is shown in Figure 1, the detailed schematic is in Figure 2.

The analog output of the circuit is a time waveform that displays, in sequence, all the bin energies as calculated by the FFT, where a *bin* describes the amplitude of energy present within a narrow frequency range [the frequency range is determined by the sampling rate and the



The hand-held voice-band spectrum analyzer uses the ADSP-2101 DSP, ADSP-28MSPO2 ADC/DAC, 74LS74, and EPROM to display incoming audio's frequency content on an oscilloscope screen.

point size for the FFT). In this implementation, the overall range of 0-4 kHz is divided into 128 bins, with a frequency resolution of 3.125 Hz each. By connecting the analog output to the scope's vertical input, the vertical amplitude versus horizontal time scales of the scope display represents spectral energy (amplitude) versus frequency (time).

Analog input from the microphone is single ended, or 3.156 volts peak to peak. The output to the scope is either differential (6.3 12 volts peak to peak) or single ended (3.156 volts peak to peak) and is connected directly to the scope. Typical settings are 2 volts per division for the vertical scale (with 10x probe) and 2 ms per division for the horizontal time base. The entire display takes 7.3 divisions at 2 ms per division.

The entire circuit is constructed from six components: an ADSP-2101 (DSP microcomputer), an EPROM

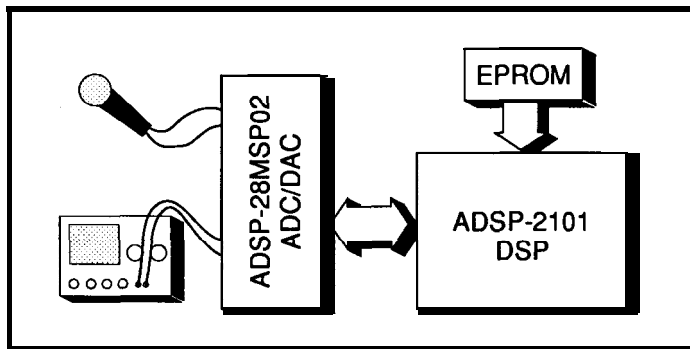


Figure 1—The voice-band spectrum analyzer takes its input from a microphone and displays its results on an oscilloscope.

containing the FFT code, a microprocessor-grade crystal (10-MHz parallel resonant), an ADSP-28MSP02 single-chip voice-band ADC and DAC, a CMOS 13-MHz oscillator chip (for the converter), and a 74LS74 D flip-flop.

THE ADSP-2101

The ADSP-2101 is a single-chip DSP microcomputer that contains three flexible computation units: a 16-bit ALU, a multiplier-accumulator (MAC) with a 40-bit accumulation register, and a barrel shifter. The

computation units can perform an operation in a single instruction cycle. As a result, the fastest version ADSP-2101 performs a multiply/accumulate with 40 bits of result in a single 60-ns cycle. These single-cycle computation units are what enable the DSP to execute a numerically intensive application like the FFT in real time. For this project, I used a slower, less costly 10-MHz [100-ns cycle time] ADSP-2101.

The chip uses a Harvard architecture with separate program and data memories (in contrast to the conventional microprocessor with von Neumann architecture that combines data and program in a single memory). In addition, the processor contains two independent *data address generators*. These devices are capable of the complex data addressing required for the FFT. In a single cycle, *simultaneously* with the calculation, the

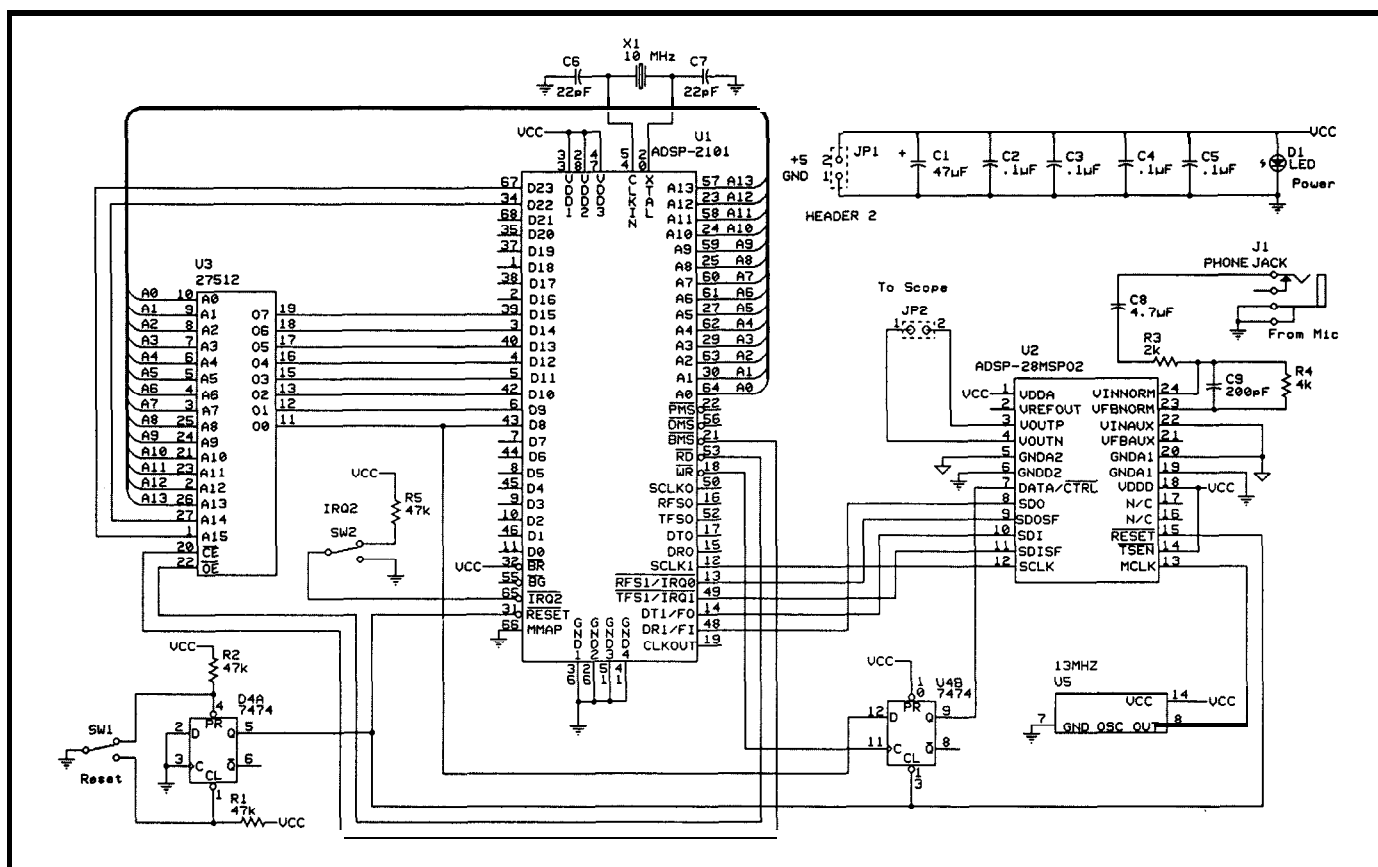


Figure 2—The ADSP-2101 loads its program from EPROM on reset and executes from then on completely from internal memory. The ADSP-28MSP02 combines ADC and DAC into a single chip and communicates directly with the DSP chip. The only other chip required is a single dual latch.

8051 Family Tools

In Circuit Emulators

The DryICE Plus is a modular emulator designed so you can get maximum flexibility from your emulator purchase. The base unit contains all the hardware necessary to support pods containing many of the most popular members of the 8051 family of embedded control microprocessors. Buy one base unit, and select one or all of the pods you need to do the job at a much reduced cost. You get the same great functionality found in our popular DryICE 8031 emulator plus real-time Execute-to-Breakpoint, Line-by-Line Assembler, and much more. And the price is (almost) unbelievable! (Yes, it works with the Mac, too!)

Base Unit (w/RS-232 IF) -- \$299

Available Pods: \$149 each

8031/32, 80C31/32, 80C154, 80C451, 80C535, 80C552/562, 80C652, 80C51FA, 8751/52, 87C51/52.

Call about 87C751/752 support

16K Trace Buffer option: Avail. 2nd Qtr '92
Standard 8031 DryICE -- Still only **\$199**

Enhanced 8031 DryICE -- **\$269**

8051 Simulation

The 8051SIM software package speeds the development of 8051 family programs by allowing execution and debug without a target system. The 8051 SIMULATOR is a screen oriented, menu command driven program doubling as a great learning tool. \$99.

Single Board Computers

8031SBC - A fast and inexpensive way to implement an embedded controller.

8031/32 processor, 8+ parallel I/O, up to 2 RS232 serial ports, +5 volt operation. The development board option allows simple debugging of 8031/51 family programs. **\$99ea**

80C552SBC - 10 bit 8 ch. A/D, 2 PWM, 1 RS232 & 2 RS232/422/485 serial ports, sockets for 64k ROM, 64k RAM, +5 volt operation; optional RT Clock w/ battery, 2k EEPROM. Development board version available. **Call for pricing!**

**Call for your custom product needs.
Free Quote - Quick Response**



HiTech Equipment Corp
9400 Activity Road
San Diego, CA 92126
[FAX: (619) 530-1458]

(619) 566-1 892

FFT Basics

There are several methods of converting time information into its frequency components. This spectrum analyzer is based on a Radix-4 Decimation-in-Time (DIT) Fast Fourier Transform (FFT). The FFT is perhaps the most often used algorithm for calculating the frequency components of a signal. In order to describe the FFT algorithm, I will mention the Fourier Series, continuous time Fourier Transform, and the Discrete Fourier Transform (DFT). The FFT is a shortcut to the implementation of the DFT. It is not an approximation, but it is an algorithm that efficiently implements the DFT.

The Fourier Series is a decomposition of a continuous time periodic signal into a sum of sinusoidal components. Any periodic signal can be represented as a combination of sinusoids. The Fourier Transform is a similar decomposition of a continuous time signal, with finite energy. Both the Fourier Series and the Fourier Transform operate on continuous time signals.

The DFT performs the Fourier Transform on sampled data points. Equation 1 shows the Fourier transform, where $X(F)$ is the Fourier Transform of the continuous time signal $x(t)$. $X(F)$ is a function of the variable F .

$$X(F) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi Ft} dt \quad (1)$$

A similar analysis can be performed on discrete time signals. Equation 2 shows the DFT, where $X(w)$ represents the frequency content of the sampled signal $x(n)$.

Notice the continuous time transform is based on an integration of the input signal, $x(t)$, multiplied by an exponential function. The discrete version, the DFT, is based on a summation of the discrete time input signal, $x(n)$, again multiplied by an exponential.

$$X(w) = \sum_{n=-\infty}^{+\infty} x(n) e^{-jwn} \quad (2)$$

The above equation gives you the frequency components of the sampled input signal $x(n)$. Equation 2 is an infinite summation. For a sampled input sequence of finite length, the DFT equation is reduced to

$$X[k] = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \quad 0 \leq k \leq N-1 \quad (3)$$

Often, a simple substitution is used in Equation 3 to simplify the notation. The complex exponential term is replaced by

$$W_N = e^{-j2\pi/N} \quad (4)$$

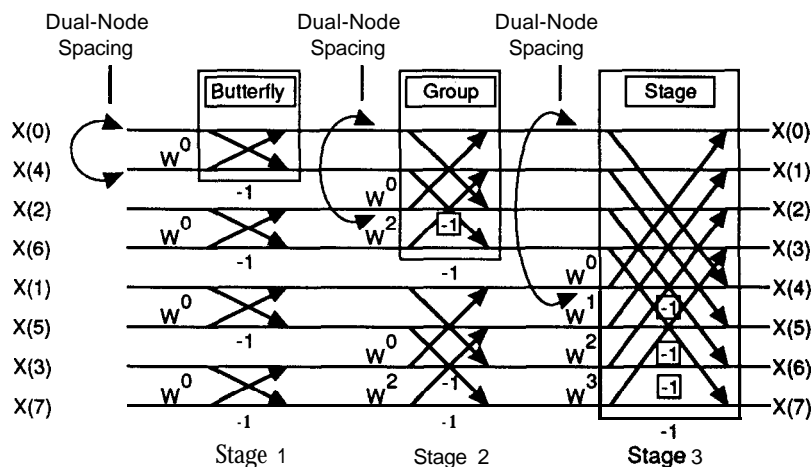
This equation yields the following DFT summation for an N-length sequence:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad 0 \leq k \leq N-1 \quad (5)$$

where $X[k]$ is a sequence of frequencies representing the frequency content of the sampled signal $x(n)$. This equation is the one that you will need to implement in the spectrum analyzer. Its direct computation will lead to a large number of calculations. For an N-length sequence, the above summation would require N^2 complex multiplications. If a complex multiplica-

tion can be thought of as four real-valued multiplications, then the number of multiplicative operations that the DSP must calculate is $4N^2$. For a 256-point DFT, the number of real-valued multiplications is over 262,000—a large number of calculations indeed. Also, $4N(N-1)$ additions and $2N^2$ evaluations of sine and cosine values are required for the complex exponential.

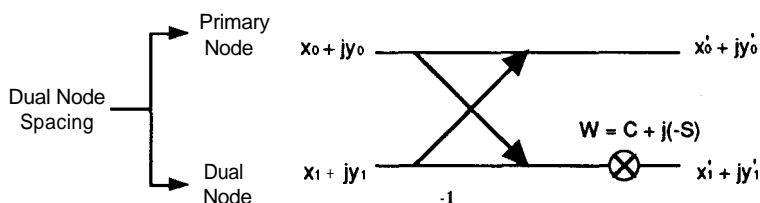
The FFT is an algorithm that efficiently calculates the DFT for a finite length signal. The FFT algorithm employs a divide-and-conquer strategy. By subdividing the work into a number of stages, the total number of computation operations are greatly reduced. For example, the following shows an 8-point DIT FFT flowgraph:



Notice that each stage divides the DFT into two smaller DFTs. The first stage divides the 8-point DFT into two 4-point DFTs. Each 4-point DFT can be subsequently divided into two 2-point DFTs.

The FFT divide-and-conquer strategy reduces the total number of calculations greatly. The total number of multiplications is reduced from $4N^2$ multiplications in the direct calculation to $N/2 \log_2 N$. For a 256-point FFT, this strategy reduces the number of multiplications by a factor of 16!

The fundamental operation of the FFT is the butterfly, which is truly a 2-point DFT, and is shown below. Each stage in the FFT algorithm is $N/2$ butterflies.



The divide-and-conquer strategy is not limited to dividing the algorithm in half for each stage. The Radix-2 FFT algorithm, described earlier, divides the DFT into two half-size DFTs with each stage. A Radix-4 algorithm would divide each stage into four quarter-size DFTs, and is the one I chose to implement on the ADSP-2101 for the voice-band spectrum analyzer.

INTRODUCING THE *New*

Cortex-I

\$749⁰⁰

Video Frame Grabber

The Cortex-II offers the ultimate in performance, efficiency, versatility, and affordability.

Featuring..

- Overlay on Live Video
- ◆ Auto Detect RS-170/CCIR
- ◆ Real Time Capture
- ◆ Prog. Offset and Gain
- ◆ Field or Frame Grab
- ◆ Input and Output LUTs
- ◆ Auto-Page Incrementing
- ◆ Optional Extended Memory Mapping
- ◆ VGA Display (False Color)
- ◆ Half Slot (ISA)
- ◆ OEM Software Keys

Performance..

- ◆ 1024(H) x 512(V) or
- ◆ 640(H) x 512(V) or
- ◆ 512(H) x 512(V) x 2
- ◆ 256 Grey Levels (8 Bits)
- ◆ 44 db S/N Ratio
- ◆ < 10 ns Pixel Jitter
- ◆ 16 Bit Data Bus Access

Input/Output..

- ◆ RS-170/CCIR, Input/Output
- ◆ Dual Video Input
- ◆ External Trigger
- ◆ VGA Connector (15 Pin)
- ◆ VGA Feature Connector

Includes..

- ◆ "C" Library w/ Source
- ◆ Image Capture Utility
- ◆ Ram Disk Emulator

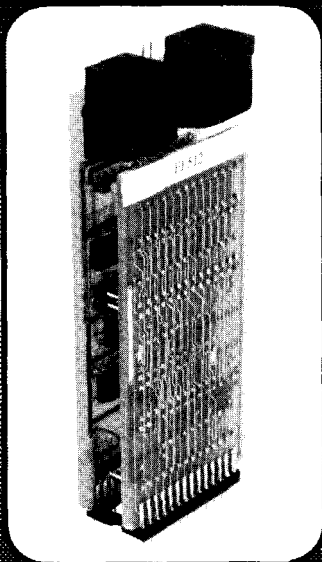
OEM PRICING AVAILABLE

IMAGINATION CORP.

P.O. BOX 84568
Vancouver, WA 98684

PH/FX (206) 944-9 13 1

EPROM EMULATORS



\$99

AND UP

- Downloads through standard PC compatible printer port
- Includes RJ-11/12 adaptor and 6' modular cable
- Generates RESET and /RESET
- Accepts Binary, Intel HEX, Intel extended HEX and 'S' files
- Unique Vertical design minimizes mechanical interference and eliminates the possibility of noise, cross-talk or transmission-line effects from the use of a cable
- Multiple EE512s or EE1Ms can be daisy-chained for 16/24/32 bit target systems

EE256 (emulates 2764-27256) \$99
 EE512 (emulates 2764-27512) \$129
 EE1M (emulates 2764-27010) \$179

10 day MONEY-BACK GUARANTEE
 90 day repair/replace warranty

Technical Solutions

PO BOX 462101
 Garland, TX 75046-2101

Voice or FAX: (214) 272-9392

Call or FAX for ENGINEERING SPECS.

ADSP-210 1 is capable of addressing two operands from two memory spaces, which means the ADSP-2 10 1 can be kept full of operands for the necessary computations.

A program sequencer in the ADSP-2101 fetches the program opcodes for execution. One key feature of the program sequencer is its capability for executing **zero-overhead** loops, meaning the ADSP-2101 can execute instructions in a tight loop without needing any added overhead to maintain the loop [checking for loop completion or exit conditions]. This feature is very important to the FFT program because the FFT consists of three nested loops that execute many times. With zero-overhead looping, the FFT executes very fast and occupies a small amount of memory.

Because the ADSP-2101 is a DSP microcomputer, it contains both internal data memory (1K words) and internal program memory (2K). The entire spectrum analyzer can execute from the internal memories. The EPROM contains the program, but it is downloaded automatically on power up into the DSP, and it then executes entirely from within the DSP. This EPROM/DSP combination has the benefits of an easily modified program memory, nonvolatility, and the speed of operation from internal memory with no external RAM or circuitry required.

I chose to implement a Radix-4 FFT on the ADSP-210 1, and limited it to 256 points in order to maintain all calculations within the real-time constraint of the sampling period. The ADSP-28MSP02 samples the speech at 8-kilosamples per second. The FFT collects 256 data points before processing them, so the algorithm must execute in less than 256 points x 125 μ s, or 32 ms, to maintain real time.

From the eight-point FFT flow-graph shown in the sidebar, you can see the three fundamental parts of the FFT. Each of these parts is a program loop that executes a number of times. The stage loop is outermost, the group loop is in the middle, and the butterfly loop is on the inside.

Besides the FFT, the ADSP-2101 processor must read data values from

the ADSP-28MSP02's ADC, write the frequency values to the ADSP-28MSP02 DAC, and perform a windowing function on the incoming data. All of these functions are performed on the 2101.

The program executes in 945 μ s, corresponding to 9450 100-ns cycles. The program memory is 1034 words, with just 379 words of actual code and the rest data for the program; the data memory for the digitized analog input is 1024 words.

HARDWARE IMPLEMENTATION

Implementing this system in hardware requires just six components. On reset, the ADSP-2101 boots its internal memory from the 27C5 12 EPROM. Eight of the ADSP-2101 data lines (D8-D15) are connected to the EPROM data lines (DO-D7). The address lines (AO-A13) provide the least-significant address bits to the EPROM. The two MSBs of the EPROM address are provided to the ADSP-2101 from the uppermost data lines of the ADSP-2101 (D22 and D23). The native address space of the ADSP-2101 is 16K locations, or 14 bits.

The ADSP-2101 features a boot **address generator**, which uses the uppermost data bits during a boot sequence to increase the address reach at boot time. This feature enables the ADSP-2101 to boot up to eight separate programs under software control. The ADSP-2101 always boots in page 0 at reset, but at any time after that, the ADSP-2101 can boot any other page under software control. This particular implementation uses only one boot page, so a 27C64 can be used in place of the 27C5 12.

A microprocessor-grade crystal is used to clock the ADSP-2101. With a frequency of 10 MHz, the parallel resonant fundamental frequency crystal clocks the ADSP-2 10 1 with an instruction rate of 100 ns. All instructions can execute in a single 100-ns instruction cycle from this crystal. The ADSP-28MSP02 requires the use of a 13-MHz oscillator.

The ADSP-28MSP02 is a linear codec designed for voice-band applications. It has a sampling rate of 8 kHz and provides 65 dB of SNR+THD. This

device contains low- and high-pass digital noise-shaping filters that limit the bandwidth from DC to 3400 Hz. It interfaces to the ADSP-2101 through a serial port. Six signals are required to interface between the DSP and the linear codec. The ADSP-28MSP02 provides a serial clock signal to the ADSP-2101. The serial clock frequency is 2.6 MHz, and bits are transferred to and from the ADSP-2101 at this rate.

There are two serial data lines on the ADSP-28MSP02. Serial Data Out (SDO) provides 16-bit data from the ADC and is connected to Data Receive

(DR) on the ADSP-2101. The second line is Serial Data In (SDI) and is used by the DSP to send data to the DAC.

The data signals provide the bits for each 16-bit word transferred between the ADSP-2101 and the ADSP-28MSP02. Two framing signals show the start of the transmit word (SDOFS) and the start of the receive word (SDIFS). These signals are connected to the ADSP-2101's RFS and TFS (Receive and Transmit Frame Synchronization), respectively. The sixth pin on the ADSP-28MSP02 serial interface is DATA/*CTRL. The ADSP-

28MSP02 has a control register that must be written to prior to the operation of the ADC and DAC.

The DATA/*CTRL signal signifies when the word written to the codec is a control word for the control register or data for conversion. This signal is latched by the flip-flop on the board. Whenever an external data memory write occurs, the least-significant bit is latched by the flip-flop. Writing a logic 1 to the external flip-flop causes the DATA/*CTRL signal to be logically high, placing the codec in data mode. Similarly, when a logic 0 is written out to external data memory, the codec is set into control mode. The next serial word received will be written into the control register.

SUMMARY

A hand-held, battery-operated voice-band spectrum analyzer can be easily constructed based on a DSP processor like the ADSP-2101. The simplicity of the circuit is due to the very high level of integration of the devices used. The processor uses its internal memories for both program and data storage, so no external memory is needed other than a bootstrap EPROM. The converter used incorporates both A/D and D/A conversion. □

Gerald McGuire is the DSP *Applications Engineering Manager* at Analog Devices and has been with the group for over five years. He holds an MS. in Electrical Engineering from the University of Vermont where he specialized in digital signal processing.

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

Supports standard DOS 3.31 interrupt functions

Built-in real-time multitasking microkernel

low royalties (\$6/copy and down)

Supports PharLap DOS extenders

The #1 Brand of DOS for Embedded Systems

GENERAL SOFTWARE EMBEDDED DOS™

Discover the DOS that hundreds of 80x86-based embedded systems developers are using to simplify their product development- Embedded DOS, the only DOS designed to run real-time embedded applications like motion controllers, high speed data acquisition, and satellite control systems.

Use Embedded DOS to leverage your DOS experience. You can run it with your code on a PC, then boot it on the target from ROM, solid-state disk, floppy, or hard disk. Even take luxurious advantage of its full reentrancy and high-performance multitasking while developing with your dependable DOS tools.

For a free bootable demo disk and an architectural tour of Embedded DOS, plus information on how to start using DOS in your next embedded application, call (206) 391-4285 and ask for Dept C67.



GENERAL
SOFTWARE,

P.O. Box 2571 Redmond, WA 98073

Tel. (206) 391-4285

Fax. (206) 746-4655

The DOS Experts

Copyright (C) 1992 General Software, Inc. General Software, the GS logo, and Embedded DOS are trademarks of General Software. Other marks property of their owners.

Shaping the World of Sound

FEATURE ARTICLE

Steven Avritch

a

s long as there are audio signals present in electronic designs, there will be a

need for circuits to shape them into the desired waveforms. Traditionally, audio waveforms have been shaped with analog components, such as resistors, capacitors, and op-amps. Unfortunately, a circuit's flexibility is limited by the hardware components of the original design. Another significant problem with analog designs is the considerable growth of their cost and complexity (due namely to component counts) as the complexity of the design increases.

With the advent of low-cost, high-performance DSP devices, digital signal processing designs now achieve a level of complexity and flexibility unparalleled in the analog world while

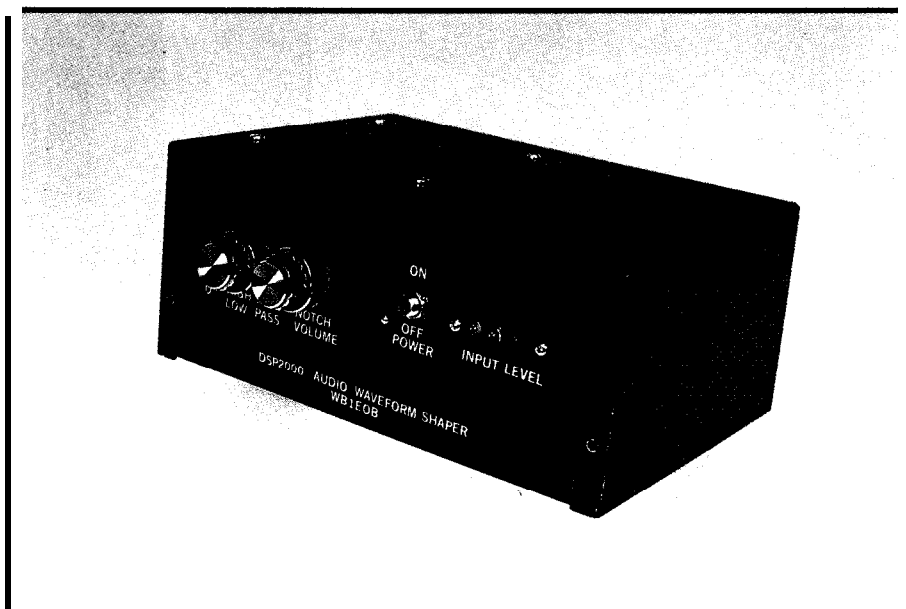
remaining economically competitive. The DSP designs require only a fraction of the number of components compared to their analog counterparts, simplifying assembly and increasing reliability.

DSP BASICS

As with most things in this world, electronic signals and the circuits they run through can be modeled with mathematical equations. A DSP design first converts the analog input signal into digital ones and zeros. The processor then performs the appropriate math to emulate the desired function(s) and converts the digital results back to analog.

While the complete theory governing digital signal processing would quickly fill a good-sized library, the basic rules and equations are fairly simple and easy to understand. The primary rule governing all digital signal processing states is that the sampling rate (i.e., the A/D conversion rate, see Figure 1) of the signal and the associated digital processing must be performed at a minimum of twice the highest frequency of interest, known as the Nyquist frequency. So if you want to filter an audio signal digitally where the highest frequency of interest in the audio signal is 3 kHz, then you must sample and calculate the filter equations at a minimum of 6 kHz. In

Have you ever tried to pick voice out of the static when listening to a radio transmission? The DSP2000 Audio Waveform Shaper was built for just such a purpose and succeeds quite nicely.



The DSP2000 Audio Waveform Shaper allows quick adjustment of the incoming audio signal. Cut-off frequencies for a low-pass, high-pass, and notch filter are set using the front-panel knobs. A digital signal processor is the core of the design, keeping parts count to a minimum.

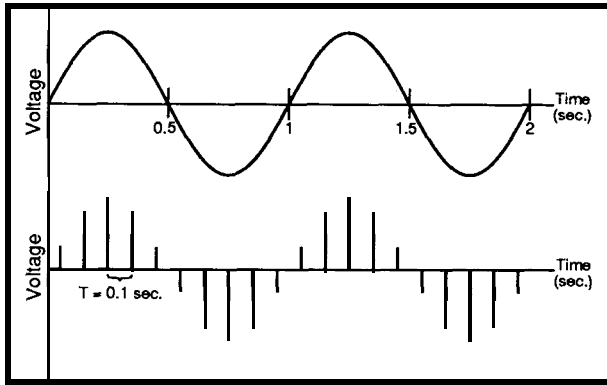


Figure 1—The continuous 1-Hz sine wave at the top is sampled every tenth of a second by the ADC. The sample rate is the inverse of the sample period, which in this case is 10 Hz.

reality, most designs will sample the input at five to ten times the highest frequency of interest in order to minimize the effects of noise induced by digitization, or **quantization noise**. This process is called **oversampling** and has the effect of increasing the signal-to-noise ratio of the digital filters. Figure 1 illustrates how an analog signal is digitized by the ADC.

Theory also dictates that frequencies present at the input to the ADC must not exceed the highest frequency of interest, otherwise aliasing (another form of induced noise) will occur. Analog antialiasing low-pass filters are normally placed in front of the ADC to remove any unwanted high-frequency components from the input signal.

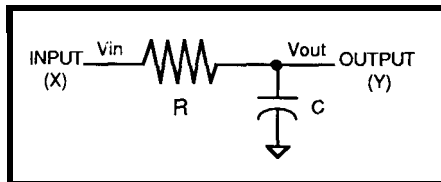


Figure 2—The most basic low-pass filter is made of a resistor and a capacitor.

FILTER EQUATIONS

The frequency response of an analog filter is usually described by equations called **transfer functions**. Figure 2 shows an example of a simple low-pass filter. Its associated transfer function is

$$H(s) = \frac{\text{output}}{\text{input}} = \frac{1}{1 + sRC}$$

$$\begin{aligned} \text{where } f_c &= \frac{1}{2\pi RC} \\ s &= j\omega = j2\pi f_c \\ j &= \sqrt{-1} \end{aligned}$$

The transfer function $H(s)$ is a frequency-based equation and cannot be easily implemented in a processor. However, this function can be converted into a **time-based** equation by replacing the s in the equation with $2(z-1)/T(z+1)$. After some algebra, you arrive at $H(z)$. $H(z)$ is the **time-based** equivalent of the

frequency-based equation and is easily implemented in software when written in the recursive form:

$$\begin{aligned} H(z) &= \frac{\text{Filter Output (Y)}}{\text{Filter Input (X)}} = \frac{V_{\text{out}}}{V_{\text{in}}} \\ &= \frac{K(z+1)}{(z-B_1)} = \frac{K(1+z^{-1})}{1-B_1z^{-1}} \\ Y_n - B_1Y_{n-1} &= K(X_n + X_{n-1}) \\ Y_n - K(X_n + X_{n-1}) + B_1Y_{n-1} &= 0 \end{aligned}$$

where X_n = Filter input this cycle
 X_{n-1} = Filter input last cycle
 Y_n = Filter output this cycle
 Y_{n-1} = Filter output last cycle

Figure 3 shows the generic second-order (two-pole) transfer functions and their equivalent digital representations for the low-pass, high-pass, and notch filters implemented in the DSP2000.

DSP2000 AUDIO WAVEFORM SHAPER

The DSP2000 Audio Waveform Shaper is based on the Texas Instruments TMS320C10 digital signal processor. The TMS320C10 is similar to normal processors like an 8088 except the DSP's internal architecture is optimized for mathematical calculations. The software written for the DSP2000 implements three digital audio filters: a second-order high-pass, a second-order low-pass, and a second-order notch filter. The cutoff frequency for each filter is independently variable from 100 Hz to 3 kHz (in 100-Hz increments). The three filters implemented in this software package are just a sample of the types of functions

that can be implemented in the DSP2000. For example, the software could be rewritten to implement other functions such as a speech scrambler and unscrambler or a Touch Tone generator and decoder, to name a couple.

HARDWARE DESIGN

A detailed schematic of the DSP2000 is shown in Figure 4. To begin, any DC components of the audio input are removed by coupling capacitor C5. The input is then biased by U8a from its input value of ± 2.5 volts up to between 0 and 5 volts, which is compatible with the ADC input. The biased signal is passed through a 3-kHz antialiasing low-pass filter (U8b) and then into the ADC. A 5.1-volt zener diode (D3) limits the input to the ADC to 5.1 volts, which protects the ADC from damage if the input audio is too strong.

The ADC (U4) is an 8-bit, four-channel, unipolar unit. The A/D conversion time is 40 μ s, which allows a sampling rate of approximately 20 kHz (about six times oversampling for the 3-kHz frequency of interest). Potentiometers are connected to the remaining three ADC channels and are

Generic second-order transfer function:

$$H(s) = \frac{\text{Filter output (Y)}}{\text{Filter input (X)}} = \frac{C_2 s^2 + C_1 s + C_0}{D_2 s^2 + D_1 s + D_0}$$

$$\text{Low-pass } H(s) = \frac{1}{s^2 + \frac{s}{2\pi F_c} + 4\pi^2 F_c^2}$$

$$\text{High-pass } H(s) = \frac{s^2}{s^2 + \frac{s}{2\pi F_c} + 4\pi^2 F_c^2}$$

$$\text{Notch } H(s) = \frac{s^2 + 4\xi_1\pi F_c s + 4\pi^2 F_c^2}{s^2 + 4\xi_2\pi F_c s + 4\pi^2 F_c^2}$$

$$\text{where } \xi_1 < \frac{1}{10} \leq \xi_2 < 1$$

Digital form of $H(s)$:

$$Y_n = K(X_n + A_1 X_{n-1} + A_2 X_{n-2} - B_1 Y_{n-1} - B_2 Y_{n-2})$$

where A, B, and K are constants

Figure 3—Transfer functions for low-pass, high-pass, and notch filters are used to implement the filters with a digital signal processor.

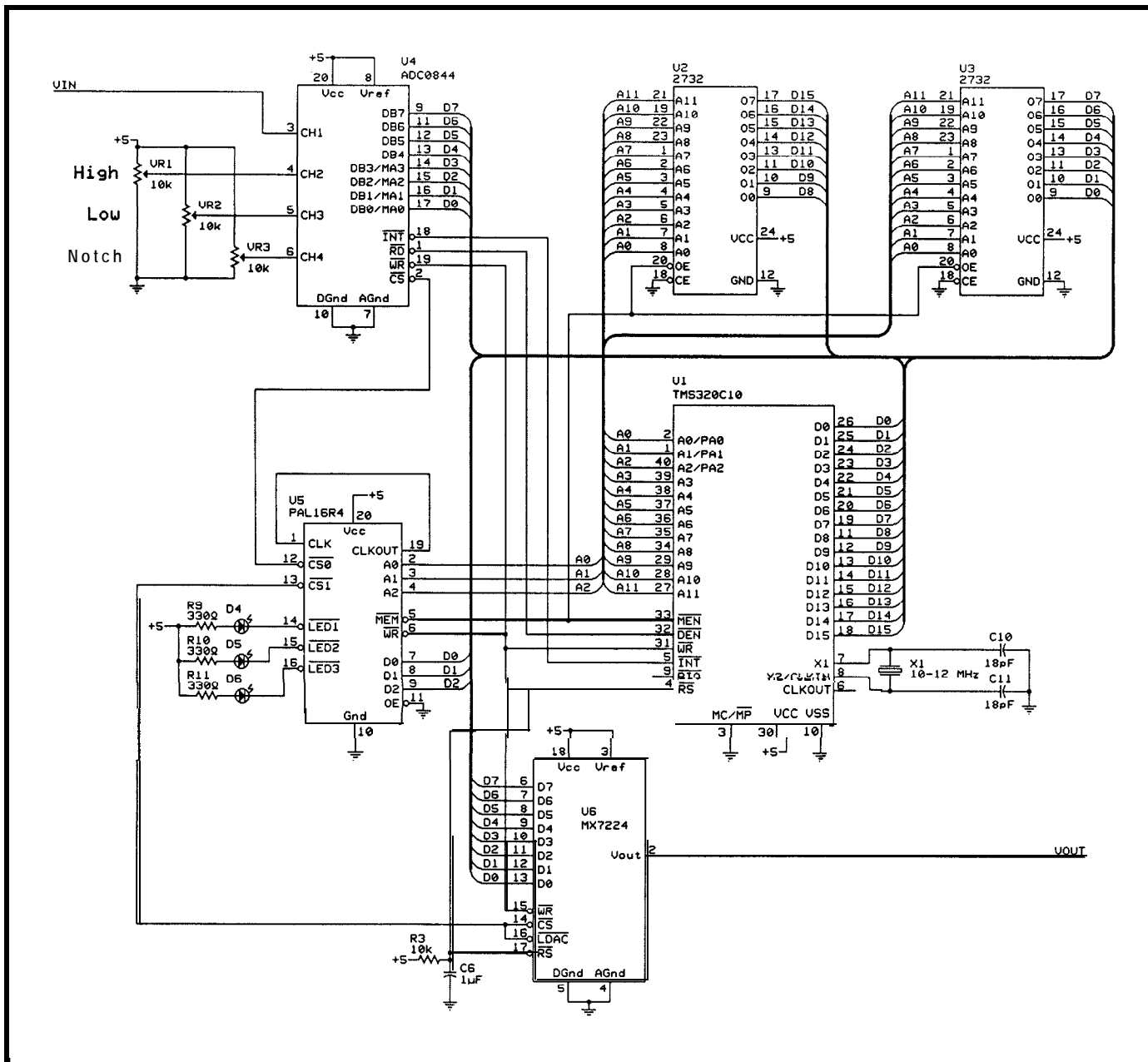


Figure 4a—The DSP2000 uses a TMS320C10 DSP chip at its core (U1). Surrounding the processor are the ADC (U4), DAC (U6), memory (U2 & U3), and a PAL used for decoding and latching.

used for varying the cutoff frequencies of the three digital filters.

The digitized audio input is processed by the TMS320C10 processor (U1). The TMS320C10 is running at 10 MHz, which yields an internal cycle time of 400 ns ($4/F_{clk}$), so requires memory devices with an access time of 350 ns or less in this application.

The digital output is converted back to analog by a MAX7224 DAC (U6) and is smoothed by a low-pass reconstruction filter (U7a). Finally, the output signal is amplified by the combination of U7b and Q1 and sent

to the speaker through coupling capacitor C7. The volume of the output audio is controlled by VR4.

U5 is a 16R4 PAL, and is used to generate the chip selects for the ADC and DAC. U5 also implements a 3-bit output latch to drive the three status LEDs (D4, D5, and D6). Listing 1 gives the PAL equations.

The DSP2000 requires ± 12 volts and +5 volts. All three voltages are derived from a single AC wall-adaptor input. The ± 12 -volt supplies are generated by D1 and D2 along with filter capacitors C8 and C9. A 7805

voltage regulator (U9) generates +5 volts from the +12 volts.

Due to high-frequency signals present in the design, having decoupling capacitors on all the ICs is a good design practice, though they aren't shown on the schematic to save space. The decoupling capacitors should be mounted as close as possible to the power pins on each IC.

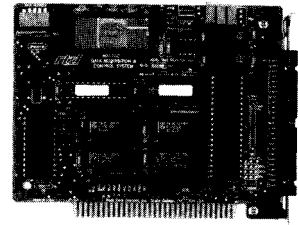
SOFTWARE DESIGN

The DSP2000's software implements three variable filters: a low-pass, a high-pass, and a notch. The software

Real Time Devices

"Accessing the Analog World"

Exciting New Products!



AD1200 - \$359

- 12-bit 125 kHz A/D Conversion
- 16 Analog Input Channels
- 3 Timers and 16 DIO Lines
- Supports DMA & Pacer Clock
- AT Bus Version Available!

for **AMPRO CPUS**

DM200 12-bit 40 kHz 8 channel analog input board; 8254 timer and 16 digital I/O lines \$295

DM406 12-bit 100 kHz 16 channel analog I/O board; 2 D/A outputs; 8254 timer; 16 digital I/O lines; Supports DMA and pacer clock. \$449

DM806 High current digital I/O board; 8254 timer; opto-22 compatible **\$195**

PC/XT/AT Boards

AD2700 16 channel 12-bit 150 kHz analog input board; Supports DMA, pacer clock and programmable gain; 8254 timer; 16 digital I/O lines; 16-bit AT bus operation. \$525

TC48 Dual Am951 3 Timer/Counters; 24 digital I/O lines; 16-bit AT bus operation with expanded interrupts \$398

O1024 High current digital I/O board; 24 buffered DIO lines; 8254 timer; Compatible with opto-22 equipment **\$195**

AD3110 Super fast 16 channel A/D board! 12-bit 200 kHz A/D rate; Burst mode operation; On board FIFO memory; Programmable gain; Timers; DIO lines; Supports DMA and pacer clock; Optional D/A outputs \$665

AD3710 Low cost version of AD3110 with on board FIFO memory and 200 kHz burst mode operation \$525

DA810 8 channel 12-bit D/A; Voltage or current loop output; 8254 timer; 24 digital I/O lines. \$589

MR16 110 VAC mechanical relay expansion board; Computer control of 16 AC loads. **\$215**

OP16 Optoisolated 16 channel digital input expansion board. \$225

TS16 16 channel thermocouple expansion board; Supports J & K types. \$298

Unparalleled choices for real world control. Over 50 hardware and software products for single and OEM users.

FREE 80 PAGE CATALOG!

Real Time Devices, Inc.
State College, PA USA



Tel.: 814/234-8087
FAX: 814/234-5218

#117

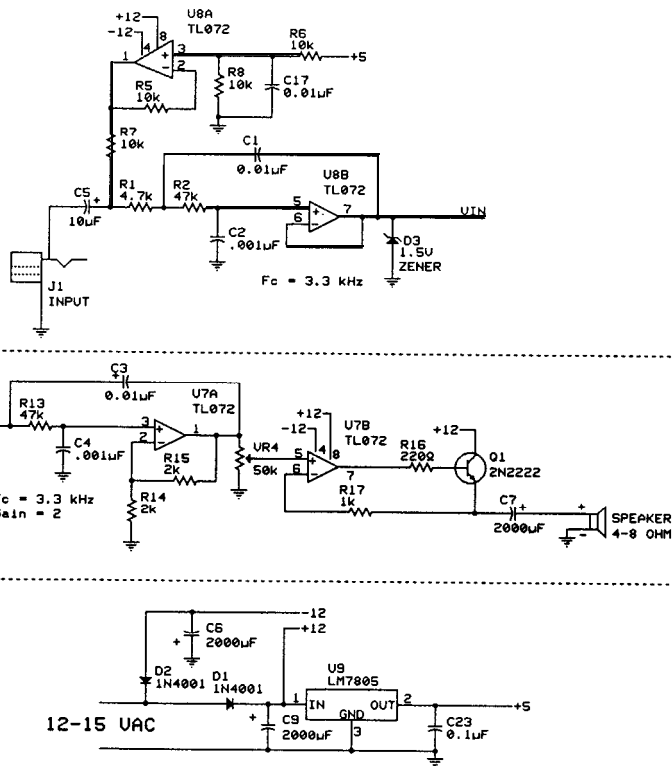


Figure 4b—Audio coming into the DSP2000 passes through a simple low-pass filter before going to the ADC. The signal from the DAC passes through another low-pass filter before being amplified and sent to the speaker. The unit may be powered off any AC power supply ranging from 12 to 15 volts.

Listing 1—The 16R4 PAL not only provides I/O decoding but also latches three status LEDs.

MODULE DSPU5_16R4
TITLE 'DSP CHIP SELECT';

```

U5_16R4      DEVICE 'P16R4';
CK           PIN 1;
A0,A1,A2     PIN 2,3,4;
MEMB        PIN 5;
WRB         PIN 6;
D0,D1,D2     PIN 7,8,9;
Q0,Q1,Q2     PIN 14,15,16;
CSOB        PIN 12;
CS1B        PIN 13;
CLKOUT       PIN 19;
H.L.X.CLK    =1,0,.X...K.;
A            =[A2,A1,A0];
D            =[D2,D1,D0];
Q            =[Q2,Q1,Q0];
    
```

EQUATIONS

```

CSOB  = A0#A1#A2#!MEMB;
CS1B  = !A0#A1#A2#!MEMB;
CLKOUT = A0#!A1#A2#!MEMB#WRB;
    
```

```

!Q0 := !D0;
!Q1 := !D1;
!Q2 := !D2;
    
```

ENDDSPU5_16R4;

also performs peak level detection on the audio input. The software continuously samples the audio input, performs the three digital filter calculations, and then outputs the result to the DAC. The peak level detector determines the peaks of the input audio and updates three status LEDs, which are used to adjust the volume of the audio input.

The software also samples the three "cutoff frequency" knobs (VR1, VR2, and VR3) in order to update the filter constants [i.e., the filter cutoff frequencies) when the user varies the filter frequencies. While the audio input is sampled at close to 20 kHz, the knobs are sampled at only 10 Hz. Sampling the knobs at 10 Hz is more than adequate for adjusting the frequency cutoffs and leaves more time for sampling the audio input. A block diagram of the software operation is given in Figure 5.

OPERATING THE DSP2000

Operating the DSP2000 is very simple. To begin, set up the DSP2000

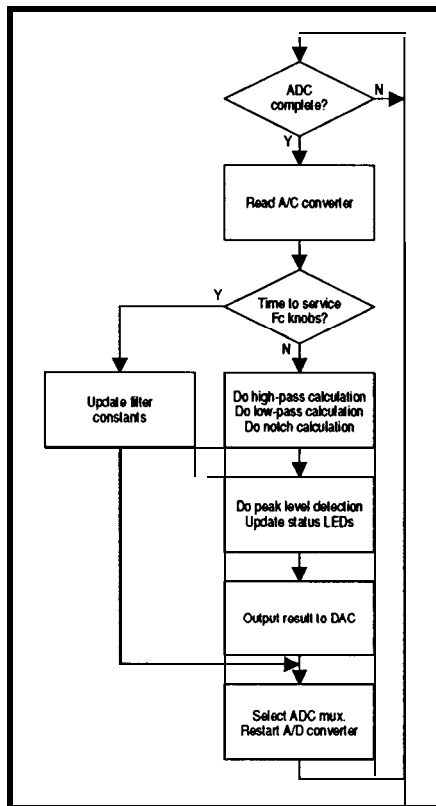



Figure 5-The software follows a basic loop that continuously processes incoming audio while periodically checking the adjustment knobs.

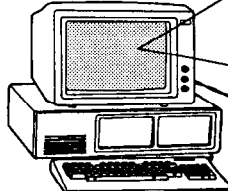
for a flat frequency response by adjusting the low-pass filter for maximum frequency (fully clockwise), the high-pass filter for minimum frequency (fully counter-clockwise), and the notch filter for midband. Next, adjust the DSP2000's volume knob to midscale and connect the audio source to the input J1. You can now turn on the unit.

Adjust the amplitude of the input audio (via the volume knob on the audio source) until the green and yellow LEDs light steadily and the red LED flickers occasionally. The three LEDs represent the level of the audio at the input to the ADC: the green LED represents quarter scale (1.25-volt peaks), the yellow LED represents half scale (2.5-volt peaks), and the red LED represents full scale (5+-volt peaks). The best signal-to-noise ratio is attained when the analog input is as close to full scale as possible without saturating the ADC. Now adjust the DSP2000's volume knob to the desired level and you're ready to start experimenting with the filter cutoff frequen-




EXOR μPLC
PROGRAMMING MODULE

LADDER LOGIC
RUN-TIME ENGINE



CPU's : 8051 Z80 8086



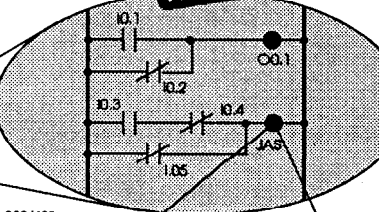
ANALOG I/O
DIGITAL I/O

YOUR EMBEDDED CONTROL SYSTEM

μPLC

LADDER LANGUAGE

AVAILABLE IN SOURCE CODE



JAS :
Call subroutines written
in your favorite language.
C, ASSEMBLER...

EXOR

ELECTRONIC R&D

The basic building block for your control system !

The μPLC operating system kernel transforms your basic microprocessor into a high-performance Programmable Logic Control.

Just plug the μPLC ROM into your favorite microprocessor card, load the integrated programmer/debugger onto your PC, connect a serial cable and begin taking the credit for a job well done!

Get started for only \$ 399

Eval. Kit \$ 99

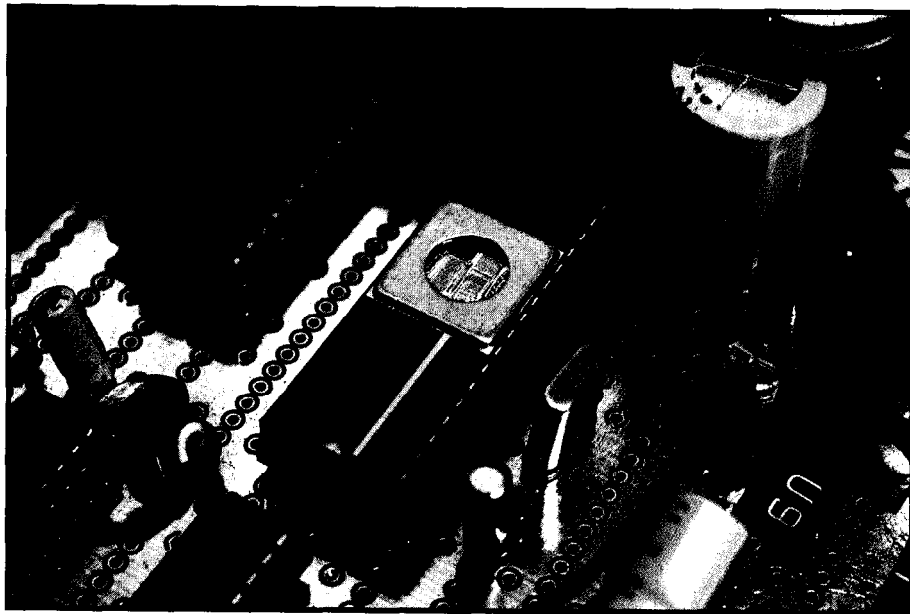
- RS 485 multidrop distributed system
- Easy porting to custom hardware
- ROMable Run-Time Engine
- Programming tools
- On-line debugging
- Target library

Eval. Kit contains :

- Parallel Port I/O or On-Screen Simulator
- Programming & Debugging modules
- PC TSR Target

Excellent for ladder language training

FAX (513) 874-3684 CALL (513) 874-4665
4850 Interstate Dr. Cincinnati, OH 45246




The DSP2000, based on the TMS320C10, is built on a prototype board that has an integral ground plane to minimize electrical noise on the board.

cies of the three filters by varying VR1, VR2, and VR3.

I use the prototype DSP2000 to filter the audio output of my ham radio receiver. The primary frequencies of interest in voice communications don't exceed 3 kHz, so the DSP2000 works perfectly in this application.

GOING FURTHER

I've presented here a very basic DSP design as an introduction to the world of digital signal processing. While the DSP2000 works well in its intended application, certain design modifications can significantly improve its performance. For example, you can use faster DSP and memory chips to increase the sample rate, which increases frequency response. Also, the resolutions of the ADC and DAC could be increased to 10 or 12 bits to increase the dynamic range. However, no matter how sophisticated the unit becomes, it will always resemble the basic design presented here. 

Steven Avritch holds a B.S. in Electrical Engineering from the University of Connecticut. He is currently employed as a lead test flight systems engineer at Hamilton Standard, Division of United Technologies. He is also an amateur radio operator.

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnectTime" in this issue for downloading and ordering information.

SOURCE

The following items are available from:

Simple Design Implementations
P.O. Box 9303
Forestville, CT 06011-9303
(203) 582-8526

1. Complete kit of parts. Includes all components, ICs, and wire-wrap sockets. Memory chips are preprogrammed. Does not include enclosure or PC board. \$99
2. IC kit. Includes all ICs and wire-wrap sockets. Memory chips are preprogrammed. \$59

Prices include shipping. Connecticut residents, please add appropriate sales tax.

I R S

- 407 Very Useful
- 408 Moderately Useful
- 409 Not Useful

FAST COMPLETE ACCURATE DRAM TEST DIPs - SIMMs - SIPs



RAMSTAR

Ins. RESOLUTION

ACCESS SPEED VERIFICATION	
80 ns. thru 180 ns. (Std.)	\$249.00
45 ns. thru 110 ns. (Fast)	\$349.00
4MEG Option	Add \$ 89.00

AUTO-LOOP
Continuous Test 6.25 MBits/sec.

ADAPTERS:

SIMM/SIP ADAPTER	\$189.00
Tests 64K, 256K, 1M & 4M Devices 8 or 9 Bit versions.	

NEW

'NTX ADAPTER	\$149.00
Tests 64 Pin Dual-Edge Laser-writer Type SIMM's	

4 X ADAPTER	\$ 89.00
Tests 64K & 256K By 4 Bit Devices	

AC ADAPTER	5 18.00
Regulated +5V @ 1 Amp.	

FREE RAMFACTS DRAM NEWSLETTER

1-800-RAMSTAR

COMPUTERDOCTORS
9204-B Baltimore Boulevard
College Park, Maryland 20740

MADE IN U.S.A. U.S. PATENT No. 4,965,799

#119

The Dawning of the Light Transistor

An Optical Computer Method Using Interference Fringe Component Regions

That super '586 system sitting on your desk could soon be replaced by a hologram. Find out about the world's first photonic transistor and what it could mean to computing as we know it.

FEATURE ARTICLE

John N. Hait

Forty years ago, vacuum tube computers produced more heat than work and were slower than a sixth grader on a slide rule. Yet they laid the ground work for the invention that topped off the 1940s: AT&T's tiny transistor. Who could have visualized the revolution that was about to take place? Now that we are into the '90s, maybe rethinking certain assumptions is in order. I'd like to describe the Photonic Transistor (patent #5,093,802), a project that I feel may be a very important piece of technology.

What is a "photonic transistor"? It is a transistor that uses light instead of electricity. "Oh, solar?" No! I said light instead of electricity! "Ah! It must be one of those Self-electro-optic Effect Devices reported in the press recently...right?" Nope! No *electro* anything, just light. No electrons at all. Photons, the basic substance of light, do the work in photonic transistors, not electrons.

Why convert to light? Won't electronic performance just continue improving? No, again. Technology is reaching the end of its rope with electrons. Given the newest methods of atomic-scale manufacturing, the basic physics of the electron places restrictions on its speed and functionality within a semiconductor. Yet the demand for increased computing power grows daily.

Photons are faster than electrons and can carry more information easier, which is why phone companies are switching from copper wire to optical fiber. However, no one has built a practical device that makes one light

beam switch another light beam on and off, a process similar to the one used by electrons in a conventional transistor—that is, until now. So, why do photons work better than electrons, and how do photonic transistors cure the problem?

WHAT'S THE PROBLEM WITH ELECTRONS?

Faster! Less money! Unfortunately this trend is true for computers and not for sports cars. An ever-growing demand for faster, yet economical communication exists. However, trying to process reams of information in a supercomputer is like trying to get thousands of commuters to work on time. The faster the drivers get to work, the quicker the work gets done.

Electrons inside a computer chip are not like race cars zipping around Daytona Speedway. They're more akin to traffic on a Los Angeles freeway—backed up, bunched up, and bogged down. While a large number of cars do make it through the maze of rush hour traffic, the individual driver takes a considerable amount of time getting to work. Continuing this analogy, an electronic transistor shortens a driver's commute by using a form of mass transit. When a bit of information is shoved into one end of a wire, the original electrons carrying this information are not the same ones that deliver it at the other end. Rather, the electrons smack into each other, one after the other, until some of them get shoved out the other end. However, even at today's clock speeds, this "bus trip" distorts a signal trying to make it to the other end of a mother board, drowning it in traffic noise.

Unfortunately, using transistors in a computer chip can also be likened to putting stoplights on a freeway. They direct the moving streams of electrons from one intersection to another, but because of inductance, capacitance, and resistance, traffic gets all bunched up behind the red lights. On green, electrons have to wait their turn to accelerate up to speed, only to pile into one another at the next red light.

As a result, chip designers are forced to slow down the traffic flow to maintain some semblance of order

within the whole process. The composite electronic device plods along, having to wait for the slowest parts to catch up with the rest. This bumper-car action between silicon stoplights, makes each electron less like an automobile and more like a horse cart full of lead bricks. It will get the driver to work, but not in any real hurry.

Chip designers are doing everything they can to get the lead out. First they put in lots of traffic lanes, then they put the stoplights closer together. Switching times decreased, current flow and heat dissipation were re-

duced, information processing got a little quicker. So every few years they come out with a new and improved silicon road maze for these electron bumper cars.

to destination. Lanes must be able to crisscross each other simultaneously in the same 3-D space without any degradation in signal quality or crosstalk between channels. It must be able to sort, select, switch, and direct traffic flow instantly. Every channel must be an express lane. Everything must move at the same speed, top speed, all the time. There can't be any slowdowns or pileups.

Squashed into gas-fumed gridlock, each driver sees the stoplight change long before the cars get moving. If all drivers could get to work at the speed

foot. Photonic transistors can be made about as small as electronic ones, so that same bit of information could have been through millions of operations in that same nanosecond. Why? Because the light pulse doesn't bog down at each gate. It can be sent through millions of photonic transistors in the same time as electrons take to wade through only a handful. Thus, computers made with photonic transistors will be able to operate hundreds of thousands of times faster than their electronic counterparts—even in serial.

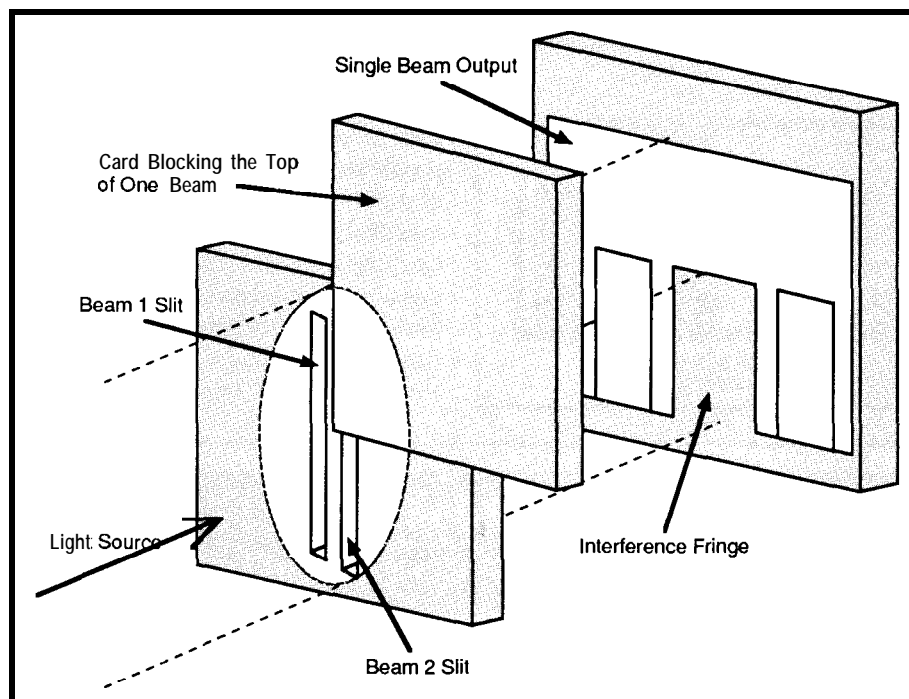


Figure 1—The double light beam interference fringe and the single beam output that produces no fringe display the wave nature of light used to produce transistorlike functions in the photonic transistor.

duced, information processing got a little quicker. So every few years they come out with a new and improved silicon road maze for these electron bumper cars.

The bumper-car effect prevents traffic lanes from being put very close together. As stoplights get closer, fewer and fewer electrons make it through. Soon traffic at one light is slopping back into the light before it. Thus, the inherent construction of electrons makes them less than ideal information carriers.

The ideal information carrier would have to be free from inductance, capacitance, and resistance. It must move rapidly and directly from source

of light, "rush hour" would become "rush second"! Replacing electrons with photons means megaFLOPS would become teraFLOPS and beyond. Thus, light is superior to electricity; it is the ideal information carrier.

How do these qualities figure into real photonic computers? The best electronic gates can switch a little better than ten times a nanosecond. If they are really tiny and placed extremely close together, the switched signal (but not the exact electrons themselves) may have made it a hair's width through the semiconductor. However, in that same nanosecond, photons carrying that exact same bit of information will have traveled nearly a

MASSIVE PARALLEL ARCHITECTURES

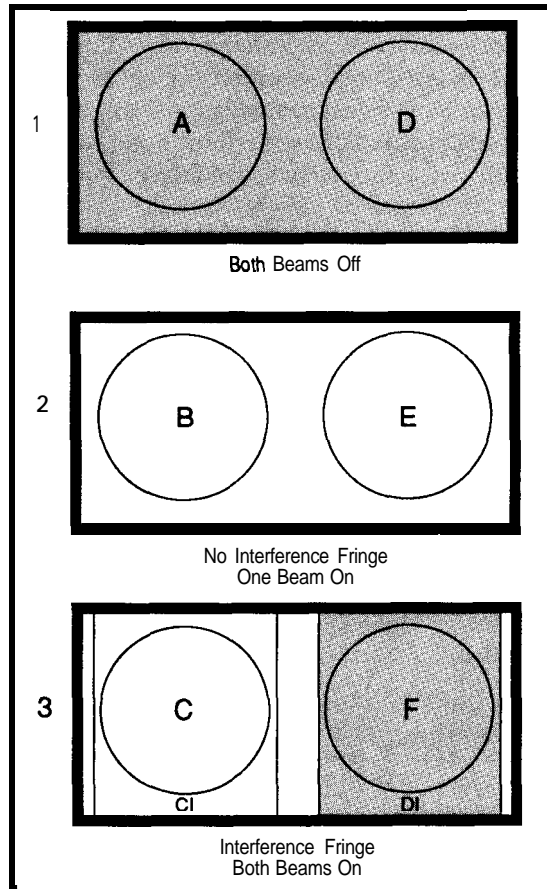
The current trend in computing is to create *parallel architectures*. In this practice, several thousand processors are wired together in order to complete an overall task in a shorter amount of time. When you read this page, you read the words serially, one right after the other. However, an image carried to your eye arrives in a massive parallel fashion. The entire image is there at the same time.

Broken down into individual little pieces, or pixels, that single image becomes millions of individual information-carrying beams of light. In a photonic computer, each beam of light can undergo millions of calculations in a short amount of time and space. That single image represents not just millions of individual pixel beams, but millions of operations performed on millions of beams simultaneously. Because these composite images are continually being modified as computation proceeds within photonic transistors, they are called *dynamic images*. Electronic circuits are simply left in the dust when trying to match such massive parallelism.

THE STUFF THAT MAKES UP PHOTONIC TRANSISTORS

As you might expect, photonic transistors are not whittled out of silicon. Instead they are made out of photographs. Inexpensive photographs. While the many beams could be interconnected using conventional optics, the versatility of the hologram

Figure 2—The three combinations of two input beams produce: (1) Both inputs off, no light output; (2) One beam on, even light distribution throughout the area; (3) Both beams on, light concentrated into the smaller areas of an interference fringe. By separating the hinge component regions into destructive interference (DI) areas and constructive interference (CI) areas, the photonic transistor is able to produce the Boolean OR and XOR functions, signal amplification, and analog signal processing.



makes it an ideal medium for hooking photonic transistors together and interconnecting photonic transistor-produced dynamic images. Holographic interconnection has been a part of the present technology for nearly 20 years. What's needed are functioning photonic transistors to complete the interconnection.

HOW DO THEY WORK?

So how can pictures be made to do the same calculating tasks now done by complex layers of silicon?

In order to be practical, photonic transistors must be simple. They must be easy to design, interconnect, and manufacture. They must be easy to understand. In order to be patented they must be simple enough that people say, "Now, why didn't I think of that?" So, among all of these massively parallel light beams, carrying immense amounts of information at the speed of light, let me focus on one tiny little function of one tiny little photonic transistor and show you how it works.

The Disk is DEAD!

DON'T SELL YOUR CUSTOMERS A PROBLEM!

Dust, moisture and vibration kill rotating disks every day. **Instant Access™ No Motion™** solid state memories offer extremely high reliability, light speed access and the ultimate in data integrity. **Instant Access™** is available in FLASH EEPROM and NVRAM and is programmable in DOS or WINDOWS. Rely on **Instant Access™**—the demonstrably superior No Motion Memory.

"Instant™ Access" is the Solution!

1000 times faster!

100 times more reliable!

Ideal for storing DOS and **programs!**

• Perfect for booting diskless workstations.

• Hard Disk Drive replacements available

Call 1-800-451-DISK to order your evaluation unit

VISA
AMEX

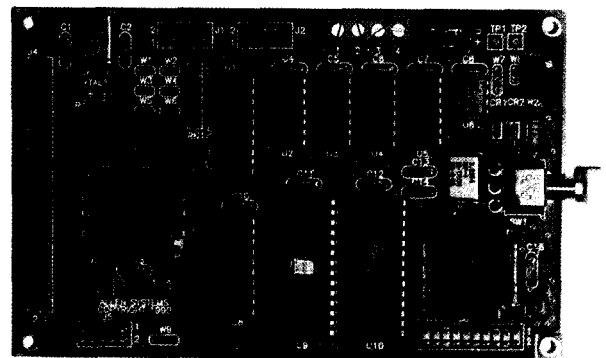
PRODUCTIVITY
ENHANCEMENT
PRODUCTS

Master
Charge

26072 Merit Circle, Ste. 110 • Laguna Hills, CA 92653
1(800) 451-DISK • (714) 348-1011 • FAX (714) 348-1310

#120

8051, 8096, 68HC11, 68332 SINGLE BOARD COMPUTERS



We feature a series of single board computers for process control applications. Each is available as a bare printed circuit board, or fully assembled and tested. Optional development software is also available. Please contact us to discuss your requirements and receive a literature package covering technical specs and pricing.

ALLEN SYSTEMS

2346 Brandon Road • Columbus, OH 43221
(614) 488-7122

Taken at their most basic level, computers operate using only a small number of circuit types repeated many times over. The transistors used to make them are arranged to imitate Boolean algebra. These simple operations can be combined to form all of mathematics; thus, they can form all of computing. Some of these basic operations are OR, AND, NOT, and exclusive OR (XOR). Connected together, they make up the familiar NOR and NAND gates worked with in electronics design every day.

According to Boolean math, only two of these operations are required to produce all of the others and all of computing. For example, an AND tied into a NOT makes a NAND. If you wanted to, you could make an entire computer composed strictly of 74LS00 NAND gates, even if each pulse does waste 10 ns to travel a tenth of an inch from an input pin to its output pin.

While the NANDs and NORs are most common in electronic computers, two others are of special interest in photonics. They are the OR and the XOR. I'm sure examples of these two types of circuits are familiar to you.

As with the other Boolean functions, combinations of these two types of circuits can create all of mathematics and every type of circuit that a computer needs, including memory. The photonic transistor performs these two functions beautifully and swiftly, along with signal amplification and a number of analog functions.

THE LIGHT AT THE END OF THE TUNNEL

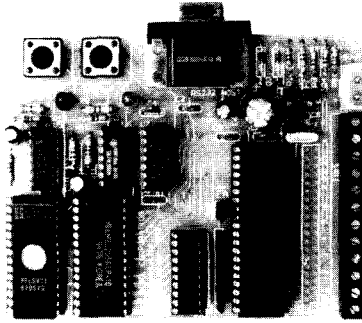
Back in 1801, Thomas Young performed an experiment that showed that light has a wavelike nature. He did this by setting up an experiment whereby two beams of light from a common source were superimposed upon each other (see Figure 1). The light pattern produced was called interference, which could be measured in a manner similar to ocean waves. Later, individual photons were also shown to possess this ability.

Today, lasers and a Michelson Interferometer are commonly used to demonstrate the effects of interference, even though the geometric configura-

A COMPLETE DEVELOPMENT SYSTEM FOR THE 8031 FAMILY OF MICROCONTROLLERS

A Price-Performance Breakthrough...

READS (Rigel's Embedded Applications Development System) and the R-31 J board constitute a complete hardware/software development and debugging system in one user-friendly menu-driven environment which runs on a IBM PC host.



Programs in the MCS-51 language may be written, edited, assembled, downloaded and debugged without leaving the integrated environment. Its advanced features such as extensive menus and on-line help makes it ideal for beginners. The professional features, such as alternate hotkey operation, source-level debugging, history screens, and a powerful cross-assembler makes READS / R-31 J an industrial-grade development system.

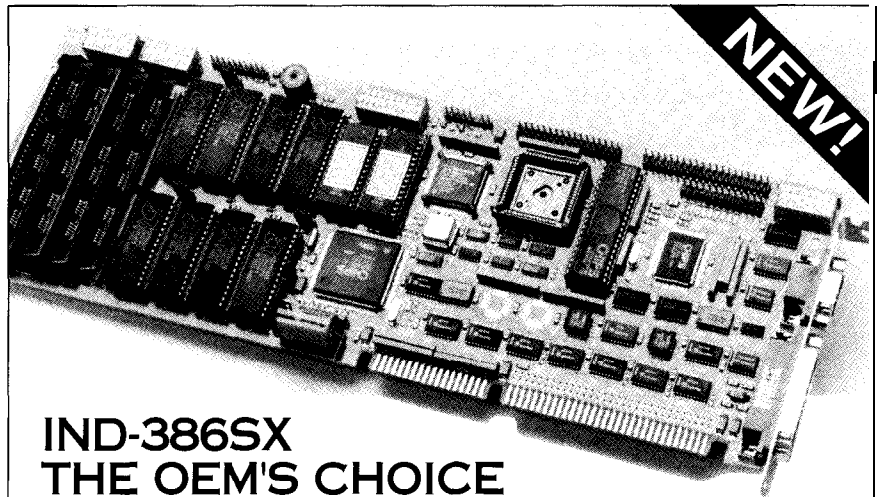
The R-31 J may be populated with Intel's 8052 Basic chip and software written in Basic for those who prefer it.

READS/R-31 J with User's Guide on disk and example programs, is priced at \$130. A kit is available for \$95. Add \$30 for the Intel 8052 Basic chip.

RIGEL CORPORATION

PO BOX 90040, GAINESVILLE FL, 32607 (904)373-4629

#121



IND-386SX THE OEM'S CHOICE

FOR RUGGED, RELIABLE PERFORMANCE

MCSI introduces the new diskless IND-386SX single board computer featuring the PROMDISK® multi-drive disk emulator.

- Occupies only a single slot!
- Low-power CMOS architecture.
- OEM configuration switches.
- 100% PC/AT compatible & directly runs MS-DOS, QNX, etc.
- High speed (25MHz).
- 16Meg DRAM.
- * 4Meg PROMDISK®.
- Co-processor socket.

Made in the USA, MCSI products are backed with a level of service that can't be matched.

For optimal performance on your next project, call MCSI at (619) 598-2177 today and ask for your free 30-day evaluation.



2598-G Fortune Way, Vista, CA 92083
Tel. 619/598-2177 • fax 619/598-2450

M C S I - T H E E M B E D D E D P C S P E C I A L I S T S

#122

See us at the Embedded Systems Conference - Booth #71

The Computer Applications Journal Issue #28 August/September, 1992

43

tion of the Young experiment differs from Michelson's. What is important here is that two beams of light from a single coherent light source are recombined by superimposing one beam on top of the other.

Figure 2 is a close-up of the light pattern in Figure 1. It has three sections, so first examine the lower one that shows the bands of light and dark. This figure illustrates what is called an **interference fringe**, which results from the recombination of two beams. The light portion is called **constructive interference** (CI), and the area of darkness called **destructive interference** (DI). (These terms are misnomers, for nothing is really constructed nor is anything destroyed.)

Photons affect one another differently when the two beams are traveling together than when only a single beam is present. When both beams are on, interference causes the photons to migrate toward each other. Photons that ordinarily would have been flying in the DI areas have been pulled to the side into the CI areas.

However, when only a single beam is on, no interference is present and the entire area is illuminated as depicted in the center section of Figure 2. The photonic transistor exploits this natural effect in order to produce the two Boolean functions OR and XOR.

Like all Boolean operators, the photonic transistor has two inputs: the two light beams of Figure 1. Switching these beams on and off can represent binary bits of information. Now take a look at the top section of Figure 2. It has no light at all, representing when both beams are off, a moot case. Thus, Figure 2 depicts three states:

1. Both inputs are off, there is no light input.
2. If either one or the other is on, the area is evenly lit although no interference fringe exists.
3. When both beams are on, the interference fringe forms.

A PHOTONIC OR

In an actual photonic transistor, the entire fringe may be used. However, to understand how they work, let

me zoom in on the small circles located in the CI and DI areas. Take a piece of cardboard or paper a couple of inches square and punch a hole in the middle of it. This piece of paper represents a photonic transistor. Figure 2 represents the input to the transistor in its various states. Your eye is the detector for viewing the output.

Place the paper so the hole is lined up with area "A" in Figure 2. In the moot case, both beams are off, so no light is output through the hole.

Now move the paper down to the center section to area "B." This point represents the exact same location as A, only now one of the beams has been activated. Note that either beam will turn on the output. Although there is no interference fringe, light is still output through the hole.

Now move your paper down to the lower section, to the area labeled "C." Again, this point represents the exact same position, only now both beams are on and the interference fringe has come into existence. Note that light is output through the hole. In fact, the

SUPER NEW FEATURES ADDED TO PCB II

EASY TO USE CAD SOFTWARE

THE BEST JUST GOT BETTER

AUTO TRACK NECKING

This feature will **AUTOMATICALLY** ADJUST the track according to the DESIGN rules you set.



CURVED TRACKS

Supports true CURVED Tracks.



GERBER VIEWING

Lets you verify your GERBER output on screen before sending it to the PC Board house.

"THE BEST PCB CAD AVAILABLE"

PCB II - STILL ONLY \$149^{US}!!!

R4 SYSTEMS Inc.
P.O.Box 451
West Hill, Ontario
Canada M1E 4Y9
(416) 399-0943

Free Evaluation Package
Write or Call Today

Download DEMO from BBS and SAVE \$10 on your order
BBS at 416 289-4554 (2400/8/N/1)

DSP

HARDWARE

Check out our complete line of DSP boards based on powerful floating-point processors like the AT&T DSP32c (25 MFlops) and the Analog Devices ADSP-21020 (75 MFlops). Several analog interface modules are available. DSP boards start at just \$995.

SOFTWARE

We have everything you need to do DSP software development, including C compilers, assemblers, source-level debuggers, algorithm development tools, and many example programs. Data can be transferred between the DSP board and host at up to 3 Mbytes/sec with the host interface library (source code included).

SOLUTIONS

Call our friendly, knowledgeable staff to discuss your applications and we'll show you how easy it is to take advantage of DSP technology.

800-848-0436

BITWARE
RESEARCH SYSTEMS

400 East Pratt Street • 8th Floor • Baltimore • MD • 21202
More Info: 800-848-0436 • FAX: 410-783-7375

light coming through the hole is four times brighter because of the CI than it is when only one beam is on at position B. That there is output through the hole in the paper mask when both input beams are on is what is important.

In your hand, you hold a photonic transistor, albeit macro in size and crude in appearance. In this position relative to the fringe, it provides the OR function. Light coming from the paper through the hole in the mask to your eye travels at the highest speed known. It doesn't have to slow down or introduce any delays when providing this basic function.

A PHOTONIC XOR

However, two are required to tango, and two Boolean functions are needed in order to produce the others [and all of computing]. So take your paper photonic transistor and place it over the area marked "D." This new position is relative to the fringe that will be used to perform the XOR function. In this state, both beams are off—

so no output. Move it down to "E," which represents the same position as D in the first state, and again, only one beam is on, so there is output.

Now move the mask so the hole is over position "F." What's different? Both beams are still on, but because of the DI, the photons have been shoved to the side and out of alignment with the line of sight through the hole. The input light is now reflected into another pathway, absorbed or whatever by the mask. So the output through the hole is OFF. The device is a light-speed XOR.

Notice that without the mask, light from the two inputs would still exist in the output. Without the separation of these fringe component regions, the function is lost. The information manifested by the existence of the fringe disappears when beams of light from the separate regions are allowed to mingle back together again. Only with the mask in place does a separation of the information occur in the fringe component regions.

OTHER FUNCTIONS

As with all XOR gates, if one beam is kept on all the time using a DI-positioned mask, and the second beam is alternately turned on and off, you're switching off and on the output—that is, when the modulated input beam is on, the output is off, and vice versa. Therefore, it is a photonic inverter, or the equivalent of a NOT circuit.

An interesting thing happens, though, when a CI positioned mask is operated with one beam always on. When the second beam is switched off, the output is on because of this constant "bias" beam. Now, when the second beam is turned on, interference relocates photons that used to be in the DI areas into the CI areas and through the hole to become the output. The intensity is now four times greater than it was with only one switched-on beam. How can that be?

Say that in a certain time, 200 photons enter from one beam, and if the second beam is on, 200 photons

► Project Parts ◀

DS1210 NVRAM controller	9.75
DS1232 Micro watchdog	6.75
IS1 U60 38 kHz IR receiver	4.80
LD273 dual IR LED (bright!)	2.10
Excellent IR filter (opaque!)	6.50
MC145030 IR encoder/decoder	6.75
MT8809 8x8 analog crosspoint	11.50
CS212 SART I/O network chip	8.10
16C55 logic analyzer w/DRAM	19.00
TW523 X10 P-way interface	45.00
IL300 linear optoisolator	12.70

More good stuff -- switching regulators & coils, single-chip micros & system support, Chip Sacks...

UPS Ground/2nd day/next day \$6/8/16 to 48 US states, COD add \$4. Check or MO only, no credit cards or POs. CT residents add 6% sales tax. Quantity discounts! Data sheets included!

Pure Unobtainium

► Your unusual part5 source ◀

89 Burbank Road
Tolland, CT 06084-2416
FAX/voice (203) 870-9304

PLIX™ HARDWARE X-10™ TRANSCIEVER CHIP

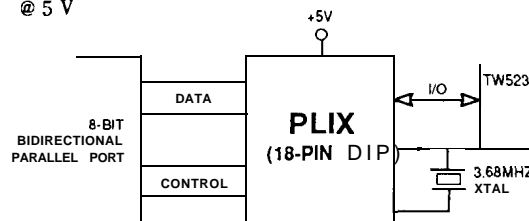
Micromint introduces its new Power Line interface for X-10. PLIX is an 18-pin ASIC chip that automatically handles all the specialized X-10 timing and bit-shuffling between a computer and a TW523 power line module.

- ✱ Complete interface between parallel port and TW523 or PL513 modules
- ✱ Performs all X-10 transmit and receive functions
- ✱ Detects AC power loss
- ✱ Simple interface and timing — operates even in interpreter BASIC
- ✱ Low power — only 1.8 mA @ 5 V

PLIX chip and data sheet - **\$20**
100 qty. OEM - **\$12**

Call 1-800-635-3355
or
Write for a PLIX data sheet

X-10 is a trademark of X-10(USA), Inc.



MICROMINT, INC.
4 PARK ST., VERNON, CT 06066
(203) 871-6170 FAX: (203) 872-2204

from it. With only one constant bias beam on, the first 200 photons are spread over the entire surface of the mask. If the hole in the mask is over only the CI area, half of the light will go through the hole and half will be stopped or diverted by the mask. So the output through the hole will be just 100 photons.

When the second beam comes on, the interference focuses all of the light into the CI areas. So along with the original 100 photons coming through the hole, the other 100 of the bias beam are shoved over into the CI area and through the hole. At the same time, the second 200 from the other beam are also focused out through the hole, so the total output is 400 photons.

Because the constant bias beam carries no **information**, the modulated signal output is greater than it was in the beginning. By using additional photonic transistors or by phasing pulses to change the fringe position, combination photonic transistors can be constructed that, when both inputs

are on, remove the constant 100 photon carrier from the output while not harming the 400.

Therefore, the photonic transistor is an amplifier like its electronic cousin. Granted, the gain is small, but that this gain exists even in these primitive examples is what is important. By using optical systems that change the shape of the fringes and the proportion of DI area to CI area, the actual gain may be tuned for optimum performance. Then, a number of photonic transistors can be cascaded together to produce an appreciable gain.

Please note this type of amplification is signal and not light, the type that takes place in lasers. That is a different process, for a different purpose.

A CI device and a DI device can be made from the same mask simply by adjusting the position of the fringe. The fringe position can be shifted by a slight phase change in one of the beams. This adjustment also makes the photonic transistor into a demodu-

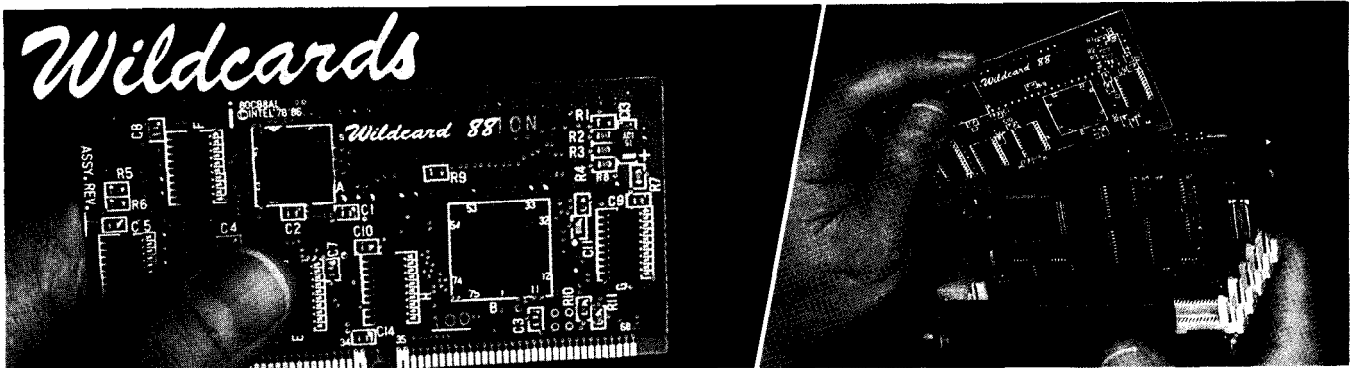
lator for phase-modulated signals because the resulting output is amplitude modulated.

A SIMPLE DEMONSTRATION

Placed between the beam-combining optics and the display screen of a Michelson Interferometer, as in Figure 3, the mask in your hand can be made to function as the world's fastest transistor. The Michelson Interferometer breaks the source laser beam into two and recombines them again in the output. By blocking the light at the two side paths of the interferometer as needed, the two input beams may be turned on and off in order to demonstrate all the input and corresponding output states of this macroscopic photonic transistor.

The switching speed of a particular photonic transistor is the time it takes light to travel from the beam-combining optics to the mask. The closer they are together, the faster the transistor. Anything smaller than about an inch is faster than the fastest electronic transistor, so imagine what

2" x 4" EMBEDDED PC



Microcontroller.

Microcomputer.

"Megatel Wildcards provide PC functionality in a flexible, small format."

Wildcard 88™

- CPU clock to 10 MHz
- Replaces full PC motherboard
- Co-processor and BIOS socket
- DMA, Bus, DRAM, Keyboard controllers

Multi/10

- On-board SCSI Host Adapter (supports up to 7 devices)
- Floppy Controller (1.44M, 1.2M)
- 2 RS-232, 1 Parallel, 1 RS-485 multi-protocol serial port

Vid/Mem:

- 640Kb User memory
- Video/Colour LCD controls CGA, Hercules®, IBM® Mono; (runs LCD Panels)

For information on our representatives please contact our head office at the number below.

(416) 245-3324

All Wildcards are low power single +5 volt operation.

125 Wendell Ave., Weston, Ont. M9N3K9 Fax: (416) 245-6505

Wildcard 88 and Megatel are trademarks of Megatel Computer Corp. Hercules is a trademark of Hercules Corp. IBM is a trademark of IBM Corp.

megatel®

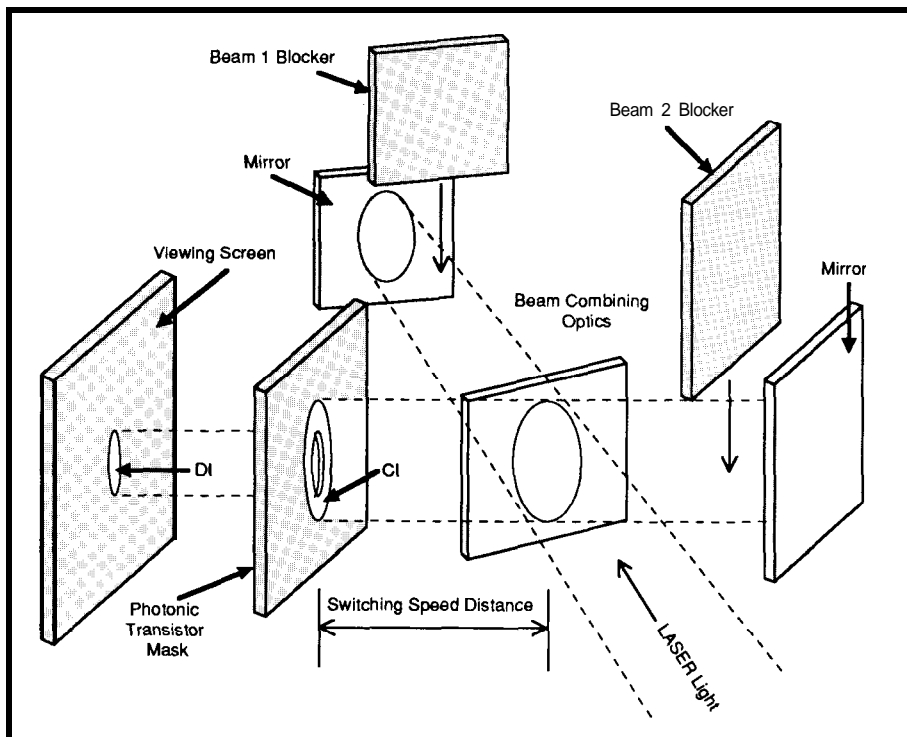


Figure 3—*Photonic transistor demonstration* using a Michelson Interferometer with a fringe component separating mask placed between the beam combining optics and the viewing screen.

kind of speed is possible with microscopic components.

In production, photonic transistors can be made very small—near the size of the wavelength of light being used. The higher the frequency, the shorter the wavelength. The shorter the wavelength, the smaller and more closely they can be packed together, and the faster the computer.

DEVELOPMENT

Will you be able to buy a desktop photonic supercomputer next week? Maybe not that quickly, but soon. While development will take some time, I estimate that the first photonic hardware may be replacing some electronic computers within five years and accelerate from there.

Photonic transistors are so general in their nature that predicting which products will be developed first is difficult. As with electronic transistors, they are applicable to just about everything. The first products will be software for the production of photonic transistor photographs and interconnecting holograms, demonstration products, and individually connected photonic transistors. These are expected very soon as we arrange R&D

with the variety of interested groups both large and small. The next products are expected to be specialized devices, such as telephone fiber-optic switching systems and add-on products for speeding up electronic processing. Then of course, fully photonic **teraFLOPS** computers as the photonic-transistor-producing software becomes operational.

Will there be problems with photonic development? Certainly. There will always be challenges. However, those difficulties will not arise from any need to research and create specialized materials as with other optical methods or to figure out some unknown quirk of physics. Rather, they are merely the geometric problems of engineering the organizational and architectural arrangement of components, using optical laws that are well understood.

There is another important reason why photonic development will be much more rapid than was the development of its electronic counterpart. Although the photonic transistor stands today where the electronic transistor stood 40 years ago, the great body of computer science was in its infancy. Modern-day manufacturing

technology didn't exist. The pictures that were used to fabricate the first computer chips were drawn by hand. The entire ordeal was time consuming. Early electronic transistors had to be individually wired in by hand. Printed circuits didn't exist.

Today, the art of holography is well understood and is used to interconnect digital light beams. Like other photographs, holograms and the photographic masks that make up photonic computers can be produced by computer, calculated into existence from the basic math they are derived from. The well-known laws of optics, existing equipment, hologram-producing programs already available, and the principles of the photonic transistor are the ingredients for the development time acceleration of the photonic computers.

Today's tools are much faster than those of yesteryear. Computer-aided design compresses years of development time into months or even hours. The great body of computer science is mature and well adapted for each new computer upgrade and is poised for the photonic conversion, which will just be the next upgrade.

Remember what happened to the slide rule? ☐

John Hait is an electronics engineer with 35 years of experience in basic circuit design and troubleshooting electronic and data processing systems. He is best known by his books and articles on practical applications of thermodynamics and electrochemistry. He is the inventor of the photonic transistor, which is only one of 85 inventions that span the wide spectrum of practical physics.

SOURCE

Rocky Mountain Research Center
P.O. Box #4694
Missoula, MT 59806
(406) 728-5951

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful



Special Section EMBEDDED INTERFACING

50 Closing the Loop on
DC Motor Control
Tom Dahlin & Don Krantz

58 Designing with
Programmable Logic
Charles R. Conkling, **Jr.**

Closing the Loop on DC Motor Control

Stepper motors have been the mainstay of most precision motor control applications. By using the LM628 for control, a DC motor may be a suitable replacement for a stepper and provide some additional benefits.

SPECIAL SECTION

Tom Dahlin & Don Krantz



Motor control is a recurring theme in the *Computer Applications Journal*. In the past, most of the motor control schemes in this magazine and in others have used stepper motors for precision positioning. We would like to present a different method: using the National LM628 DC motor control chip to drive a DC motor. With the LM628, you can interface a DC motor to a microprocessor and control it as easily as a stepper motor.

STEPPER MOTORS VERSUS DC MOTORS

Stepper motors are popular for several reasons. They are relatively easy to interface with a microprocessor. They are easy to actuate, unless you try something exotic like microstepping in software. They can be controlled open loop with a reasonable amount of confidence—that is,

they don't require a position transducer to tell you how far they have moved and what is the present angle of the shaft.

Steppers do have several disadvantages. They "clunk" from step to step, introducing vibration unacceptable in some applications. They can slip under high inertial load either when run near the high end of their speed range or when counter torque is applied. (If a stepper motor does slip, the controller loses track of the actual position and may not know of the slip unless it periodically indexes the load.) Stepper motors also have relatively low top speeds (ZOO-400 RPM) and require high currents even when stopped.

A DC motor is capable of much higher speeds than a stepper motor. It also runs more smoothly because it doesn't move in discrete steps. It causes less electronic noise than a stepper and is available in much higher torque ranges. However, a stepper motor is more capable of determining and controlling position and velocity than the DC motor. A DC motor requires a shaft encoder, or other external sensor, if it is used in positioning applications, and it requires a more sophisticated control system than a stepper motor.

Using the LM628, you can now mitigate the DC motor's disadvantages. The following description presents a simple control interface for position and velocity determination in DC motors.

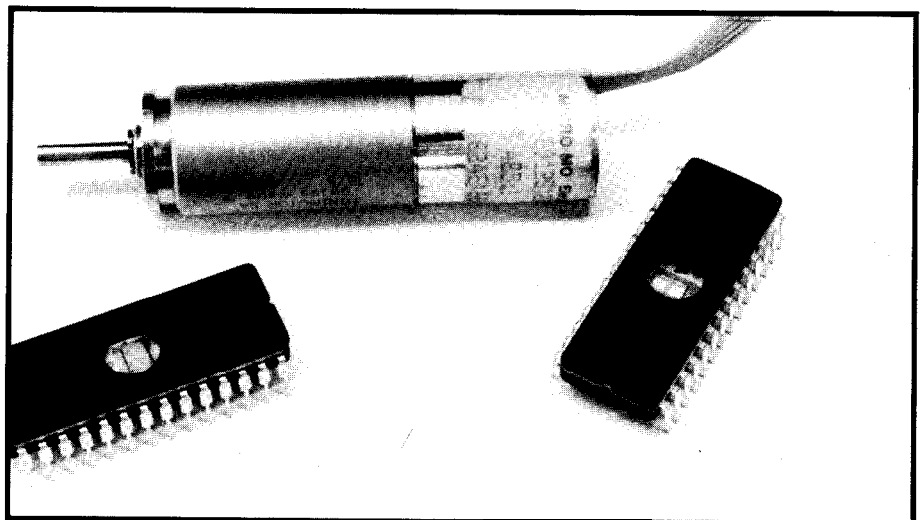


Photo 1—The Micro-Mo 1624E01 is an example of a DC motor with built-in shaft encoder that is ideally suited for use with the LM628.

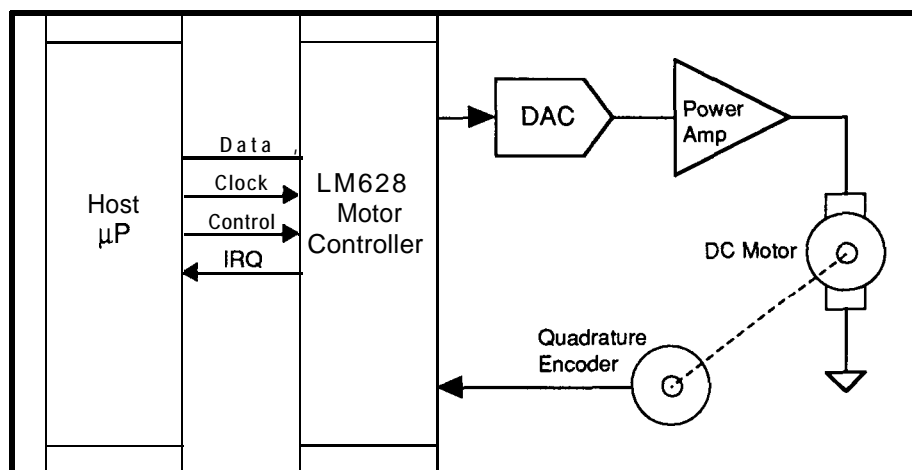


Figure 1—In a typical application of the LM628, a DAC and power amplifier control a DC motor while a shaft encoder within the motor provides position feedback to the LM628.

THE LM628 AND LM629

The LM628 is a motion control processing chip manufactured by National Semiconductor. It is specifically designed to control a DC motor and quadrature encoder combination. The LM628 is an analog-output device [counting the external DAC). National also makes the LM629, an identical part that has an added pulse-width modulated (PWM) output. We generally use the LM628 because it is easier to debug and tune in the types of systems we build. The PWM version of the part is useful in high-power or high-efficiency applications.

Figure 1 shows a block diagram of a typical LM628 system. The LM628 connects directly to a DAC, which connects to a power amp. The DAC can be either an 8-bit or a 12-bit model. The choice between 8 or 12 bits is one of those personal things like religion or mouse-driven GUIs that doesn't seem amenable to reason. We use 8-bit DACs and they work just fine. As an added point, the interface to a 12-bit DAC is more complicated than the 8-bit model because the extra four data bits are multiplexed with the basic eight data bits.

A quadrature encoder attached to the motor shaft connects directly to the LM628. Usually, encoders are factory installed on suitable

DC motors (see Photo 1 for an example). An optional index signal can also be connected to the LM628 to signal a "home" position.

The LM628 uses the quadrature feedback to monitor the motor. The LM628 will run in two basic modes—position or velocity control—either separately or at the same time. In position control mode, the user tells the LM628 how many encoder counts to move [how many steps to take). In velocity control mode, the user tells the LM628 the desired motor velocity. In both modes, the user can program acceleration and deceleration ramps, maximum velocity, and PID (Proportional-Integral-Derivative) filter

constants.

HOW IT WORKS

Figure 2 is a functional block diagram of the LM628. The major components are the host interface, the trajectory generator, the position decoder, and the PID filter processor.

The host interface synchronizes the host commands to the internal processor of the LM628. Table 1 shows the LM628 command set.

The trajectory generator sets up a desired motion profile, or trajectory. The trajectory is constructed using the control mode (position or velocity), the acceleration ramps, and the maximum velocity.

The quadrature decoder allows calculation of the actual motor position using the quadrature inputs. Quadrature encoding uses two out-of-phase pulse trains, allowing the LM628 to detect direction of travel as well as the angular distance moved.

Actual position from the decoder is summed with the desired position from the trajectory generator to produce an error term, which is fed into an internal PID filter processor. The PID filter output is placed in the DAC output pins. PID algorithms have been discussed elsewhere [1], so we will not cover them here.

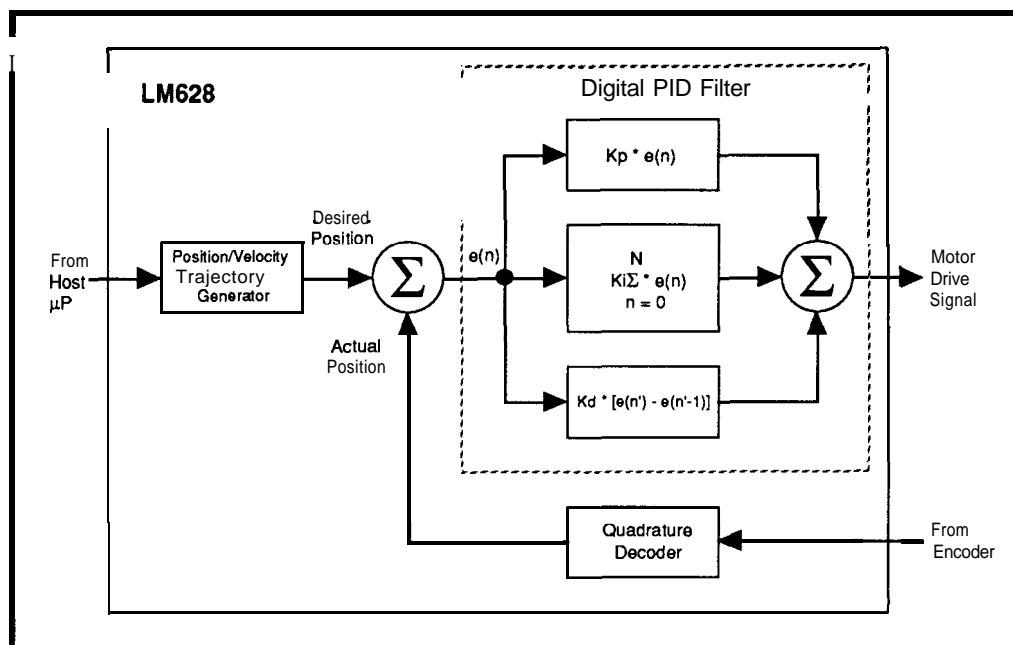


Figure 2—The LM628 has a proportional-integral-derivative (PID) filter built tight into it to off-load much of the motor control burden from the main processor.

amps for this purpose. At this point, the output is an analog voltage that swings positive for forward and negative for reverse. Zero volts are used for motor stop.

Next, you'll need a power amp. We normally use an LM675 power op-amp for this stage, but this part has an annoying tendency to oscillate near unity gain no matter what compensation, bypassing, short leads, or ground planes are used. To solve this problem we attenuated the input by a factor of 20, then ran the LM675 at a gain of 20. While this solution isn't exactly elegant engineering, it does work. When we have the time and inclination, we will probably select a different part for this application.

Remember that a motor looks like a big inductor to the power amp. You have to use diodes to the power rails to protect the power amplifier. The LM675 has these diodes built in, but other amplifier ICs may not.

The encoder outputs from the motor should be tied directly to the LM628 encoder inputs. If the encoder (or other component of the system) has an index pulse output, connect it to the LM628 also. Otherwise, tie the index pulse input high.

PROGRAMMING STRATEGY

The LM628 requires some setting up before it can control a motor. Listing 1 contains some basic C routines to interface to the LM628 and example reset and setup routines taken from one of our projects. The basic programming operations are simple, but deciding what to program into the chip is somewhat more complicated.

Most of us programming types don't have the theoretical controls background (or know the system's physical characteristics well enough) to determine filter parameters a priori. If you know something about Laplace transforms and have good data on the system inertial loading, you can use the analytical approach (see Futher Reading).

Fortunately, there's nothing magic about PID filters. Unless you're designing the pump controllers for heart-lung machines, the empirical approach works well enough and is

Distributed Control and Data Acquisition

NBS-10 Interface Card provides gateway to network

RS 485 twisted pair

PC Master

Embedded controllers on a multidrop network

slave controller 1 8051

slave controller 2 68HC11

slave controller 3 Z180

Connectivity is implemented at low cost with a network designed by **CIMETRICS TECHNOLOGY**.

Our hardware/software solution includes:
Microcontroller code, PC/XT/AT interface card, NSP protocol, and application development software toolkit.

CIMETRICS TECHNOLOGY
the network solution for embedded controllers

120 West State Street, Ithaca, New York, 14850
Phone: 607-273-5715 FAX: 607-273-5712

#127

See us at the *Embedded Systems Conference—Booth #602*

286 to 486

UPGRADE NOW!

Easy to install module plugs directly into your 286 CPU socket.!

Only \$499

Free Info By FAX Today
Upgrade Tomorrow!

TECHDirect

FAX: 813-442-5184
VOICE: 800-275-8344

#128

usually even faster than the analytical method. The difference is because accurate data on the system inertial loading is hard to come by, so you end up tweaking the analytical parameters anyhow.

To set up the system empirically, you need a functional physical system (i.e., a motor and a board) connected to the controller. Start by programming a trajectory [3,4]. National recommends starting with a very low proportional gain ($K_p = 1$) and zero derivative and integral terms ($K_D = K_I = 0$). This arrangement allows you to see if the feedback phasing is correct [if not, the system will run full speed or oscillate]. In our case, the software guy assumed (correctly) that the hardware guy had hooked it up backwards, but still he had to prove it.

Once the loop phase is correct, turn K_p up to about 20. This adjustment will give the system a "springy" response. The controller will hold the motor shaft in place. If the shaft is deflected, the system will apply increasing voltage to the motor to

Listing 1—A C program that deals with the LM628 consists primarily of statements that send commands to the chip.

```
/*-----
LM628 motor controller interface & setup routines
-----*/

/* Example address map for the motor controller LM628 */
#define CMD (* ((unsigned char *) 0x9000))
#define STATUS (* ((unsigned char *) 0x9000))
#define DATA (* ((unsigned char *) 0x9002))

/*-----
Writes 16 bits of data to the LM628. in the prescribed order
-----*/

write_data16(unsigned int d)
{
    while (STATUS & 0x01)
        ;
    DATA = (d >> 8) & 0xFF;
    DATA = d & 0xFF;
}

/*-----
Writes 8 bits of data to the LM628
-----*/

write_data8(unsigned char c)
{
    while (STATUS & 0x01)
        ;
    DATA = c;
}
```

(continued)

8052 development system



Introducing the BD52, a complete 8052 development system providing all hardware and software needed to develop 8032/8052-based products. A flexible embedded controller with pseudo-EPROM

capabilities allows you to download and test your code for easy software development. Memory and interface extensions enhance the BD52's capabilities. A PC-based windowed interface gives you complete control of software and hardware functions. The powerful development tools include a fully featured assembler, C-compiler and debug monitor. Software and hardware are closely integrated for greater productivity at an extremely competitive price.

Hardware

- 8032 microcontroller and EPROM loaded with debug monitor
- ability to address 64K of code space and 64K of data space
- pseudo-EPROM capacity: 30K with 32K SRAM (included)
- RS232 serial interface, 11 parallel I/O lines
- 120V AC power supply; documentation and schematics
- sockets for extensions: watchdog & power monitor, 4 A/D & 1 D/A converters, 512 byte EEPROM for non-volatile data storage, 7-line relay driver, 32K SRAM for up to 62K pseudo-EPROM capacity

Development Software

- MICRO-C/52 'C' compiler featuring 5 memory models and a standard library with extensions to control BD52's hardware features
- ASM52 cross assembler
- PC-hosted control/debug software with easy-to-use windowed interface that provides access to all of the BD52's features

Order a BD52 at the introductory price of \$199.95* US from:

*please add \$10.00 for shipping

Dunfield Development Systems
P.O. Box 31044, Nepean, Ont. Canada K2B 8S8
Phone: 613-256-5820

Complete pricing and free catalog of hardware and software products available.

THE \$99.95 EMBEDDED EDUCATION

THE PRIMER MICROPROCESSOR TRAINING SYSTEM

- TEACHES:**
- * INTEL 8085 PROGRAMMING
 - * DIGITAL & ANALOG INTERFACING
 - * PROGRAMMING INTEL PERIPHERALS
 - * MICROCOMPUTER DESIGN & ASSEMBLY
- FEATURES:**
- * MONITOR O.S. SOFTWARE IN EPROM
 - * OVER 100 PAGE SELF INSTRUCTION MANUAL
 - * 6 DIGIT, 7 SEGMENT, LED DISPLAY
 - * 20KEYKEYPAD
 - * DIGITAL INPUT PORT WITH DIPSWITCH
 - * DIGITAL OUTPUT PORT WITH LEDs
 - * ANALOG TO DIGITAL CONVERTER
 - * DIGITAL TO ANALOG CONVERTER
 - * TIMER/COUNTER WITH SPEAKER OUTPUT
- OPTIONS:**
- * BASIC OR FORTH LANGUAGES IN EPROM
 - * RS232 SERIAL PORTCONNECTSTO PC
 - * BATTERY BACKED CLOCK AND RAM
 - * 9 VOLT 500 MA. POWER SUPPLY

EMAC OFFERS A COMPLETE LINE OF MICROPROCESSOR TRAINING SYSTEMS STARTING AT \$99.95 QUANTITY 10 FOR THE PRIMERKIT.

EMAC, inc.

618-529-4525 FAX: 618-457-0110

P.O. BOX 2042 CARBONDALE, IL 62902

```

/*-----
Writes a command byte to the LM628
-----*/
write_cmd(unsigned char d)
{
    while (STATUS & 0x01);
    CMD = d;
}
/*-----
Returns 8-bit data value from the LM628
-----*/
unsigned char read_data8()
{
    while (STATUS & 0x01);
    return(DATA);
}
/*-----
Programs a simple proportional filter setup into the LM628
-----*/
NOLOCAL void
setup_mc()

    write_cmd(0x1E);      /* LFIL (Load Filter) */
    write_data16(0x000F); /* parameter mask */
    write_data16(0x0100); /* Kp */
    write_data16(0x0000); /* Ki */
    write_data16(0x0100); /* Kd */
    write_data16(0x7FFF); /* IL */
    write_cmd(0x04);      /* UDF (UpDate Filter) */

```

(continued)

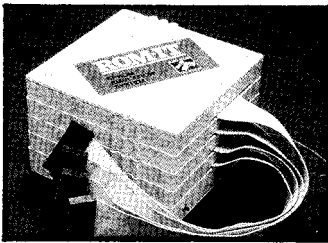
return the shaft to the starting position. If the motor oscillates or rings when you try to make this change, back K_p off until it quits.

Next, you set up the trajectory values. Turn up the acceleration and velocity and give the system **short-motion** commands. Monitor the actual and desired velocities and positions by reading the registers of the LM628 during the acceleration and **constant-speed** parts of the motion profile. When the errors start increasing, you've reached the limits of the physical system. If the acceleration and velocity parameters are set too high, you will have trouble setting up the filter parameters. If they are too low, the system will be sluggish. Acceleration and velocity are related, but will exhibit some independence, so complete the final tweaking on these parameters individually.

Once the trajectory parameters are tuned, you can begin tuning the PID filter by iteratively turning up the proportional term (K_p) and the derivative term (K_d). As K_p increases, the

ROM-IT™

EPROM EMULATION SYSTEM



The Most Flexible EPROM Emulator You Can Get Today!

- Emulates up to 8 **4-Megabit EPROMS** through one standard serial port.
- Downloads 2-Megabit programs in less than 23 seconds.
- Allows you to examine and modify individual bytes or blocks.
- Accepts Intel Hex, Motorola S-Record and Binary files.
- Software available for IBM PC and compatibles.
- Base 27256 EPROM System \$395.00. Other configurations available.

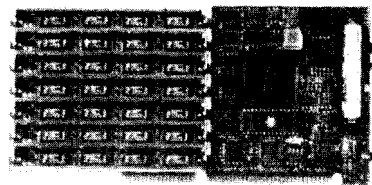
**ORDER TODAY—IT'S EASY
CALL OR FAX FOR MORE INFORMATION**



Incredible Technologies, Inc.
(708) 437-2433
(708) 437-2473 Fax

Visa and MasterCard accepted.

ROMDISK™



High Performance Multimegabyte Disk Emulators

NEW MODELS / LOWER PRICES

- Floppy Drive and multimegabyte emulators for ISA bus computers
- 180K to 14 MB capacities
- EPROM, Flash or SRAM technologies
- Autobooting, Single or Dual disk emulation under PC or MS DOS
- List prices from \$195

CURTIS, INC.

418 W. Country Road D . St. Paul, MN 55112

612/631-9512 FAX 612/631-9508

PC DOS is a trademark of IBM. MS DOS is a trademark of Microsoft

motor will begin to overshoot and oscillate about the final position or velocity. When this happens, K_p should be decreased. The derivative term supplies gain related to the change in error, having the effect of damping out the spring action of the K_p term. When you begin adjusting the K_D term, you will also need to set a sampling interval. Eventually, you will reach a point where the system cannot be made stable.

The final values of K_p and K_D are a tradeoff between response time and stability. To some extent, the steady-state position or velocity error should be considered as well, although the integral term (K_I) is used to control these areas.

The integration term zeros out any steady-state error. An integration limit (il) is input with K_I . Very high values of K_I will decrease stability. Very low values have little or a delayed effect on the steady-state error. Unless a very low steady-state error is important to your system, the integral term can be safely ignored.

Listing 1—continued

```

write_cmd(0x1F);          /* LTRJ (Load TRajectory) */
write_data16(0x1828);      /* control word, V and A */
write_data16(0x0001);      /* acceleration (32 bits) */
write_data16(0x0000);      /* velocity (32 bits) */
write_data16(0x0000);      /* STT (STart Trajectory) */
write_cmd(0x01);
}
/*-----
Reset routine as recommended by National application notes
-----*/

NOLOCAL void
reset_nc0
{
    for (;;) {              /* loops until correct status returned */
        reset_port_p_bit(0); /* initiate reset pulse using PIA */
        for (i=0; i<20; i++); /* wait pulse min. duration */
        set_port_p_bit(0);   /* end of reset pulse */
        for (i=0; i<1500; i++); /* 1 ms pause for internal reset */
        /* wait until status shows bits 3 and 7 set */
        for (i=0; (i<30)&&((STATUS&0xBF)!=0x84); i++);
        if (i == 30)          /* after 30 tries, chip is hung */
            continue;        /* .so try again */
        write_cmd(0x1D);      /* 'RSTI (ReSeT Interrupts) */
        write_data16(0x0000);
        while (STATUS & 0x01) /* wait until not busy */
            if (STATUS == 0x80) /* look for correct status to exit */
                break;
    }
}

```


\$299.

Engineer's Evaluation Kit
Release I

C Compiler
Assembler
Linker
Debugger/Simulator
C Library

Win the battle for control of your 8051 projects. The Engineer's Evaluation kit can free you from the morass of assembly code or obsolete compilers. Based on Franklin's winning technology, the Engineer's Evaluation Kit is a complete set of C language tools. Franklin's C delivers the lean, rugged code of an assembler, with the coding and debugging power you need to win the development wars. So for only \$299, you can junk your old tools!

OTHER RESOURCES

Besides the obvious data sheet for the part [2], you really need its two application notes [3,4]. Expanding your resources always takes a little time and effort. Make this particular investment and you have a chip that presents an alternative to the motor control schemes of the past. When you're dealing with applications requiring precision positioning, there is nothing better than having an option regarding your choice of motors. 

Tom Dahlin is a Software Engineering Specialist at the 3M Company in St. Paul, Minn.

Don Krantz is an Engineering Fellow at Alliant Techsystems Inc. (formerly Honeywell Ordinance Division) in Minnetonka, Minn.

IRS

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

REFERENCES

1. S. E. Sams and J. Woehr, "Demystifying PID Control," **Embedded Systems Programming**, vol 3., no. 3, August 1990.
2. **Linear Data Book, Vol. 3, Special Devices**, National Semiconductor.
3. Steven Hunt, LM628 **Programming Guide AN-623**, National Semiconductor, April 1990.
4. David Dale, LM628/629 User **Guide AN-706**, National Semiconductor, August 1990.

FURTHER READING

Steve Ciarcia and Ed Nisley, "Circuit Cellar Stepper Motor Scanning Sonar System," **Circuit Cellar INK**, issue #4, July/August 1988.

Thomas Mosteller, "Control Theory for Embedded Control-

lers," **Circuit Cellar INK**, issue #17, October/November 1990.

Theodore F. Bogart, **LaPlace Transforms and Control Systems Theory for Technology**, John Wiley and Sons, New York, 1982.

Caxton C. Foster, **Real-Time Programming-Neglected Topics**, Addison-Wesley, Reading, MA, 1981.

P. Emmanuel and E. Leff, **Introduction to Feedback Control Systems**, McGraw-Hill, New York, 1979.

Bodine Motor Application Handbook, 4th ed.

Electrocraft Motor Control Handbook.

Jacob Tal, **Motion Control By Microprocessors**, Galil Motion Control Inc., Mountain View, CA, 1984.

Benjamin Kuo, **Automatic Control Systems**, Prentice Hall.

Nothing beats the right tools-- in power, speed, code efficiency, or ease of use, the Professional Developer's Kit commands success. It includes the industry's top rated 8051 C Compiler, high-level language debugger/simulator, banking linker, and the first multi-tasking RTOS small enough for single chip solutions. The advanced GUI and mouse control both the simulator and debugger, as well as your Intel or Nohau 8051 emulator. Franklin's best dominates where other tools fail!

For Quick Action, call the 24 Hour Hotline:

(408) 296-8056, Ext. 110

Please have your FAX number ready.

\$2495.

*Professional Developer's Kit
Release V*

Optimizing C Compiler
Real Time Multitasking OS
Graphical User Interface
HLL Debugger/Simulator
Banked Linker
Macro Assembler
Extended C Library

**FRANKLIN
SOFTWARE, INC**

888 SARATOGA AVE., #2, SAN JOSE, CA 95129
PHONE: (800) 283-4080 EXT. 890 #133

Issue #28 August/September, 1992

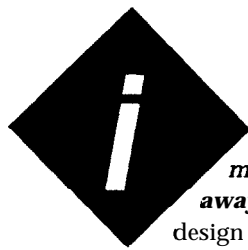
57

SPECIAL SECTION

Charles R. Conkling, Jr.

Designing with Programmable Logic

We've all seen those mysterious little custom chips called PLDs used in circuits. There are some very definite guidelines that must be followed when using PLDs, as Charles Conkling's years of experience have shown.



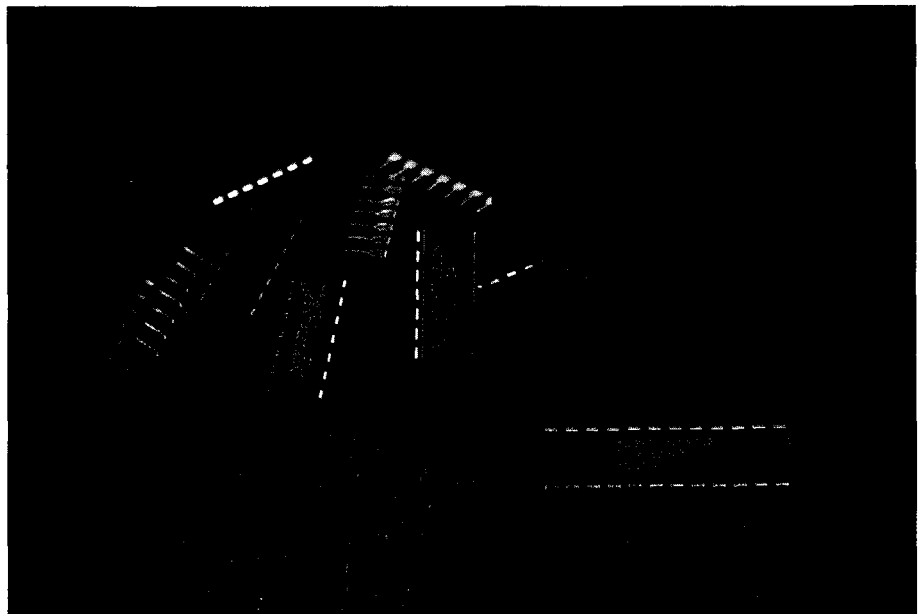
*ignore it, and
maybe it will go
away* is one of my

design philosophies. For instance, I ignored tunnel diodes and RTL, and sure enough they went away. In 1980, I wanted to ignore Programmable Array Logic chips, or PALs, because at first they had some strange logic organizations and a 45-ns propagation delay. However, just to be safe, I purchased five of each type and put them away in a drawer.

Then in 1982, a friend of mine had a design problem: 37 chips on a 36-chip board. The 37th DIP was hanging in space, wire-wrapped to its DIP pins. The only solution was a PAL, which combined the functions of two chips in one. I did my first PAL design using manual logic simplification, a marked-up photocopy of the data sheet, and a hand-keyed Prolog programmer (remember the black suitcase?). I solved the space problem, but I was asked to check the logic every time a new debugging problem arose. The new device was always suspect, equations were inverted, and you couldn't probe the minterms.

Shortly after I completed that design, I had an opportunity to replace the Prolog. I purchased a universal programmer that could program present and future programmable logic and memory devices without adapters. It came with a copy of CUPL, a logic compiler, which took about a year of "lunch times" to master. From then on, the number of 74xx chips used in my new designs decreased until one of my last designs had almost none.

To get started with programmable logic, two tools are required: a must have and a nice to have. You must have a programmer (you can't get away with a bench-top kludge here), and a logic compiler is convenient, but not absolutely necessary.



Programmable logic devices have revolutionized the way **circuits** are designed. **A single PLD can often replace** a handful of **discrete** logic with no **speed** penalty.

A programmer can cost \$400 on up. For the home experimenter, there are a number of programmers in the \$400 to \$1000 range that will program both logic and memory devices. For business purposes, you can start at \$1000 and go up from there. In either case, I would look for a programmer that has a single 40-pin programming DIP socket and doesn't require adapters for popular DIP devices. That includes most PROMs, EPROMs, 20- and 24-pin PALs, and a few EPROM-based single-chip microprocessors.

I believe the introduction of logic compilers made programmable logic devices useful. Manually simplifying logic equations and converting the results into a fuse map is difficult enough, but keying in the fuse map as devices grow more complex is an even greater problem. Most programmable logic device manufacturers provide a logic compiler. They used to be free, but because this technology has become more complex, a fee may be involved. Those compilers still included in the no-cost category are Signetics' SLICE (a no-cost version of SNAP-16 without schematic entry), Intel's PLDshell, and TI's ProLogic. AMD's PALASM, the original logic compiler, just recently went from free to \$125. Naturally, each handles only those devices made by their particular manufacturer. If you are going to program a broad spectrum of devices, I recommend ABLE from Data I/O or CUPL from Logical Devices. Finally, I have seen a few ads for low-cost logic compilers and schematic-capture logic compilers, but I have no experience with these programs.

I use CUPL at home and at work, so I'll use CUPL notation here. At home, I have a simplified (low-cost) version of CUPL made available by TI to introduce their programmable logic devices. My home copy of CUPL has the full logic compiler capability for the 16R8 family. An inexpensive copy of CUPL for the 16R8 and 20R8 families is available for about \$100 from JDR Microdevices.

Before I get into a design, I'll go over a few PAL design problems: metastability, glitches and testability, and illegal states.

METASTABILITY

Metastability occurs in flip-flops when you violate the setup or hold time specifications. It can also occur if you violate minimum clock, direct set pulse-width, or reset pulse-width specifications. Thus, metastability occurs in registered PALs with asynchronous external inputs. Sometimes the register flip-flop D-input transition will occur very close to the

The shift register buffer is built into the input macrocell in some of the latest programmable logic devices. The input macrocell allows a choice of direct, registered, and synchronized inputs. A feature available in a new Cypress device is an on-chip clock doubler. Thus, finding a double-frequency clock is easy (and has to be; a 100-MHz sample clock is required by some of today's microprocessors).

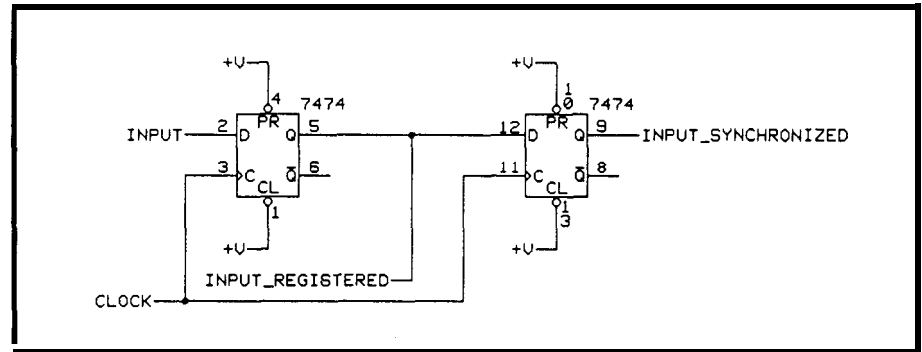


Figure 1—If the input to a two-stage shift register is synchronous to the system clock, the first stage is a "pipeline register." If the input is asynchronous, the second stage is the synchronized output.

clock edge. When this happens, some register flip-flops will do nothing, some will start to switch toward the new state but return to the old state, and some will complete the transition to the new state. During this period of uncertainty, the register flip-flops are said to be metastable.

To correct the problem, the external input must be made synchronous to the register clock. A 1- or 2-stage shift register is the usual cure. Although statisticians say this cure is not a complete one, the possibility of metastability occurring cannot be eliminated—it is always measurable. The two-stage shift register (see Figure 1) has two outputs: registered and synchronized. If the shift register input is synchronous to the system clock, the first stage is a "pipeline register." If the shift register input is asynchronous, the second stage output is the synchronized output.

To avoid a two-clock-period delay on the leading and trailing edge of the asynchronous input signal, use a synchronizer clock that is double the system clock frequency. This method works until the asynchronous input clock approaches half the logic family's maximum clock rate—then you have a problem.

A constant used to express a device's metastability is tau (τ). Tau is a measurement of the flip-flop's failure to stabilize as clock and data separation are varied. From tau, a figure of merit, T100, can be calculated. T100 is the time one must wait after the clock for the output to be stable, with a mean-time-between-failure (MTBF) of once a century. However, because metastability is a probabilistic occurrence, after waiting T100 seconds, there could be an error.

I "discovered" metastability the hard way: my first PAL state machine controller (SMC) design didn't work. A year later, I read the effect had a name—metastability—in the *IEEE Transactions on Computers* (vol. C-32, no. 12, December 1983). By that time, I had developed a few rules-of-thumb. If your clock period is greater than twice the device propagation delay, and a metastability glitch will not hurt you, you can use a single flip-flop synchronizer. Twice the propagation delay seems to fit the device T100 time (tau or T100 are now listed in some manufacturers' specification sheets).

I found that the synchronizing flip-flop can be an SMC register stage "if and only if" the input affects one register flip-flop. A PAL SMC is a

VMAX®

TOTAL PACKAGE PERFORMANCE

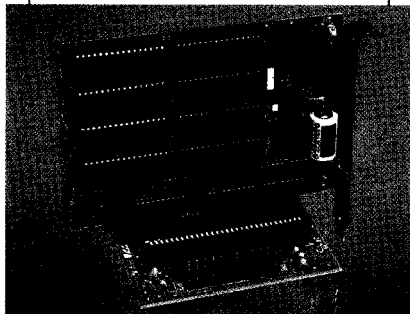
QUALITY PRODUCTS

RESPONSIVE SERVICE

RELIABLE DELIVERY

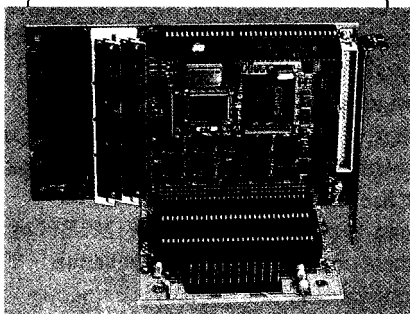
SOLID STATE DISK - \$124*

½ Card 2 Disk Emulator
EPROM, Flash and/or SRAM
1 Meg Total, Bootable



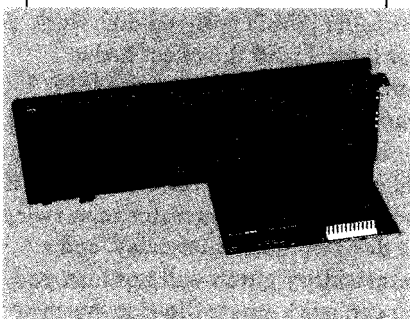
TURBO XT WITH FLASH DISK - \$266*

2 Serial, 1 Parallel Port
Up to 2 Flash Drives, 1 Meg Total
Software Included, 2 Meg DRAM



386DX Complete CPU - \$1099*

40Mhz, IDE, FDC, SVGA
1Par, 2Ser (232/422/485), Cache
Solid State Disk-2 drives to 1.5 Meg



All Tempustech VMAX® products are PC Bus Compatible. Made in U.S.A., 30 Day Money Back Guarantee
*Qty. 10, Ok, Call for Quantity Pricing

TEMPUSTECH, INC.

TEL: (800) 634-0701

FAX: (813) 643-4981

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

#134

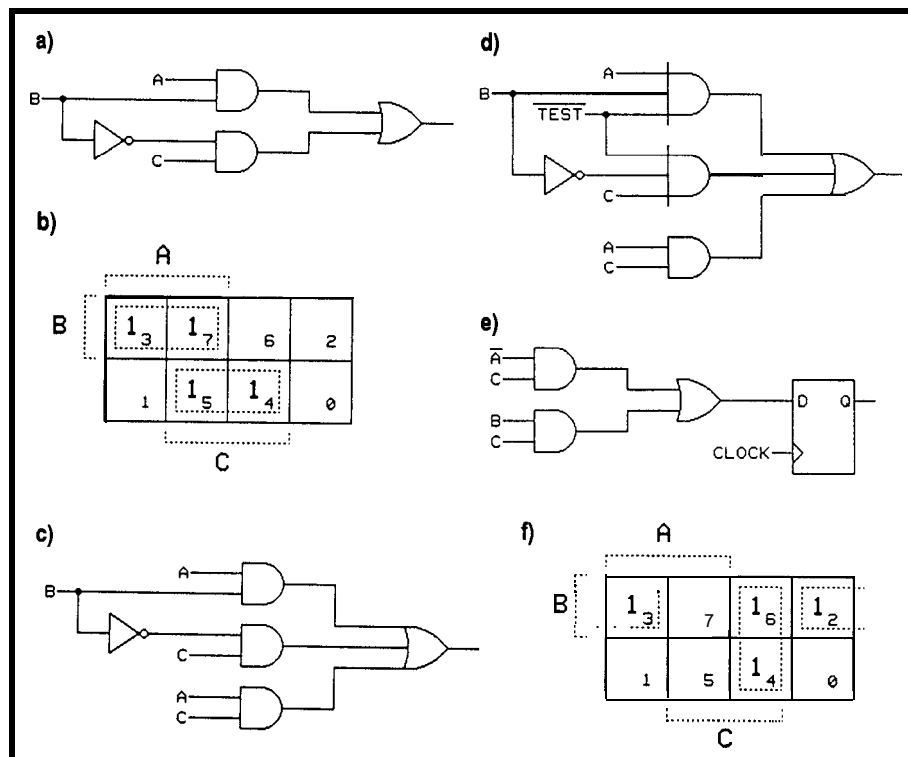


Figure 2-(a) In a typical combinational circuit, a glitch could potentially occur due to the extra delay introduced by the inverter. (b) Eliminating overlap on the Veitch diagram is one solution to the glitch problem. (c) Adding a redundant term to the circuit solves the glitch problem, but makes the circuit untestable. (d) A TEST input can be added to facilitate testing of the deglitched circuit. (e) In the case of a counter, a redundant term can't be used to eliminate glitches, so a register is added. (f) A typical PAL flip-flop is a D-type device, so must be reminded at each clock transition of its state following the clock.

synchronous machine. The current state and input conditions determine the next state and then the SMC register inputs. Thus, a metastability condition that occurs after the clock and stabilizes before the next clock can't hurt you (famous last words). Don't do this step if the asynchronous input affects two or more register stages. Each stage may react differently, and the SMC could assume an erroneous state.

GLITCHES AND TESTABILITY

Glitches and testability are closely related and difficult to separate. A cure for glitches is redundancy. However, redundancy makes the circuit untestable, a paradox.

Glitches are spikes in the output of combinational logic caused by "unbalanced" logic. The unbalance usually comes from the inverter required to complement a signal. Thus, the complement signal has a slight delay in time. When a complement signal is combined with other in-time signals, a glitch can occur.

Testability is an important commercial production design practice. Test not as a working device, but as independent circuit elements. Test engineers don't care if the PAL works in situ, they want visibility to each PAL node. Obviously you can't probe the buried nodes of the typical AND-OR circuit, but you can infer a fault (stuck-on-zero/one) if there are no redundancies. For example, $ABC \# BC$ is redundant. The obvious simplification is BC , but if left unsimplified, you could not determine if AND gate ABC was stuck-on-zero.

As an example of glitches and testability, say you have a circuit with the following equation: $A \& B \# !B \& C$ (see Figure 2a). If A, B, and C are ON, and B switches to OFF, the OR output should be constant-ON. However, a glitch can potentially exist. $!B$ is delayed by an inverter; thus, the OR input ($A \& B$) may terminate before $!B \& C$ takes over.

How do you get rid of glitches? Redundancy. The original expression of four terms nicely simplifies into

two terms that do not overlap on the map (see the Veitch diagram, Figure 2b). Add a redundant term, A&C (see Figure 2c), which will cure the glitch, but will render the circuit untestable. The redundant AND gate (A&C) can't be tested for stuck-on-zero. Thus, you can't be sure the redundant gate is actually working on delivery even if the circuit works in situ.

To make this circuit testable, a TEST input is added (see Figure 2d). The conversion of the two original gates to three input gates does not cost anything. The extra gate inputs are available, but the primary cost is a TEST input pin. Sometimes an extra pin can be very costly, because another PAL is required to complete the logic.

I describe the above glitch fix as partial in nature. It will fix the condition described, but what if the circuit was connected to a 3-bit binary counter? The Veitch diagram in Figure 2b shows that the glitch fix covered a transition from binary 7 to 5, but what about the transition from binary 3 to 4? The transition is exactly the same.

If the circuit decodes a counter output, the glitch can't be "covered" by a redundant term. The PAL counter glitch fix is a registered output (Figure 2e). If the PAL containing the counter is registered, the registered output is a no-cost option and is testable. The registered glitch fix worked because I had a priori knowledge of the circuit input state changes: a binary count. The same was true for the redundant term glitch fix; it was designed to cover the 7-to-5 transition because I knew that transition would occur.

If the circuit input transitions are purely random, no glitch fix is complete. The registered output also demonstrates an important PAL design consideration. The circuit output is ON for counts 3, 4, 5, and 7. The register flip-flop inputs are counts 2, 3, 4, and 6 (see the Veitch diagram in Figure 2f). But why 3? Shouldn't the flip-flop remember to remain set during count 4? No, the typical PAL flip-flop is a D-input device, so it must be reminded at each clock transition of its state following the clock.

Are all glitches bad? The answer depends on what you are doing with the signal. In synchronous or pipelined logic, glitches occur just after the clock, and stabilize before the next clock-no problem. Synchronous logic does cure glitches, but it does not eliminate them. In asynchronous logic, if the signal is a following stage clock, you would be surprised how small a glitch will clock a flip-flop (metastability never works in your favor). In this case, you'd better examine your logic for the possibility of a glitch. Are all circuits completely testable? No, but test engineers shoot for as close to 100% as economically feasible.

ILLEGAL STATES

An SMC must be completely controlled. Illegal states really mean unused or unassigned states. An SMC must have an initialization or reset state, and should have no illegal states. This rule means all 2^n states must be assigned even if they are not used.

In the SMC design I describe here, all states are assigned. Unused states

OPTIMIZE YOUR MCU PROGRAM DEVELOPMENT !

INTEGRATE THE POWER OF

- ◆ Editors ◆ Cross Assemblers ◆ Disassemblers ◆
- ◆ Cross Compilers ◆ Data Conversion Utilities ◆
- Simulators • Serial Communications ◆

ARMADILLO +™

A Unique, Universal Development /
Communications Environment Supporting :

- ◆ All families of cross-assemblers and compilers.
- ◆ Communications with your target CPU.
- ◆ User definable utilities menu.
- ◆ Pull-down menus with mouse or keyboard control.
- ◆ IBM PC or compatible.

Now you can EDIT, ASSEMBLE, UPLOAD,
DEBUG, and MORE, all from within ONE, FAST,
EASY-TO-USE MENU DRIVEN ENVIRONMENT !

\$99.00 + \$2.00 P/H

TO ORDER CALL (804) 479-3893

LIFE FORCE TECHNOLOGY

5477 RUTLEDGE RD., VB. BEACH, VR. 23464

Cross-Assemblers from \$50.00

Simulators from \$100.00

Cross-Disassemblers from \$100.00

Developer Packages

from \$200.00 (a \$50.00 Savings)

Make Programming Easy

Our Macro Cross-assemblers are easy to use. With powerful conditional assembly and unlimited include files.

Get It Debugged -- FAST

Don't wait until the hardware is finished. Debug your software with our Simulators.

Recover Lost Source!

Our line of disassemblers can help you recreate the original assembly language source.

Thousands Of Satisfied Customers Worldwide

PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

Processors

Intel 8048	RCA 1802, 05	Intel 8051	Motorola 6805
Motorola 6800	Motorola 6801	Motorola 68HC11	WDC 65C02
Hitachi 6301	Motorola 6809	MOS Tech 6502	NSC 800
Rockwell 65C02	Intel 8080, 85	Zilog Z80	
Hitachi HD64180	Mot. 68k, 8, 10	Intel 8096, 196kc	

New
Zilog Z8
Zilog Super 8
• All products
require an IBM PC
or compatible.

For Information Or To Order Call:

PseudoCorp

716 Thimble Shoals Blvd., Suite E

Newport News, VA 23606

(804) 873-1947

FAX: (804) 873-2154

#135

cause a return to the initialization state. Thus, if the SMC arrives at an unassigned state (via a transient), it will reset by returning to the initialization state. In the SMC designs described, state 0 is always the initial state. In the **PAL16R8** family, there are no synchronous or asynchronous register preset or clear inputs, so system reset is a term in state expressions. The system reset is normally both a power-on reset [lasting until power is stable), and a manual reset function.

SIMPLE DESIGNS

To start you designing with **PALs**, let me quickly discuss "rat" logic, the "glue" that interconnects your **MSI/LSI** circuits. Moving rat logic to a **PAL** will reduce the number of chips, the number of logic stages, and the associated logic delay. This logic "flattening" allows the use of **PALs** that are not as fast as the logic elements being replaced.

To convert the logic, write the logic equations (as is), and let the logic compiler do its thing. The compiler will find the redundancies and simplify the logic. One note: declaring intermediate output **terms** as intermediate variables is necessary. If this declaration is not made, the compiler will decode the intermediate output terms and connect them to output pins. Then the compiler will feed the intermediate outputs back into the **AND** matrix, causing reconvergent logic. To convert the **intermediate** variables to outputs, just equate them to output pins.

Another way to convert the logic is to "feed" a section of your schematic into the logic compiler. A number of compilers will accept schematics, although usually the schematic has to be redrawn using special symbols.

Many times the first (or second, or third, etc.) compile will terminate with an error, usually due to an excessive number of product terms. This occurrence illustrates my number one logic compiler selection requirement: the compiler must list the compiled (and simplified) logic equations as long as there are no

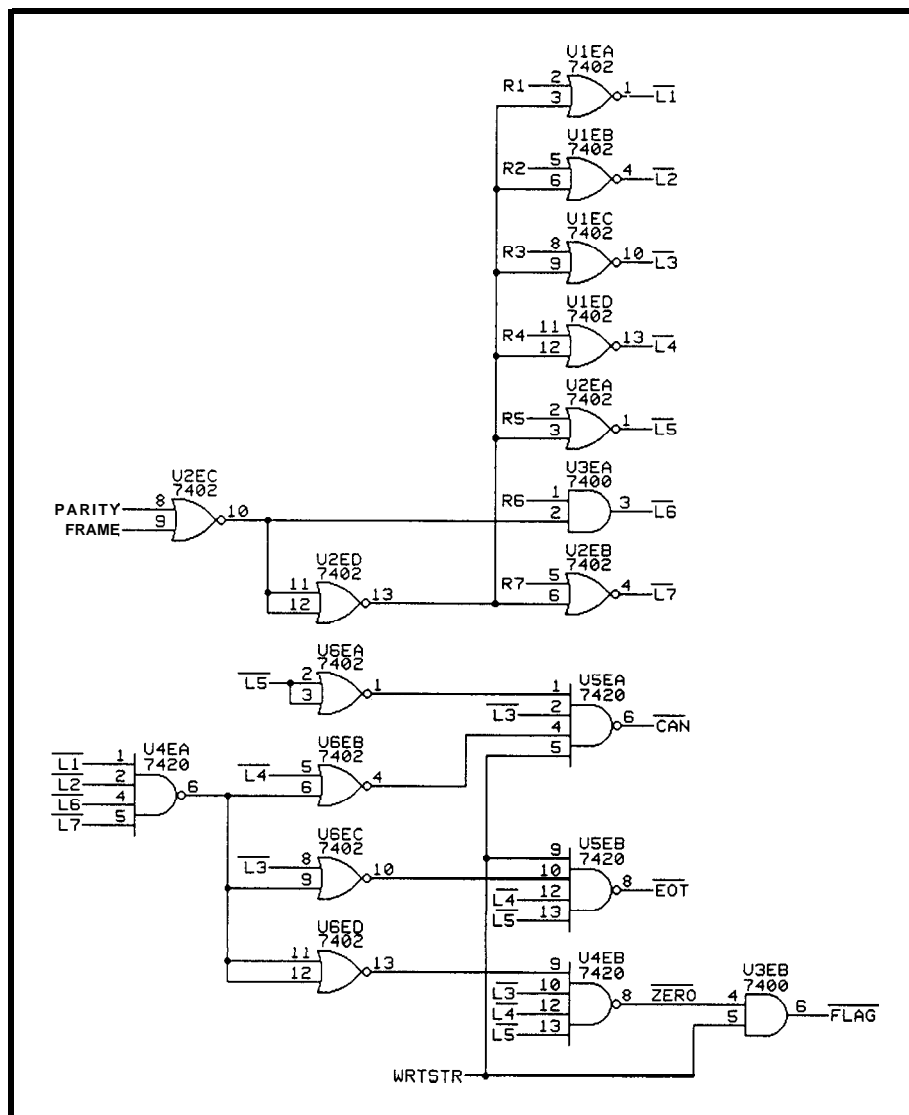


Figure 3—Using a **PAL** to replace combinational or "rat" logic can usually reduce chip count significantly.

syntax errors. Some compilers just terminate on a product term overflow with no indication of the problem. With the equation listing, you can find the problem and fix it by revising the logic or selecting another **PAL**.

The most common logic conversion problem is an excessive number of product terms. The required logic is a large positive **AND**. Most simple **PAL** outputs invert and have no more than eight **OR** terms, limiting a positive **AND** output to eight terms. Often you can accept a negative output and invert in a following logic element. However, if you can't see the problem in a compiler listing, you might start a major redesign. New universal **PALs** with programmable output macrocells have reduced the number of times this problem occurs.

Some combinational logic will never fit in a **PAL**. A binary full adder with a fast carry (74283) or a magnitude comparator (7485) are examples. The number of minterms expand exponentially as the number of stages increase—that is, if you try to flatten the logic to a sum-of-products to minimize propagation delay.

RAT LOGIC

An example rat logic conversion is shown in Figure 3. I used a **22V10** in the example, which may be overkill, but it had the pin count. The **22V10** is a 24-pin **PAL** with ten macrocell outputs, combinational or registered, with programmable output sense. The circuit was used to accept the parallel output of a **UART**, decode certain characters, swallow null codes, and

"It is not Wisdom to be only Wise...." **

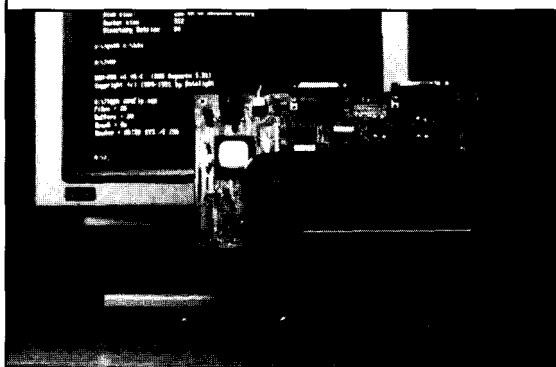
Lessons from my experiences with State Machine Controllers

1. Bubbles are not necessary when designing an SMC. The logic compiler state machine syntax eliminates graphical steps, although you can keep them if you wish.
2. Numbering states consecutively may cause trouble. The resultant SMC logic may be too complex for the target device, so make sure you examine states for events or sequences in common, such as output, test, decision, jump, or loop. Align the similar events (minor states), and find a common binary increment for each major state (4, 8, 16, etc.).
3. You must have an initial state and account for unused states--enough said.
4. An SMC must be programmed much like a computer. For example, if you are looking for the trailing edge of a positive pulse, test for pulse presence, then test for pulse absence. On the first test pass for absence, you have found the trailing edge (**as close as you will get to it with an SMC**).
5. To synchronize an asynchronous SMC input pulse, the maximum SMC clock period must be equal to or less than the input pulse period. This statement is true assuming an SMC clock sample just misses an input pulse and the input pulse duration is just long enough to be sampled by the following SMC clock.
6. The SMC takes a clock period to sample a **synchronized** input and react. Thus, the bare minimum SMC clock reaction time is twice the minimum asynchronous input pulse period.
7. Due to SMC reaction time, the SMC clock is often ten times faster than the input clock.
8. Windows save a lot of time and money. I don't mean Microsoft Windows, but the windows in the PALs. Most **PALs** have an EPROM-like erasable cousin (sometimes it is EEPROM-like only without a window). Simulation time is not worth an equivalent amount of time in situ, **so try** it in windows before you blow fuses.

• 'George Santayana, *O World, Thou Choosetest Not*, 1894.

DOS Where You Want It

-Runs on Disk, ROM, FLASH



Already in widespread use on a variety of platforms, ROM-DOS has proven its dependability.

Develop your programs on a standard PC and run them in ROM with ROM-DOS, the flexible, affordable operating system that was designed for embedded applications. Compatible with MS-DOS 3.3 1, ROM-DOS boots and runs from ROM, disk, or FLASH memory. Features include: small size (low as 36K); runs executable files from ROM; FLASH disk support; remote disk and multimedia support; and a powerful utility suite. Confirmed compatible by VeriTest, an independent evaluator, ROM-DOS is already in widespread use on multiple system types. Developers Kit-which contains all the tools and utilities needed to configure ROM-DOS to your system, and includes license for 20 copies-costs only \$495. ROM-DOS royalty fees depend on quantity purchased, and range from \$3 to \$25 per copy.

Free Demo Disk! Call Today Toll-Free 1-800-221-6630

Datalight

17455 68th Ave NE, Suite 304, Bothell, WA 98011, USA • 206-486-8086 • fax 206-486-0253

convert the output if there was a parity or a framing error.

The equations, shown in Listing 1, are written exactly as shown in the schematic. The intermediate outputs, !L1-!L7, were made intermediate variables !I1-!I7. This arrangement keeps the compiler from developing reconvergent paths for CAN, EOT, and FLAG. In the first input file, I assigned the I/O pins in drawing order. In the final CUPL input listing, I switched the pin assignments of FLAG and EOT because of excessive product terms in FLAG for the original output pin.

The equation output listing shows positive outputs. The 22V10 is instructed to select the inverting macrocell output in the JEDEC file. Notice the flattening: seven levels to two. The slowest 22V10 available will beat the 70- to 100-ns standard TTL logic delay.

If converting microprocessor memory enables (e.g., PROM, RAM, memory-mapped peripherals), use the logic compiler range operators (if available). The range operators allow an enable to be described as "enable from address XXX X to address Y Y Y Y." The CUPL notation is [XXXX. .YYYY]. Therefore, if your logic compiler has range operators, go back to the original design and don't translate the logic.

STATE MACHINE CONTROLLERS

Programmable logic and logic compilers have really simplified SMC design. I previously used both counter- and counter/memory-based SMCs. Counter-based SMCs are difficult to design if they exceed four stages. The equations become difficult to write and simplify. Besides, you have to be right the first time, or debugging means wiring changes. Counter/memory SMCs are easier to design. Debugging them is also easier: just reprogram the memory. The problem is a trash can full of dead "bugs" if you use bipolar PROMs. SMCs come in two flavors: Moore and Mealy. Moore SMC outputs are registered while Mealy includes combinatorial outputs. Thus, the Moore outputs only change on the clock, while Mealy outputs may change asynchronously. The SMC I describe here is a Moore machine.

Listing 1--The *schematic in Figure 3 can be converted directly to PAL equations.*

```

****RATLOGIC****
DEVICE P22V10;

PIN 1 = R1;
PIN 2 = R2;
PIN 3 = R3;
PIN 4 = R4;
PIN 5 = R5;
PIN 6 = R6;
PIN 7 = R7;
PIN 8 = PARITY;
PIN 9 = FRAME;
PIN 10 = WRTSTR;
PIN 11 = NC12;
PIN 13 = NC13;
PIN 14 = !L1;
PIN 15 = !L2;
PIN 16 = !L3;
PIN 17 = !L4;
PIN 18 = !L5;
PIN 19 = !L6;
PIN 20 = !L7;
PIN 21 = !CAN;
PIN 22 = !FLAG;
PIN 23 = !EOT;

/*INTERMEDIATE VARIABLES*/
!I1 = !(R1 # !(PARITY # FRAME));
!I2 = !(R2 # !(PARITY # FRAME));
!I3 = !(R3 # !(PARITY # FRAME));
!I4 = !(R4 # !(PARITY # FRAME));
!I5 = !(R5 # !(PARITY # FRAME));
!I6 = !(R6 & !(PARITY # FRAME));
!I7 = !(R7 # !(PARITY # FRAME));

/*OUTPUTS*/
!L1 = !I1;
!L2 = !I2;
!L3 = !I3;
!L4 = !I4;
!L5 = !I5;
!L6 = !I6;
!L7 = !I7;

!CAN = WRTSTR & !I5 & !I3 & !(I4 # !(I7 & !I6 & !I2 & !I1));
!EOT = WRTSTR & !I4 & !I5 & !(I3 # !(I7 & !I6 & !I2 & !I1));
!FLAG = !(WRTSTR & !(I7 & !I6 & !I2 & !I1) & !I5 & !I4 & !I3));

****COMPILER OUTPUT****
CAN => !FRAME&!PARITY&!R1&!R2&!R3&R4&R5&!R6&!R7&WRTSTR
EOT => !FRAME&!PARITY&!R1&!R2&R3&!R4&!R5&!R6&!R7&WRTSTR
FLAG => R7&WRTSTR#R6&WRTSTR#R5&WRTSTR#R4&WRTSTR#R3&WRTSTR
      #R2&WRTSTR#R1&WRTSTR#PARITY&WRTSTR#FRAME&WRTSTR
I1 => R1 # PARITY # FRAME
I2 => R2 # PARITY # FRAME
I3 => R3 # PARITY # FRAME
I4 => R4 # PARITY # FRAME
I5 => R5 # PARITY # FRAME
I6 => !FRAME & !PARITY & R6
I7 => R7 # PARITY # FRAME
L1 => R1 # PARITY # FRAME
L2 => R2 # PARITY # FRAME
L3 => R3 # PARITY # FRAME
L4 => R4 # PARITY # FRAME
L5 => R5 # PARITY # FRAME
L6 => !FRAME & !PARITY & R6
L7 => R7 # PARITY # FRAME

```



REAL-TIME MULTITASKING KERNEL

8086/ 88, 80x86/88 80386
Z80, 64 180, 8080/85 68000/ 1 0r20

- Fast, reliable operation
- Compact and ROMable
- PC peripheral support
- DOS file access
- C language support
- Preemptive scheduler
- Time slicing available
- Configuration Builder
- Complete documentation
- Intertask messages
- Message exchanges
- Dynamic operations
 - task create/delete
 - task priorities
 - memory allocation
- Event Manager
- Semaphore Manager
- List Manager
- Insight™ Debugging Tool

THE BEST

Join over 1000 developers such as
IBM®, Xerox, Hewlett Packard,
Hayes, Hughes Aircraft and NASA.

CHOOSE AMX

The best low-cost, high-performance
real-time multitasking system
available today.

No Royalties
Source Code Included

Demo Disk and
Manual only **\$85 US** Call for prices for
AMX 86 **\$3000 US** other processors.
(Shipping/handling extra)

IBM is a registered trademark of IBM Corp
Z80 is a trademark of Zilog, Inc.
AMX, AMX 86, Insight are trademarks of
KADAK Products Ltd.

KADAK Products Ltd.

206-1847 West Broadway
Vancouver, B.C., Canada
V6J 1Y5



Telephone: (604) 734-2796
Fax: (604) 734-8114

#1

Listing 2—Complete state machines can be implemented inside a PAL, sometimes replacing the need for a microprocessor.

```

****IN STATE****
DEVICE- 22V10;

PIN 1  = CLOCK;
PIN 2  = EOM           /*END OF MESSAGE FLAG*/
PIN 3  = ADRO;         /*BUS ADDRESS*/
PIN 4  = ADR1;
PIN 5  = DEVSEL;
PIN 6  = SW0;          /*LSB's OF DEVICE ADDRESS*/
PIN 7  = SW1;
PIN 8  = NC8;
PIN 9  = NC9;
PIN 10 = !STROBE;      /*SYNCHRONIZED TO 20MHz CLOCK*/
PIN 11 = !RESET;
PIN 13 = NC13;
PIN 14 = !WRITE_COUNT
PIN 15 = !MSB_COUNT;
PIN 16 = !LSB_COUNT;  /*LOAD ACTION_COUNT*/
PIN 17 = !Q0;         /*STATE MACHINE COUNTER*/
PIN 18 = !Q1;
PIN 19 = !Q2;
PIN 20 = !Q3;
PIN 21 = !WRITE_DATA; /*WRITE DATA*/
PIN 22 = !MSB_DATA;   /*LOAD DATA*/
PIN 23 = !LSB_DATA;   /*LOAD DATA*/

/*END-OF-MESSAGE FLAG*/
STATUS = !EOM;
ADDRESS = !(ADRO $ SW0) & !(ADR1 $ SW1) & DEVSEL;
field COUNT = [Q0..3];
/*ASYNCHRONOUS INPUT SMC RESET*/
O0.ar = RESET;
Q1.ar = RESET;
Q2.ar = RESET;
Q3.ar = RESET;
/*ENABLE INPUT BUFFER OUTPUT ENABLE & WRITE DATA TO FIFO & COUNT*/
/*DATA BUS AND DATA BUS ENABLES ARE TRI_STATE*/
LSB_DATA = 'b'1; /*SET BINARY MODE*/
MSB_DATA = 'b'1;
LSB_DATA.oe = COUNT:'h'8; /*RESTORE HEX MODE*/
MSB_DATA.oe = COUNT:'h'A;
WRITE_DATA = COUNT:'h'8 # COUNT:'h'A;
/*ENABLE COUNTER OUTPUT & WRITE ACTUAL COUNT TO FIFO*/
LSB_COUNT = COUNT:'h'C;
MSB_COUNT = COUNT:'h'E;
WRITE_COUNT = COUNT:'h'C # COUNT:'h'E;
sequence COUNT {
/*SEARCH FOR END OF MESSAGE FLAG*/
present 0 if !STROBE next 0: if STROBE next 1:
present 1 if STROBE next 1: if !STROBE next 2:
present 2 if ADDRESS & EOM next 4:
if (!ADDRESS # !EOM) next 0:
present 3 next 0;
/*FOUND EOM LOAD MESSAGE & COUNT*/
present 4 if !STROBE next 4: if STROBE next 5:
present 5 if STROBE next 5: if !STROBE next 6:
present 6 if !ADDRESS next 4:
if ADDRESS & !EOM next 8;
if ADDRESS & EOM next C;
present 7 next 0:
/*END OF MESSAGE STORE MESSAGE WORD COUNT*/
/*REMAIN IN LOAD MESSAGE MODE BY RETURNING TO STATE 4*/
present 8 next 9: /*LSBYTE DATA TO FIFO*/
present 9 next A;
present A next 4: /*MSBYTE DATA TO FIFO*/
present B next 0;
present C next D: /*LSBYTE COUNT TO FIFO*/

```

(continued)

Most of my SMC designs don't look like the elevator or traffic light controls I've seen at PAL design seminars. Many of them were essentially conversions of counter/memory SMCs to PALs to save board space.

INSTATE

I implemented this particular SMC using, again, the 22V10 (see Listing 2). One improved feature of the 22V10 is that the number of product terms has been increased over older parts, allowing more complex designs. Remember this addition if you run out of product terms; their number varies from 8 to 16. Thus, a little moving of expressions among macrocells may allow a complex design to fit. Also, each 22V10 output register stage has programmable asynchronous preset or reset inputs, making SMC initialization easy.

In the SMC, an external bus data clock-STROBE (2.5 MHz)—is sampled by the SMC clock. External to the SMC, I synchronized STROBE to the SMC clock (20 MHz) because metastability is ever present. The SMC looks for STROBE, and if it is found, the SMC waits for the trailing edge of STROBE to perform its functions. The external bus contains data, device addresses, and flags at STROBE time.

Initially the SMC is not synchronized to data bus messages. The SMC searches for its own address and then examines the data and flags for an EOM, which indicates the completion of a message. Following EOM, the SMC is synchronized to the bus message structure and waits for the start of the next message. The major states are

0-Three counts [0..2]. From RESET, search for STROBE. If it is present, wait for the trailing edge of STROBE. Following the trailing edge, if ADDRESS and EOM are present, then move to state 1. If they are not, then return to count 0.

1-Three counts [4..6]. EOM detected, now search for a complete message. Again search for STROBE. If it is present, then wait until the trailing edge. After the trailing edge of STROBE, complete one of the following steps:

- if ! ADDRESS, then data was for another device, return to count 4.

Listing 2—continued

```
present D next E;
present E next 4:/*MSBYTE COUNT TO FIFO*/
present F next 0;)

****COMPILER OUTPUT****

ADDRESS =>
    !ADRO & ADRI & DEVSEL & !SWO & SW1
    # ADRO a ADRI a DEVSEL a swo a SW1
    # !ADRO & !ADRI & DEVSEL & !SWO & !SW1
    # ADRO & !ADRI & DEVSEL & SWO & !SW1
LSB_COUNT => !Q0 a !Q1 a Q2 a 0 3
LSB_DATA => 1
MSB_DATA.oe => !Q0 a !Q1 a !Q2 a a3
MSB_COUNT => !Q0 & 01 & 02 & 03
MSB_DATA => 1
MSB_DATA.oe => !Q0 a Q1 a !Q2 a Q3
Q0.ar => RESET
Q0.d => !Q0 a !Q1 a a3 # !Q1 a !Q3 a STROBE
Q1.ar => RESET
Q1.d => 00 a !Q1 a a3 # 00 a !Q1 a !Q3 a !STROBE
Q2.ar => RESET
Q2.d => !Q0 a Q1 a Q3
    # !ADRO a ADRI a DEVSEL a EOM & !Q0 a a1 a !Q3 a !SWO a swl
    # ADRO & ADRI & DEVSEL & EOM & !Q0 & 01 & !Q3 & SWO & SW1
    # ADRO & !ADRI & DEVSEL & EOM & !Q0 & 01 & !Q3 & SWO & !SW1
    # !ADRO a !ADRI a DEVSEL a EOM a !Q0 a Q1 a !Q3 a !SWO a !SW1
    # !ADRO a !Q0 a Q1 a Q2 a !Q3 a swo
    # !Q1 a 02
    # !DEVSEL a !Q0 a Q1 a a2 a !Q3
    # !ADRI a !Q0 a Q1 a Q2 a !Q3 a swl
    # ADRI a !Q0 a a1 a Q2 a !Q3 a !SW1
    # ADRO a !Q0 a 01 a Q2 a !Q3 a !SWO
Q3.ar => RESET
Q3.d => !Q1 a a3
    # !ADRO & ADRI & DEVSEL & !Q0 & Q1 & 02 & !Q3 & !SWO & SW1
    # ADRO & ADRI & DEVSEL & !Q0 & Q1 & 02 & !Q3 & SWO & SW1
    # ADRO a !ADRI a DEVSEL a !Q0 a a1 a a2 a !Q3 a swo a !SW1
    # !ADRO & !ADRI & DEVSEL & !Q0 & 01 & 02 & !Q3 & !SWO & !SW1
STATUS => !EOM
WRITE_COUNT => !00 & 02 & 03
WRITE_DATA => !00 & !Q2 & 03
LSB_COUNT.oe => 1
MSB_COUNT.oe => 1
00.oe => 1
01.oe => 1
Q2.oe => 1
Q3.oe => 1
WRITE_COUNT.oe => 1
WRITE_DATA.oe => 1
```

• if ADDRESS and ! EOM, then go to state 2.

• if ADDRESS and EOM, then go to state 3.

2-Three counts [8..A]. Enable LSB_DATA(count 8) and MSB_DATA(count A) on input data bus. Latch data input bus register with WRITE_DATA strobe. Return to state 1 to search for next data byte or EOM

3-Three counts [C..E]. Enable LSB_COUNT(count A) and MSB_COUNT(count E) of message word count. Latch

count with WRITE_COUNT strobe. Return to state 1 to search for next message start.

Two methods are used to generate the SMC outputs. LSB_DATA and MSB_DATA are connected to tristate control lines. The logic input to the output buffer is always true. The output signal is controlled by the output buffer tristate enable. The other SMC outputs are always enabled, and the logic input controls the output level.

In assigning the SMC state counts, I realized there were four major states. Each major state had three counts. Instead of assigning 12 consecutive counts, the states were assigned in groups of four counts, based on a form of intuitive binary reasoning [and my previous experiences with SMC compile failures-too complex). When a PAL SMC compile fails, an inspection of the compiled logic will often lead to a design change that will better fit the logic to the PAL. For instance, in an SMC, make similar events occur at similar binary counts as in the above situation. This design practice also helps to simplify the final output logic.

The unused counts 3, 7, 11, and 15 contain a reset statement, and entry into them is a sequence error. The function performed is arbitrary and depends on the designer's SMC requirements. You could initialize, continue, or reset. However, make sure all possible states are described, because there are no illegal states allowed.

THE NEXT STEP

I have also used these simple PAL design tools to program the newer, more complex field programmable gate arrays (FPGAs). I recently fit an old TTL design into a single FPGA by writing the equations into a 22VIO. (I knew the equations would never fit, I just wanted the simplified form.) Then, I took the compiler listing and "word processed" an equation listing for the FPGA. The new listing was input to a partial compiler provided by the FPGA manufacturer to see if it would "fit," which it did. Now all that is required to complete the project is the financing. 📁

Charles Conkling has spent the last 30 years designing logic and computer systems in both military aircraft and commercial applications.

416 Very Useful
417 Moderately Useful
418 Not Useful

SOURCES

Advanced Micro Devices
(Monolithic Memories, Inc.)
90 1 Thompson Pl.
Sunnyvale, CA 94088

Cypress Semiconductor
390 1 North First St.
San Jose, CA 95134

Data I/O
10525 Willows Rd. NE
Redmond, WA 98073-9746

Intel Corporation
3065 Bowers Ave.
Santa Clara, CA 9505 1

JDR Microdevices
22.33 Samaritan Dr.
San Jose, CA 95124

Logical Devices, Inc.
1201 NW 65th Pl.
Ft. Lauderdale, FL 33309

Signetics Corporation
811E. Arques Ave.
Sunnyvale, CA 94088-3409

Texas Instruments, Inc.
P.O. Box #655303
Dallas, TX 75265

Affordable 8031 Development

Single Board Computers, Assemblers, Compilers, Simulators, and EPROM Emulators

Control-R Series, Single Board Computers

Two models of Control-R series computers make prototyping, one of a kind products, or small production runs easy and economical. Both feature RS232 compatible serial ports, single 5 volt supply operation, and direct access to Ports 1 and 3 of the 8031. Additional features are as follows:

Control-R Model 1 \$49.95

Fully populated board with I/O header for Ports 1 and 3, serial port, and 8K EPROM socket. 3.0" x 4.0"

Control-R Model 2 \$79.95

Same features as the Control-R 1 plus 8K of SRAM and expansion bus with data, address, RST, INT1, WR, RD, PSEN, ALE and T1. 3.5" x 4.5"

Software and Hardware Development Tools

Control-C 8031 Cross-Compiler \$200.00

The Control-C 8031 cross-compiler is a full featured K&R style C development system available at an affordable price. Optimized for embedded system use, it will produce ROMable code for any 8051 based system including designs using only the 128 bytes of internal RAM. Package includes compiler, pre-processor, assembler, simulator, printed documentation and complete library source code. Requires IBM PC or compatible. 5.25", 360K disk.

PROMulator 256 \$189.95

An EPROM emulator lets you avoid "Bum and Test" development cycles. In circuit emulation of 2K-32K 27xx series EPROMs. ABS Plastic case. Assembled or compiled code is downloaded directly to the target hardware.

Cottage Resources Corporation

Suite 151, 10271 South 1300 East
Sandy, Utah 84094

VISA/MC, COD. Call to Order: (801) 268-2875

DEPARTMENTS

70

Firmware Furnace

80

From the Bench

86

Silicon Update

92

Practical Algorithms

98

ConnecTime

HCS II War Stories and I/O Links

Four pairs of eyes trained to pay attention to detail couldn't possibly miss an obvious blunder, could they? Find out as Ed describes a debugging session that took far too long plus more HCS II tricks.

FIRMWARE FURNACE

Ed Nisley



OW would you handle this phone call from Steve

Ciarcia: "I just added a DIO-Link and my HCS II network died. Ken and Jeff are on their way over. Care to join us?"

My response? I grabbed source code, laptop, Jensen tool kit, and left with the doorknob smokin'. . . .

We started at the main HCS II panel, which has mutated only slightly from the picture shown *on* page 25 of *Circuit Cellar INK*, issue #26. Steve showed us the four-wire cable from the offending DIO-Link. We verified that the system worked correctly without the cable, so the Supervisory Controller (SC) software and node firmware seemed to be OK.

THE LAYOUT

Steve led us along the network cable, which terminated in a junction box near the garage door. The RS-485 network signals on the red/green wires (R/G) joined the orange/green wires (O/G) of a 14-conductor cable that was already being used for other functions. The black/white pair went through unchanged to the larger cable.

The 14-wire cable penetrated the concrete foundation, entered a buried conduit, crossed the driveway, and ended in a junction box inside Steve's detached, two-car garage. There, the white lead stopped at the barrier strip while the black wire and the R/G pair joined another cable that crossed the garage and exited to another conduit underneath the parking pad. That conduit led to a junction box inside his four-car garage (hey, he likes cars, what can I say?), where a final run of

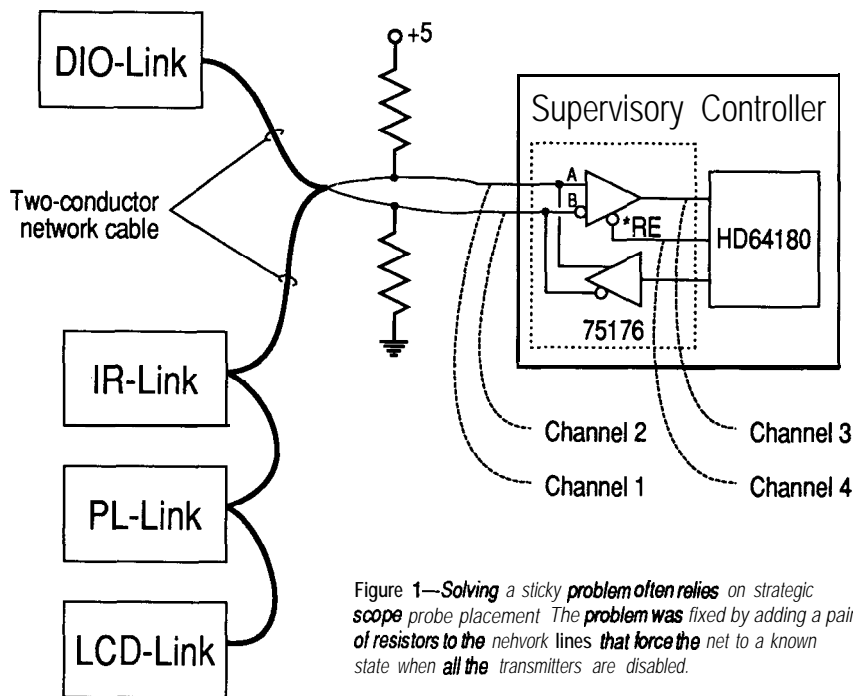


Figure 1—Solving a sticky problem often relies on strategic scope probe placement. The problem was fixed by adding a pair of resistors to the network lines that force the net to a known state when all the transmitters are disabled.

four-wire cable crossed the garage and met the DIO-Link board, which was powered by a wall-wart transformer.

About 300 feet of multiwire cable ran with the black (common) lead continuous from the HCS II board to the DIO-Link. We verified that there was little ground offset, which Steve expected because all structures are tied to a common lightning rod ground system with half-inch braided copper cables.

We noticed that, although the **HOST** status display showed continu-

ous network data errors, the DIO-Link was actually functional. Steve had set up the SC program to flip a DIO-Link output bit at regular intervals, and the LED on that pin was blinking merrily away.

When we terminated the RS-485 network at the DIO-Link board, the data errors continued, but we knew the DIO-Link stopped receiving messages because the LED stopped blinking. Removing the terminator restored the previous behavior. During all of this time, the LCD-Link at the

main panel was displaying status messages!

With the laptop hitched to the RS-485 network at the SC, we found the network was carrying perfectly valid messages in both directions, much to our surprise. The DIO-Link stopped responding when we terminated it, but the SC continued to send messages to other nodes after the DIO-Link message timed out.

We discovered that the DIO-Link would work when connected to just the green network wire, but failed when only using the red wire. About 30 seconds later, four pairs of eyes saw what you've probably already noticed: in that first junction box the R/G pair connected to an O/G pair, but the second box mated two R/G pairs. The orange wire from the SC end of the network wasn't connected to anything at the DIO-Link end. That poor DIO-Link board managed to receive messages with only one network input and no ground at all!

We ribbed Steve a bit as he wielded the wire strippers, patted ourselves on our collective back, and returned to the main panel.

The **HOST** status display showed continuous network data errors.

THE EVIDENCE

We connected a scope to the network, with Channels 1 and 2 in differential mode across the RS-485 data lines, Channel 3 showing the SC's

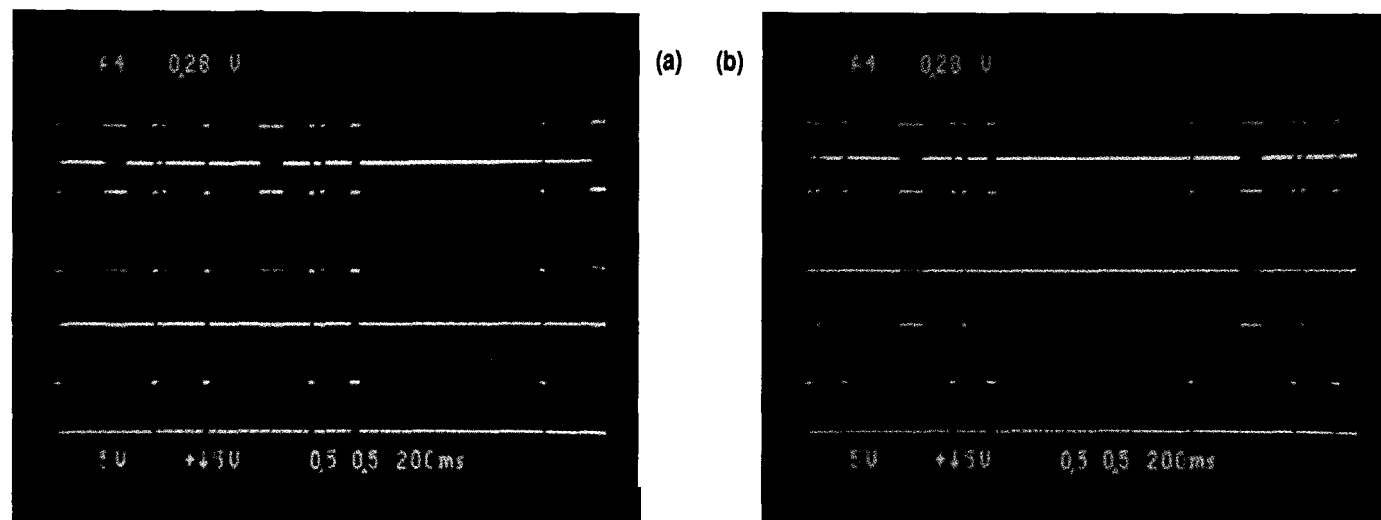


Photo 1—The top trace shows Channel 1—Channel 2; the middle trace shows Channel 3; and the bottom trace shows Channel 4. The scope is triggered on Channel 4. a) The problem can be seen in the middle trace: the receiver output floats to the wrong polarity when the network is inactive. b) When the resistors shown in Figure 1 are added, the output is pulled high during inactivity as it should be.

receiver pin, and Channel 4 showing the SC's Receiver Enable line (● RE). We triggered from Channel 4 and sat back to think about the situation while watching both the scope and the laptop's ASCII display.

Figure 1 shows the connections and Photo 1a shows what happens with the DIO-Link cable connected.

These photos are from a network simulation because I wasn't set up for scope photography while we were debugging. However, you see pretty much what we saw: the signals are clean and clear, the laptop showed no data errors, and the SC indicated serious trouble.

When we removed the DIO-Link and its termination from the network, the 300 feet of empty cable acted as a pretty good 60-Hz antenna. However, with the network connected as designed, there were only a few dozen millivolts of AC interference, which was not enough to cause any glitches.

Receiver Enable signal (*RE)	Differential input voltage ($V_{ID} = V_A - V_E$)	Receiver Output State
High	Don't care	High-Z
Low	$+200\text{ mV} \leq V_{ID}$	High
Low	$-200\text{ mV} < V_{ID} < +200\text{ mV}$	Indeterminate
Low	$V_{ID} \leq -200\text{ mV}$	Low

Figure 2—The 75176 W-485 transceiver converts a differential input voltage into a TTL-compatible output signal. This table shows the output state for various input voltages, which are measured at input A with respect to input B.

Photo 1b shows the situation with the DIO cable removed. Compare with Photo 1a and see if you don't get the same "Ah-ha!" we did after half an hour of discussion.

The idle state of the receiver should be, by definition, inactive. However, as you can see in Photo 1, the TTL level on that pin (the middle trace) is low when messages aren't in progress. The inactive state is high—see it!

A CASE OF BIAS

The problem turned out to be a 75 176 RS-485 transceiver specification

that, in many situations, doesn't make any difference. However, under the right conditions, it can be a killer.

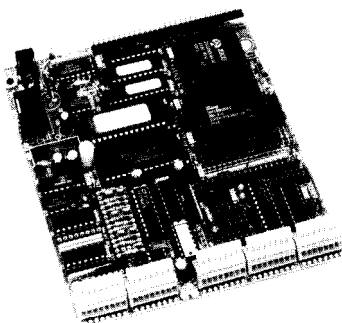
The receiver section of the 75 176 transceiver converts the differential voltage at the network pins into a TTL output signal. Figure 2 is a truth table showing how the data and ● RE pins determine the output value.

When the receiver is disabled by a logic-high input, the output pin floats in a high-impedance state regardless of what the network input lines are doing. The serial input pin on your CPU must have a pull-up resistor so the logic level remains inactive rather than drifting in the breeze. If you are using an 803 1, you can take advantage of the on-chip pull up, but the HD64180 must have an external pull-up resistor. A 10k resistor is fine.

Setting ● RE low enables the receiver so the output tracks the input. The RS-485 network uses differential signal levels; therefore, the second

MICROCONTROLLERS

- C Programmable
- Data Acquisition
- Control / Test
- Excellent Support
- From \$159 Qty 1
- New Keyboard Display Modules



Use our **Little Giant™** and **Tiny Giant™** miniature controllers to computerize your product, plant or test department. Features built-in power supply, digital I/O to 48t lines, serial I/O (RS232 / RS485), A/D converters to 20 bits, solenoid drivers, time of day clock, battery backed memory, watchdog, field wiring connectors, up to 8 X 40 LCD with graphics, and more! Our \$195 interactive **Dynamic C™** makes serious software development easy. You're only one phone call away from a total solution.

Z-World Engineering

1724 Picasso Ave., Davis, CA 95616
(916) 757-3737 Fax: (916) 753-5141
 Automatic Fax: (916) 753-0618
 (Call from your fax, request catalog #18)

HARD DRIVE ENCYCLOPEDIA

The Hard Drive Encyclopedia is a reference work covering PC-compatible hard drives, with a companion diskette of utilities. Now you can learn how drives are built, how they work, and how they are interfaced to the computer. And you can see the drive type tables in your computer's BIOS with the utilities on the disk.

ST506, ESDI, SCSI, and IDE specs are here, along with five other interfaces. It has hard drive physical and electrical characteristics, logical encoding schemes, and file formats. Listings contain over 1600 models, so you can install a drive without a spec sheet.

This big new book, with its diskette, is only \$89.00 plus shipping. We accept credit cards, COD orders, or company P.O.s. And tell us where you saw this ad when you order and we'll include a FREE copy of our famous XT-AT Handbook!

ANNABOOKS

15010 Avenue of Science, Suite 101
 San Diego, CA 92728

1-800-462-1042 619-673-0870 619-673-7432 FAX

See us at the Embedded Systems Conference-Booth #407

#14

Listing1--The routines that convert M6242 clock registers to and from an ASCII timestamp string use this formatting string. Characters between \x80 and \x91 mark the location of specific registers so it is easy to change the timestamp layout for special applications.

```

/*--- Timestamp format
/* \x8- are M6242 registers
/* \x90 is location for 3-char fractional seconds
/* \x91 is "day of week" from following table
/* others are literals inserted as-is
char IOTimeFormat[32] =
"\x89\x88/\x87\x86/\x8B\x8A \x91 \x85\x84:\x83\x82:\x81\x80.\x90";

```

column of Figure 2 represents the voltage between the two input pins as measured with respect to Input B. You (or a noise source) can add equal voltages to both pins (in "common mode") as long as the total remains within the 75 176's specification.

There is no question what the output will be when the differential input voltage exceeds 200 mV, which it will whenever a transmitter is active. The 75 176 transmitter's differential output voltage exceeds 1.5 volts, so a considerable margin is

obviously present to handle line resistance and noise glitches.

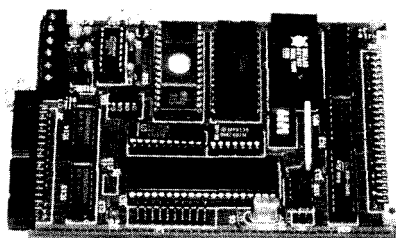
The problem occurs when all of the transmitters on the network are disabled, in which case the terminating resistors "pull" the two data lines together and reduce the differential input voltage to nearly zero. The actual signal will be determined by the driving impedance of any noise sources, which can produce virtually anything from a few millivolts to a few tens of millivolts of differential noise signal.

Our previous experience with RS-485 networks showed that, at least for the transceivers we had used, there was no problem as long as the firmware disabled the transmitters after the final stop bit. In that situation, the receiver output went high with the stop bit and remained high even though the differential input voltage fell below 200 mV. But, as Figure 2 shows quite clearly, the receiver's output voltage is simply not defined and can be either high or low.

As it turns out, different brands of 75 176 transceivers react differently to the same input conditions. Some remain high while others go low, and there is no way to distinguish the two. Steve's rather hostile installation certainly picked out that fault, though!

The solution we settled on is shown in Figure 1. A pair of resistors applies a bias to the terminating resistors when the transmitters are disabled to ensure the receivers never see a voltage that results in an indeterminate output. Each network needs only one pair of bias resistors.

ProControl!



Introducing the Most Expandable 3 1/2 Board Available

MCU-31/2 from \$149.95

On Board Options Include:

- * 16Channels 10 bit A/D -14µs w/ S/H
- * Dallas 1287 RTC
- * Flexible mem config's: RAM -8/32KB
ROM -8/16/32/64 KB
- * RS-232 or 485 -jumper selectable
- * Watchdog timer w/ jumper sel, reset source
- * Pre decoded external bus for very easy user interface -
directly compatible with other ADS boards
(see back issues of Circuit Cellar INK)

More I/O modules are available.
Call for our FREE catalogue today!

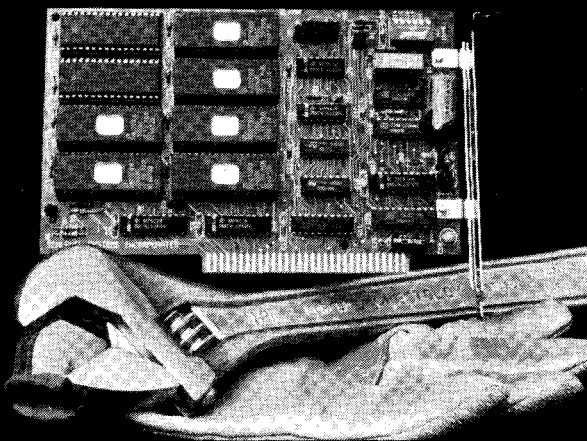
(404) 352-4788

**ADS Advanced
Design
Solutions**

1920 Moores Mill Road
Atlanta, GA 30318

SOLID STATE DISK

- BOOTABLE
- VERSATILE
- SIMPLE
- COMPATIBLE
- DURABLE
- FAST
- RELIABLE
- ANOTHER SEALEVEL/ANNABOOKS INNOVATION



SEALEVEL
COMMUNICATIONS & I/O

SEALEVELSYSTEMS INC.
PO BOX 830
LIBERTY, SC 29657
(803) 843-4343

Determining the resistor values is a straightforward application of Ohm's law as long as you remember to include the loading due to the 32 receivers allowed by the RS-485 specification. The maximum value of the resistors must keep at least 200 mV across the terminators with no transmitters enabled. The minimum values cannot strain the transmitter's 60 mA current limit.

With two 120-ohm terminators, the bias resistors should be about 680 to 750 ohms. I've seen lower values in other sources, so you'll want to run the arithmetic for your own network.

In principle, the terminators should match the transmission line's characteristic impedance. In practice, I don't think many networks have a single, neat impedance value because of all the junctions and stubs, but 100 to 120 ohms is typical for most cables. The specification calls for a terminator at both ends of the longest cable run, so try to avoid "star" networks.

The system worked perfectly after we added a pair of bias resistors to the network. However, there was conspicuously less back-patting this time.

BACK TO THE FIRMWARE

After your HCS II network has a PL-Link driving your appliances, an IR-Link tracking people, and an LCD-Link (or two) showing the system status, what's left? Well, how about some ordinary I/O? That's what the ADIO- and DIO-Links are all about.

Over the past few years, you've seen enough analog and digital I/O in this column that I don't have to do it again, so I'll cover a few interesting details and leave the rest to your imagination. These details apply to the "Version 1" ADIO-Link and DIO-Link boards because the "Version 2" designs are still slightly in the future.

The ADIO-Link prototype was an RTC3 1 with either one or two RTCIO boards stacked atop it. Unlike all of the other Links, the ADIO firmware must figure out what hardware is attached to it and do the right thing regardless of what commands it receives from the SC (or your own controller, for that matter). There are only two I/O commands: S sets a new

Listing 2—This code reads the clock and formats the timestamp string. It ensures that the chip is not busy and halts the clock so he registers are consistent throughout the process. Some error checking code is omitted to save space.

```

/* Formats time string given clock device ID          */
/* Halts the clock while reading to avoid bad results */
/* Returns 0 for OK, 1 for error                      */

int IOFormatTime(DevCode)
WORD DevCode;
{
    int RetCode;
    BYTE DevNdx;
    BYTE DevChan;
    unsigned char *pFormat;
    char *pOut;
    BYTE Reg;
    BYTE Busy;
    BYTE WasBusy;
    RetCode = 1;
    DevNdx = IO_DEVICE(DevCode);
    DevChan = IO_CHANNEL(DevCode);
    strcpy(IOClkBuff,IOName[DevNdx]); /* set up device name */
    strcat(IOClkBuff,"=");
    do {
        IOWritePort(DevNdx,13,0x5); /* turn HOLD on, IRQ=1 */
        Busy = 0x02 & IOReadPort(DevNdx,13); /* fetch BUSY flag */
        if (Busy) { /* if BUSY active... */
            WasBusy = 1; /* remember for later */
            IOWritePort(DevNdx,13,0x4); /* . . . turn HOLD off */
        } while (Busy); /* until no longer BUSY */
        pOut = IOClkBuff + strlen(IOClkBuff);
        pFormat = IOTimeFormat;
        while (*pFormat){
            if (0x80 <= *pFormat){
                switch (*pFormat){
                    case 0x90 :
                        sprintf(pOut,"%03u",MSPERTICK*(WORD)FracSec);
                        pOut += 2;
                        break;
                    case 0x91 :
                        Reg = 0x0F & IOReadPort(DevNdx,12);
                        strcpy(pOut,IODays[Reg]);
                        pOut += strlen(IODays[Reg])-1;
                        break;
                    default :
                        *pOut = '0' + (0x0F & IOReadPort(DevNdx,
                                                                    (*pFormat)-0x80));
                }
            }
            else {
                *pOut = *pFormat;
            }
            ++pOut;
            ++pFormat;
        }
        RetCode = 0;
    Done:
        if (RetCode){
            CmdShowMsg(IOClkBuff); /* can suppress errors */
        }
        else {
            IOWritePort(DevNdx,13,0x4); /* turn HOLD off... */
            strcat(IOClkBuff,WasBusy ? " * " : " "); /* hit busy? */
            Nputstr(IOClkBuff); /* show the results */
        }
        return;
    }
}

```

value in a device, while Q queries the current value.

Each device is known by name: the DACs are A00 and A01 (that's "oh zero" and "oh one"), the ADCs are A10 and A11 ["eye zero" and "eye one"], the digital I/O ports are DPO and DP1, and the clocks are CLK0 and CLK1. Why you'd want two clocks isn't clear, but I suppose you could track different time zones or cross-check the pair.

The first RTCIO board at addresses E000–3F holds the "zero" devices, while the "one" devices are on the second RTCIO at addresses E040–7F. You can omit any devices you don't need from the boards and even skip the second board entirely.

The firmware imposes some uniformity on the chaotic collection of device modes by allowing you (the controller) to address all the devices in the same manner. Setting a DAC channel is just like writing to a digital port: S A00.1=ED or S DP0.1=ED.

Even better, the firmware records the values you write, so you can "read back" the last value from a device like the DAC that is normally write-only: Q A00.1 or Q DP0.1.

You can also address bits within the ports: S DP0 . 1.7=1 turns on the highbit of port 1 and Q DP0.1.7 returns the current bit value. Although you can do this to the DAC (and even the ADC!), I don't know why you'd want to... but I couldn't justify adding any code to disable it!

The RTCIO board uses an 8255, which can be configured in a bewildering variety of ways. Rather than wrap a firmware coat around this beast, you simply set the control port directly and are thereafter responsible for using it correctly. For example, to set all three ports to output mode, send the command: S DP0. 3=80.

All three 8255 ports start out as inputs, so the firmware fakes an initial control port value of 9B. Even though some 8255 parts do not allow you to read back the control port, the firmware stores the values for later "readback" or bit twiddling.

The DIO-Link prototype is simply an RTC31 with eight Port 1 bits called DP.0 through DP.7. However, that

Listing 3—*This code translates a timestamp string back into the values needed for the M6242 registers and sets the chip accordingly. Some error checking code is omitted to save space.*

```

/*-----*/
/* Un-format time string and set the clock */
/* Fractional seconds are automatically cleared when next */
/* second ticks */
/* Function return value is 0 for OK, 1 for error */

int IOSetTime(ppCmd,DevCode)
char **ppCmd;
WORD DevCode;
{
    int RetCode;
    unsigned char *pFormat;
    char *pChar;
    BYTE Reg;

    RetCode = 1;
    pFormat = IOTimeFormat; /* aim at format string */
    while ((*ppCmd) && ('.' != **ppCmd) && (*pFormat)){
        if (' ' == **ppCmd) { /* strip blanks */
            ++(*ppCmd);
            continue;
        }
        if ('.' == *pFormat) { /* from format, too */
            ++pFormat;
            continue;
        }
        if (0x80 > *pFormat){
            if ((*pFormat) != **ppCmd) I
                sprintf(CmdRespBuff,
                    "**** Separator mismatch at [%s], clock may be bad\n",
                        *ppCmd);
            CmdShowMsg(CmdRespBuff);
            goto Done;
        }
        ++pFormat; /* skip literal char */
        ++(*ppCmd); /* both places */
    }
    else {
        switch (*pFormat){
            case 0x90 : /* fractional seconds */
                while (isascii(**ppCmd) && isdigit(**ppCmd)){
                    ++(*ppCmd); /* skip digits */
                }
                break;
            case 0x91 : /* day of week */
                pChar = IOClkBuff; /* extract to buffer */
                memset(IOClkBuff,0,sizeof(IOClkBuff));
                while (isascii(**ppCmd) && isalpha(**ppCmd)) {
                    *pChar = toupper(**ppCmd);
                    ++pChar;
                    ++(*ppCmd);
                }
                for (Reg=0; Reg<7; ++Reg) { /* look up in table */
                    if (!strcmp(IOClkBuff,IODays[Reg])) {
                        IOWritePort(IO_DEVICE(DevCode),12,Reg);
                        break;
                    }
                }
                if (8 == Reg) {
                    sprintf(CmdRespBuff,
                        "**** Invalid day of week [%s], clock may be bad\n",
                            IOClkBuff);
                    CmdShowMsg(CmdRespBuff);
                    goto Done;
                }
                break;
        }
    }
}

```

(continued)

exposes "bare" CPU pins to the outside world, which is not generally a good idea. The official DIO-Link board provides four buffers in each direction to protect the 8031 from transients and to provide some useful output drive.

The firmware allows you to send a complete ASCII string to the port by pulsing P3.5 low after each byte, which forms a simple parallel output port. There are no handshaking signals, so your printer or other widget must accept strings of up to 200 bytes at about 500 bytes per second.

TELLING TIME

Most of the HCS II Links display their data as either hex or decimal values with little need for extensive formatting. As you have seen throughout this series, most of the output is handled by the C printf or sprintf functions.

The M6242 clock presents a challenge, because it's not really helpful to present the current date and time as hex register values. Even though the user may be the SC or

another robot, we figured it was handy to dump the date and time in a simple, readable format should a person have to decipher it.

The ADIO firmware includes two special cases to simplify working with

the clock. The command Q C LK0 without a port number produces the output

CLK0=04/07/92 TUE 16:46:54.430

Listing 3—continued

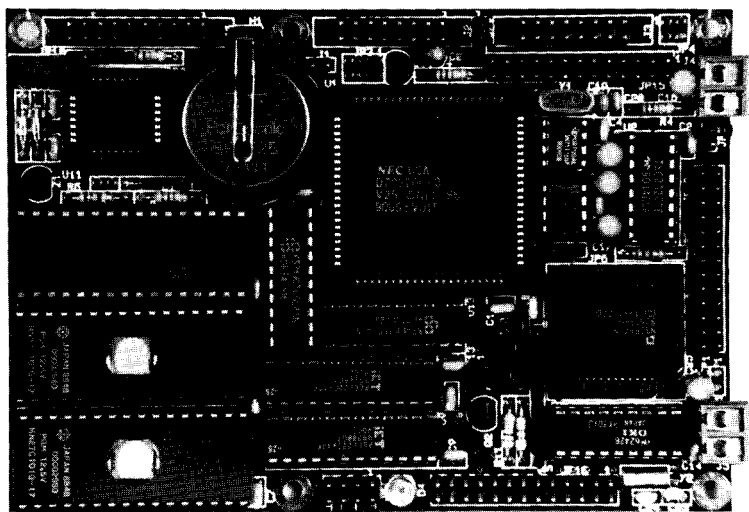
```

default :                               /* register formatters */
if (isascii(**ppCmd) && isdigit(**ppCmd)){
    Reg = **ppCmd - '0';                /* convert to digit */
    IOWritePort(IO_DEVICE(DevCode),(*pFormat)-0x80,
                Reg);
    ++(*ppCmd);
}
else {
    sprintf(CmdRespBuff,
"**** Invalid time string at [%s], clock may be bad\n",
            *ppCmd);
    CmdShowMsg(CmdRespBuff);
    goto Done;
}
}
++pFormat;
}
Done:
return RetCode;
}

```

V25 POWER COMES TO EMBEDDED CONTROL!

Micromint's RTCV25 is the perfect marriage of an 8088-compatible processor, programming convenience, and control I/O. Forget the need for cross-assemblers and cross-compilers; your favorite high-level language for the PC and a "locate" facility are all you need. The RTCV25 enhances the V25's power with parallel I/O; A/D conversion; RS-232 and W-485 serial ports; up to 384K of RAM and EPROM; a battery-backed clock/calendar; 128 byte EEPROM, ROM monitor, and the RTC stacking bus. Ease of code development combined with its small size and low power consumption make the RTCV25 ideal for all embedded control applications. And of course the RTCV25 is compatible with Micromint's full line of RTC peripheral boards and products.



Features:

- 8 MHz CMOS NEC V25 processor
- Upto 384K of RAM and EPROM
- Two serial ports
- Battery-backed clock/calendar
- 128 byte EEPROM
- 40 parallel I/O lines
- 8-channel, 8- (or 10-) bit ADC
- RTC stacking bus
- Small 3.5" x 5" size
- 5-volt only operation (150mA max.)
- Full featured ROM monitor
- ROMable operating system

100 Qty.
OEM configuration

\$270



MICROMINT, INC.

4 Park St. • Vernon, CT 06066

call 1-800-635-3355

(203) 871-6170 • Fax: (203) 872-2204

See us at the Embedded Systems Conference—Booth #919

You set the clock by sending an `CLK=` command with a string that looks much like that, which is certainly easier than figuring out the register settings! We figured that the clock would rarely need setting, so a human-format string made a lot of sense. After all, I was going to be setting the clock a lot!

Although I could use `printf` to format the output string, how to build a reasonably rugged input translator to go the other way wasn't clear. After trying a few rather ugly contraptions, I finally settled on a pair of routines that work from a formatting string defining what values appeared where.

The `IOTimeFormat` strings shown in Listing 1 uses characters between `\x80` and `\x8F` to represent the 16 M6242 registers; each such character is replaced by the corresponding register contents. Because the values are always expressed as two-digit decimal numbers, there is no need for `printf`'s extensive controls over the numeric format; a single special character suffices.

Two other special characters mark additional timing information. Character `\x90` indicates the three digits of fractional seconds and `\x91` selects the three-character day name. The fractional seconds count comes from an 8031 CPU timer ticking every 5 ms, while the day name is simply a table lookup based on the contents of a clock register.

Most of the output routine shown in Listing 2 is a loop that passes each `IOTimeFormat` character through a `Switch` statement. Characters below `\x80` are copied directly to the output, while the others are picked off by case tests. All 16 M6242 registers are handled in one statement because the formatting is identical for each, even though it may not make much sense to display registers C through F in this manner.

Listing 3 shows the code required to go from an ASCII string to M6242 register values. A loop marches through the input string and `IOTimeFormat` in lock step, using the latter to decide what to do with the former. While this arrangement does not allow much input format flexibil-


ity, to insist on one time-stamp layout here seemed perfectly reasonable to me.

You can use this same trick whenever your code must do odd things to produce an output. The basic idea is to put all the ugliness in one place, with a simplified control string to orchestrate the results. If you get carried away you'll create a little language with opcodes and operands, complete with an interpreter to translate opcodes into actions.

In fact, that's what I wound up doing for a recent project. The object was to build a CD jukebox; the problem was it had to work with many different CD players. I designed a CD player control language with instructions to select a disk, find a track, start and stop playing, and so forth. The main jukebox program [in C, natch] invoked these instructions when it needed to do something with the player.

The instructions boiled down to specific arithmetic operations ("repeat the next instruction ten times") or IR remote control outputs. I needed a different set of instructions and signals for each player, but the jukebox code remained unchanged. Worked like a champ...I'll have to do a column on mini interpreters, but you've already got the general idea.

RELEASE NOTES

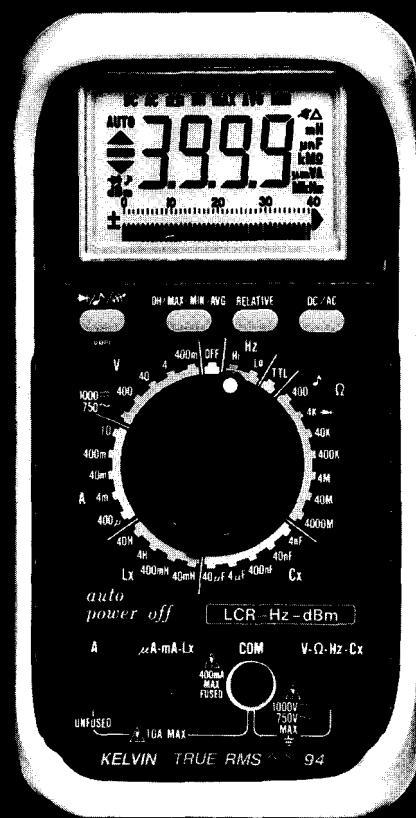
The BBS files include the executable EPROM hex files for both the ADIO-Link and DIO-Link modules for your noncommercial use. The source routines shown above are also included. The ADIO-Link and DIO-Link source code may be licensed from Circuit Cellar Inc. (not INK). 

Ed Nisley is a Registered Professional Engineer and a member of the Computer Applications Journal's engineering staff. He specializes in finding innovative solutions to demanding and unusual technical problems.

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

LCR - Hz - dBm - True RMS



Model 94

1% ACCURACY on DC Voltages

TRUE RMS PLUS

MAX/MIN/AVERAGE MEMORY RECORD
RELATIVE MODE 1 DATA HOLD

DC/AC VOLTMETERS

DC Range: 400mV, 4V, 40V, 400V, 1000V
AC Range: 400mV, 4V, 40V, 400V, 750V

DC/AC AMP METERS

DC/AC Ranges: 400uA, 4mA, 40mA, 400mA, 10A

OHM METER

Range: 400, 4K, 40K, 400K, 4M, 40M, 4000M Ohms

FREQUENCY COUNTER-AUTORANGING

Range: 4KHz, 40KHz, 400KHz, 4MHz (Trigger Low), 20MHz (Trigger High)

LOGIC PROBE

AUDIBLE CONTINUITY TESTER

CAPACITANCE TESTER

Range: 4nF, 40nF, 400nF, 4uF, 40uF

DIODE TESTER

dBm TESTER

Range: -25.7 dBm to 50.7 dBm

INDUCTANCE TESTER

Range: 40mH, 400mH, 4H, 40H

10MEGA OHM IMPEDANCE

10A HIGH-ENERGY FUSE PROTECTION
AUTO SLEEP & AUTO POWER OFF

WATER & DUST RESISTANT

MODEL 94 COMES COMPLETE WITH
TEST LEADS, YELLOW HOLSTER,
TILT STAND, BATTERY & FUSE

Stock No.
990111

\$199⁹⁵
ONLY

2 YEAR WARRANTY

KELVIN
ELECTRONICS

10 HUB DRIVE, MELVILLE, NY 11747

(800) 645-9212

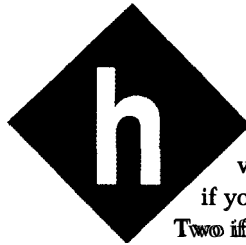
(516) 756-1750 (516) 756-1763/FAX

#153

Approaching PCB Nirvana

FROM THE BENCH

Jeff Bachiochi



ow long will you wait? Nine months if you want a child.

Two if you're growing tomatoes. Weeks if you're looking for the next issue of your favorite magazine. Or days until the weekend. Tomorrow's paper will take a day, whereas your pictures could be ready in an hour. However, if you've finished a new design and are looking for prototype boards, then you'll probably be twiddling your thumbs for at least a week if not two or longer!

Sure, you could simulate your hardware to show its feasibility, but showing off (or selling) a piece of simulated hardware is tough. You have no sense of satisfaction, no sense of accomplishment without being able to touch that finished project. Even a week (as advertised by some prototyping houses) can seem like forever. How can you leap-frog into the assembly phase without all that loitering? Cut out the middleman.

How many of you have attempted to fabricate your own printed circuit board? The path is not as quick, clean, and cheap as we all wish it was. Assuming you have finished artwork, the positive (or negative) is exposed to a chemically treated copper-clad board using a light source. The chemicals react differently to the absence or presence of light. When developed, trace and pad areas on the board retain the chemicals that protect the copper from being etched away, while the other areas are rinsed clean and are removed by the acid bath that follows. Once the acid has etched away the unwanted copper, the board is drilled. This process is a bit more involved

when producing double-sided boards because the side-to-side alignment is very critical. A big problem is the inability to make plated-through holes.

Although plated-through holes are still not practical, the photochemical step has been all but eliminated. If you are one of the regular callers to the Circuit Cellar BBS, you've read through the threads that deal with PCB fabrication. It didn't take Yankee or any other regional ingenuity long to figure out that the toner deposited by laser printers and copiers was a good etcher resist. However, even if copper-clad material could pass through the mechanism without removing the surface of the drum, the toner would not deposit well on the metal surface.

I had partial success using the acetate film normally used for overhead transparencies. The clear, high-temperature film can withstand the bonding temperatures of a laser printer or copier. The resultant image (which must be a mirror image of the original artwork) is transferred to the copper-clad board by reheating the film against the copper. Great idea, poor quality. The film doesn't accept the toner adequately, leaving holes and thin areas. The toner that does get on the film has a hard time choosing whether it will stay resident or be transferred to the copper-clad board when reheated. So the end product is a bit less than perfect.

TWO BIRDS, ONE STONE

The people at DynaArt Designs have come up with a cure for this troublesome procedure. In fact, it has some interesting additional benefits that I will discuss a bit later. But first, what makes this stuff so special? Remember how the transparency material wouldn't take the toner efficiently? Well, this material uses a paper base, so it absorbs most of the toner when used in standard paper copies. A "secret sauce" coats the fibers and prevents the fused toner from bonding to anything but the sauce. When reheated against the copper board, the paper is thoroughly stuck and, like the transparency, would rip off some of the toner when cooled and separated. Now for the

The search for the best way to make your own prototype PC boards is often like the search for the Holy Grail. Sir Jeff's come pretty close this time with his newest drinking cup.

secret of the sauce. When soaked in water, the coating dissolves and paper completely releases from the toner, which is now fused to the copper. This procedure brings back memories of soaking decals to decorate my models.

HYPE VS. REALITY

Sound good! Let me put it to the test. Back in *Circuit Cellar INK*, issue #22, I put together a power control module that fit on a 9-volt battery clip. The module consisted of a 5-volt regulator with edge-triggered inputs to independently turn the power on or off. It's a good test circuit because it is a single-sided design and has thin (10-mil) traces that must pass between leads on a SOL (small outline package) with 50-mil lead spacing.

The Schema PCB package I use can produce mirror-image Gerber photoplot files, although I had previously saved this job as standard image files. The **GerberJet** program I use to print review plots on my HP LaserJet prior to having the job photoplotted also allows mirror imaging. Using the mirror image function here rather than going back to PCB was the quickest way to get a full-size mirrored artwork, printed on standard copier paper.

The specially coated transfer paper comes in 8.5" x 11" sheets (big enough for most projects). Still, I hated to use a full sheet for an artwork of smaller than 2 square inches. So, cutting off a 2-inch square section, I grabbed the plot that had just been ejected and centered the transfer material (like a patch) over the area previously printed. I secured it on opposite sides using clear tape and reinserted it into the printer's paper tray. Seconds after resending the file to the printer, the page popped out shouting, "Transfer me!"

HOMEWORK

"Uh, Beverly? Where's the steam iron?" I said, desperately trying to think of an excuse. "Aren't you thoughtful," my wife replied. "It's right there under that pile of ironing." Too late, she had managed to outfox me. I quickly bit my tongue to cut my losses. If she found out why I need the iron it would only get worse.

I thought of playing the idiot and "accidentally" melting a few pieces of synthetic clothing, but promptly realized that would only backfire. Ruined garments would be a legitimate excuse to go shopping at the mall. "I guess this time I'm stuck," I thought. Then I caught sight of the dirty clothes hamper... After thoroughly burying the ironing when she wasn't looking, I slipped down to the basement to complete the second step of my experiment.

While the iron was getting hot (steam, not soldering), I prepared the

circuit board. Rubbing feverishly with a small wad of fine steel wool, I buffed the copper-clad board first in one direction and then the other. A washing with dish detergent removed any oil from the board that might interfere with the adhesion of the toner [it also left my hands soft and lemony]. I inspected the surface of the board for defects, such as a dimple or a dent, and made sure the edge of the board had no burrs. They would have prevented the iron's heating surface from making smooth and continuous contact with the copper. I placed the

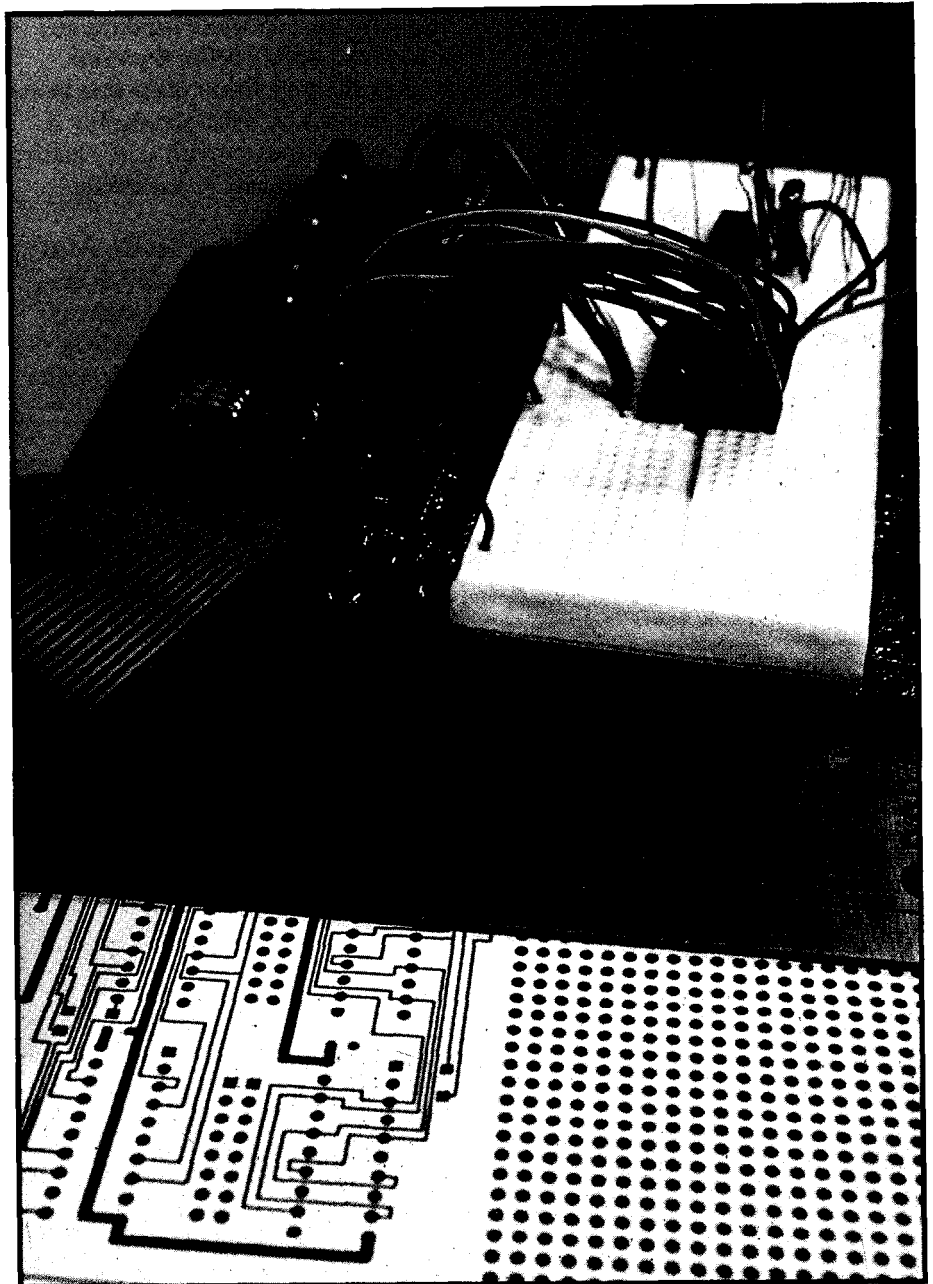


Photo 1—The quick path to reality: Minored artwork is printed on special transfer paper, which is then transferred to copper-clad board, etched, drilled, and assembled. It's still up to the designer to get the circuit working.

board, copper side up, on a hand towel to prevent the board from sliding around and centered the mirrored artwork on the board toner side down. A special nonstick sheet is provided in the kit that prevents the iron's surface from accidentally sticking to transfer paper. Although our iron is coated with Teflon, I used the sheet anyway.

By now the iron was up to temperature: 300°F (cotton setting). For boards smaller than the iron, little movement is needed. Larger boards require moving the iron in a circular motion, from one area to the next, so all parts of the board are heated equally. No need to press down; the image transfers clearer when just the weight of the iron is used.

The magic starts when the board and transfer are placed in a dish of water. Within seconds, the transfer becomes saturated and the special coating starts to dissolve. As the paper floats to the surface, it leaves behind the rebonded toner now attached to the board. At this point, I inspected the artwork for missing, smeared, or

cracked traces. Small defects can be touched up with a marking pen to add resist or a sharp knife to remove it. If you find gross defects, remove the toner and reapply a fresh image. The board is now ready for the normal etching process.

Ferric chloride is the most widely used chemical etcher for copper. A room-temperature bath of acid will remove the unprotected copper in about an hour. Etching time can be decreased by aerating the acid, heating it, or doing both.

I'd like to quickly cover some rules to live by when working with acid. Use caution while handling any acid. Use a double boiler if you are heating the acid. Don't place acid in a metal container. Wash thoroughly if you come in contact with acid. Finally, dispose of spent acid as hazardous waste.

If you are making a double-sided board, protect the copper clad of the second side with tape while etching the first side. When the first side is complete, drill guide holes in opposite

corners of the board. You may want to place pads for this purpose outside of the artwork's perimeter to ensure the hole does not interfere with the adhesion of the second artwork. Align the guide holes in the board with the corresponding pads on the second artwork, affix it in place with a few pieces of tape, then proceed with the transfer and etching as before. Remember to protect the first side with tape when etching the second side to prevent the first side from overetching. Rinse and soak the circuit board to completely remove all acid once the board has been etched. Any acid left on the board will continue to remove copper until the traces no longer exist!

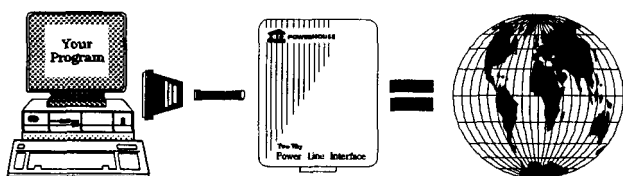
THE HOLE PROBLEM

The largest and still unsolved problem has to be plated-through holes. On a single-sided design [circuitry on the bottom and components on the top or circuitry on the top and surface-mount components on the top], plating through is not a concern. In multilayer boards (more than one

TW523 Power Line Interface

Developers Kit

Interface Your Computer To Transmit And Receive X-10 Codes Over Your AC Power Line. Two-Way Communication. Real Time Environment Control.



Kit Includes
TW523, Cable, Interface Connector (S/P)
Documentation. Source Code Supplied in
"C", Pascal, BASIC or Run Time.
Disks 5.25in & 3.5in Format.

\$75.00



Bat-an-Harper Group Inc.

Voice (416) 294-6473 BBS (416) 471-6776
Fax (416) 471-3730

PC-Based Logic Analyzers



New Prices for 1992

ID160 (50MHz) \$595

ID161 (100 MHz) \$695

*High Speed • 8K Trace Buffer • 16 timing channels
expandable to 32 state channels *Multi-Level Triggering
*State Pass Counting *Event Timer/Counter • Performance
Histograms *Hardcopy Output *Disassembles S-bit micros
*Supports VGA /EGA/HGA • Demo diskette available
30 Day Money Back Guarantee



INNOTECH DESIGN, INC.

6910 Oslo Circle, Suite 207
Buena Park, CA 90621
Tel: 714-522-1469 FAX: 714-527-1812

circuitry layer), connecting layer traces to one another is accomplished by plating the inside of each hole, making a mechanical and an electrical connection between layers. Such a process is presently not practical for the average individual.

A couple rules should be followed to ensure the prototype is workable without using eyelets or other connecting devices. Keep vias out from underneath components. Place traces on the component side only if a connection is solderable and not lying beneath a component.

Well, this tiny, single-sided board was a piece of cake. In fact, by using multiple images (step and repeat of the same artwork), I got eight of these tiny circuits all on the same 3" x 5" board.

Enough of this miniature stuff, now's the time for a more challenging design. I always wanted to make a parallel port interface with a few I/O ports that would allow easy experimentation with new chips. I dug up an old design and double-sided layout I had completed a while ago, but never

had any prototypes made. Photo 1 shows this design turned into reality without having to go to a fab house. This double-sided board was considerably more difficult because of its larger area and longer trace runs. However, after working with the smaller design, I had the confidence and techniques necessary to succeed.

Drilling is an important step, and I recommend using at least a Dremel-type drill mounted in a drill press frame. Use the smallest drill that will still allow the component lead to fit through; I used 0.030" for most components. Trying to hand-drill IC holes is a sure way to miss the pad centers. I called out 50-mil IC pads. Next time they will be as large as possible—60 mil minimum. This step is most critical on double-sided boards where the circuits might be misaligned, causing the drill to rip off part of the pad on the solder side. Place the board being drilled on a piece of new (no holes) wood to help the copper stay attached to the bottom side of the board.

APPLY DECALS <HERE>

Further discussion of layout strategies, etching, plating, and other construction techniques will not be covered at this time so I can bring you this exciting alternative to rub-on lettering. The same process that provides your copper-clad boards with an etcher resist can also—you guessed it—decorate your project's exterior. With the introduction of color copiers, you can produce decals in dazzling color.

Start with a right-reading image—as opposed to a mirror image—and apply three good coats of clear lacquer over the image. The lacquer adheres to the toner image on the specially coated transfer paper. When dry, it acts as the decal's clear base, holding the toner after it is released from the paper after being soaked in water. The same coating that allows the toner to be released from the paper also acts as an adhesive once it is dry, so the decal will stick to most smooth surfaces. If a mirror image is used, you can apply the decal to the inside surface of a

The Quick Way into Embedded Software Development



Integrated Development Environment
For (Almost) Any Language

IDEAL brings together all of the tools used to develop software for Single Board Computers (SBC) and blends them into a menu-driven, windowed environment that speeds your development process.

- Automates the Compile-Assemble-Link-Download Cycle
- Works with any Compiler, Assembler, Linker, etc.
- Use the built-in Editor or link to your own Editor, it's easy
- Great for BASIC and FORTH interpreters too
- Interrupt driven serial communications to the SBC
- Swaps memory to make room for the largest tools
- Resizable, Zoomable Windows for Editing, Browsing and Target Communications
- Fully integrated mouse and keyboard control
- In use for 2 years at Major Universities and Corporations

Requires PC/XT/AT w/512K
Demo Available on BBS

30 day Money-Back Guarantee



creative applications engineering, inc.

Phone (415) 494-2363
CompuServe: 74156,1207
BIX: ecarrier
P.O. Box 9524, Stanford, CA 94309 BBS (415) 494-8463

DC/CAD

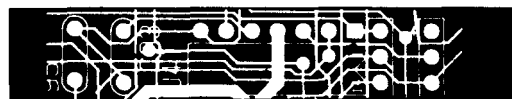
introducing...

THE TERMINATOR

Super High Density Router
(Complete with Schematic & PCB EDITOR)
Features the following powerful algorithm & capability:

- Rip - up and Retry
- Pre-routing of SMT components
- Real-Time via minimization
- Real-Tie clean up passes
- User defined strategies
- Window 3.0 capability as DOS Task
- 1-mil Autoplacer and AutoPanning
- Two-way Gerber and DXF
- Automatic Ground Plane w/ Cross-Hatching
- Complete w/ Schematic & Dolly Libraries
- Optional simulation capability & protected mode for 386 users

* PCB LAYOUT SERVICE AT LOW COST *
LEASE PROGRAM & SITE LICENSE AVAILABLE



Design
Computation

1771 State Highway 34
Farmingdale, NJ 07727
(908) 681 - 7700 • (908) 681 - 8733 (FAX)

"DC/CAD... The focal point of future CAD market"

clear faceplate for the ultimate protection. The back surface can then be painted to further protect the image as well as add a background color to the image.

I don't have access to a full-color copier, so I couldn't try any fancy stuff, but I did make a two-color project decal using a plain old copier that offers a few colors by changing toner cartridges. I drew the decal design using the text- and line-drawing attributes of Schema. I made two printouts of the design: one with the black elements and one with the red elements (of course, these both came out black on the laser printer, but were in perfect registration with the paper's edges). I placed the page with the images I wanted black in the copier and passed the special paper through it. I replaced the toner cartridge with a red one, placed the other artwork in the copier, and passed the special sheet through a second time.

At this point, you have to give the two toner colors deposited on the paper something to adhere to; you

can't very well iron this onto a plastic faceplate. This step is completed by building up a few layers of clear acrylic spray right over the special coating and deposited toner [if you are using a mirrored artwork, this spray can be a colored acrylic]. When soaked in water, the acrylic layers act as a clear base for the toner. It is extremely fragile and may tear if the acrylic isn't thick enough, so treat it gently.

One last note about copiers: they are not all created equal. Not in toner quality nor in reproduction size. Make test copies of everything on plain paper first, looking for strong, even toner deposits and checking the length and width of the copies. One copier I checked enlarged the length by 0.1" over 6", but did not enlarge the width at all.

ON A ROLL

I hope this information has started those little wheels a-turnin' upstairs. I think you'll agree this product is the kind that will open up all sorts of possibilities. I find this stuff to be one

of the most cost-effective tools available. And because it is useful in a number of different applications, it should have a strong future, allowing us all to present our projects with a more professional look. □

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on the Computer Applications Journal's engineering staff. His background includes product design and manufacturing.

SOURCE

Toner Transfer System
#TTS 585 1 1-392

DynaArt Designs
3535 Stillmeadow La.
Lancaster, CA 93536
(805) 943-4746

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

The Ciarcia Design Works

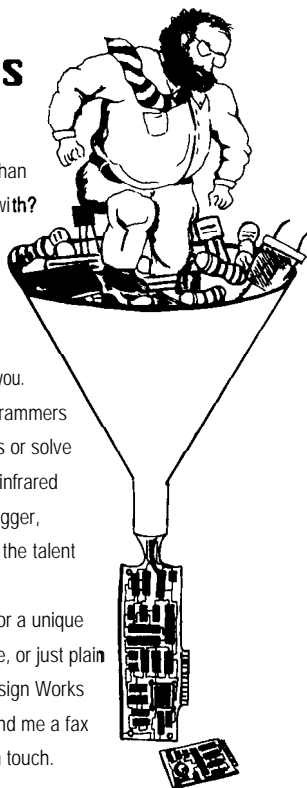
Does your big-company marketing department come up with more ideas than the engineering department can cope with?

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

Steve Ciarcia and the Ciarcia Design Works staff may have the solution for you.

We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems. Need an infrared remote controller, multi-channel data logger, or 7-axis drill controller? The team has the talent to design and manufacture it!

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



Remember...a Ciarcia design works!
Call (203) 875-2199 • Fax (203) 872-2204

68HC705K1

REAL TIME In-Circuit Emulator

An add-on board, with re-created 68HC705K1 interrupt logic in hardware, converts the Motorola 68HC05P8 EVS into a complete 68HC05 K family real time ICE, not a simulator. Comes with advanced PC driver software, 16 pin emulator cable and plug. (Demo disk: \$5.00)
Part number: EVSK1-ICE Price: \$68.05

68HC11 A,D,E,F

- PC based user friendly real time 68HC11 ICE.
- New 64K memory module supports different probes.
- Active probe reduces propagation delay to target system.
- Full speed up to 20 MHz, single chip and expanded modes.
- On-board 64K emulation RAM maps to 4K blocks.
- 64K real time hardware breakpoints.
- Breaks on address, address range and memory RD/WR.
- 40 pin logic analyzer connector.
- Full symbolic debugging. Supports all A,D,E, and F parts.

64K memory module \$295.00
Any probe (with PC driver) \$450.00
44 PLCC to 40 DIP or
52 PLCC to 48 DIP adapter \$55.00

Call: (708) 894-1440

Wytec

Suite 140
185C E. Lake Street
Bloomingdale, IL 60108

Z8

WICE Z8 emulator \$995
86C08 adapter w/analog comparators \$55

#158

The Ultimate Desk ACCESSory?

If DEC and Philips/Sigmetics have anything to say about future PCs, your days of crawling around under your desk routing cables may soon come to an end.

SILICON UPDATE

Tom Cantrell



As a longtime user of both **Macs** and PCs, I'm uniquely qualified to throw in my

two cents regarding the relative strengths and weaknesses of each.

With my Mac hat on, I amuse myself by watching Microsoft rake in the dough with constant "upgrades" of Windows (e.g., 286, 3.0, 3.1, NT?), which millions of users seem to accept with good grace. Me! I won't consider anything sooner than NT and probably at least version 3+ at that.

Meanwhile, as a PC proponent, I point out the plethora of engineering software that's available and the incredible price-to-performance comparison of clones. When I have to develop embedded hardware and software, the PC is king.

One area where the Mac has a decided advantage is built-in networking. The most well-known example is AppleTalk, which provides as-painless-as-it-gets networking of **Macs** and laser printers. More recently, the Mac also features the Apple Desktop Bus (ADB), which provides a low-cost, user-friendly way to daisy chain a variety of desktop input devices. For example, setting up a system with a keyboard, mouse, trackball, and digitizer is easy.

By contrast, configuring the same setup on the PC is an ugly job involving a strange variety of boards (each with the dreaded DIP switches, of course), connectors, and cables. The latter wind their way in an inevitably tangled mess from desktop to the PC.

If technology is developing at such a great rate, why am I still driven to my knees fumbling in that forbidding rat's nest of cables that lurks behind

my PC? Presumably, "wireless" technology will eliminate cable clutter someday, but until then, there's got to be a better way.

ACCESS.bus TO THE RESCUE

In an effort to bring ADB-like sanity to the PC world, DEC and Philips/Sigmetics are proposing the ACCESS.bus. Like the ADB on the Mac, ACCESS.bus is a simple, low-cost daisy-chain bus (see Figure 1). However, with the benefit of hindsight, ACCESS.bus offers a number of improvements compared to ADB including

- **High-Speed Data Transfer**—ACCESS.bus is much faster at 100K bits per second versus ADB 10K bits per second. This speed may seem like overkill for a keyboard, but it does allow for plenty of expansion without performance problems. Also, a fast version of ACCESS.bus—400K bits per second—will be offered later.

- **Larger Number of Devices**—Although ADB theoretically supports 16 devices, Apple documentation states "performance will probably deteriorate if more than three devices are daisy chained" (!). Also, ADB limits the power delivered to devices to a total of 500 mA. Meanwhile, thanks to a higher bandwidth, ACCESS.bus can support 14 devices, and no upper limit is imposed on total device power consumption.

- **Longer Cable**—More devices need more cable, so ACCESS.bus stretches the limit from the 5-meter ADB limit to 8 meters. This amount seems quite adequate for desktop work, but if it is not, active "repeaters" can be used.

- **"Hot Plugging"**—Apple specifically warns against the practice of adding a device to an active ADB bus. The ability of ACCESS.bus to let something plug in at any time could support unique identification and security applications, such as an ACCESS.bus/EEPROM-based "key."

Testifying to the credibility of the new bus are connector leaders Molex and Amp, who provide the neat ACCESS.bus phonelike modular connector (see Figure 2). It's a little bigger than a phone connector and

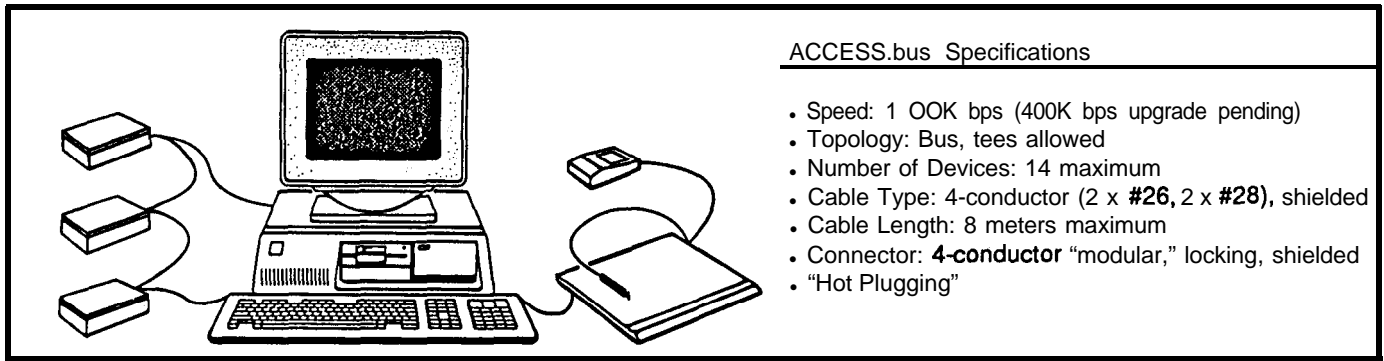


Figure 1—ACCESS.bus aims to eliminate the clutter of incompatible cables running out the back of your PC by using a "desktop bus" scheme similar to that used by today's Macintosh computers.

shielded to minimize RF problems. The specification calls for four-conductor cable (also shielded) with heavier wire for +5 volts and ground (#26 vs. #28). Unlike the DIN connectors used by ADB and current PC keyboards, the modular ACCESS.bus connector has the advantage of easy orientation. I don't know about you, but I inevitably end up "spinning" DIN connectors a lot even when I can see what I'm doing, not to mention when I'm groping through the all too typical "blind insertion."

Another plus is positive locking. Unlike a phone connector, the dual-release ACCESS.bus connectors seem to make "getting a grip" easier. Notice

how the entire connector is streamlined, easing wire routing and minimizing back panel clutter. The design also minimizes the "fishhook" syndrome exhibited by existing connectors [e.g., those DB-25s with the long knurled screws], which seem to get hung up on every possible obstacle as if they were possessed by some mystical attraction.

PC THE LIGHT

Besides causing grief for users, a multitude of desktop interfaces caused problems for DEC keyboard manufacturing. They had to offer X distinct keyboards, where X equals the number of popular layouts multiplied by the number of different interfaces. Thus, the seeds were sown for ACCESS.bus.

DEC approached Apple to see if they would consider offering ADB as an open standard. Apparently, Apple's response was something along the lines of "Hey, great idea—not!" so DEC started casting around for alternatives.

Here's where Philips/Sigmetics enters the picture. To make a long story short, the resulting ACCESS.bus is simply a derivative of that company's Inter-Integrated Circuit (PC) bus.

PC was originally designed as kind of a "LAN-in-a-Box," allowing easy connection between processors and interface chips without the bulk and expense of a full-speed parallel bus. Though you may not be familiar with it, PC is arguably the world's leading LAN because the bus is widely used in high-volume consumer electronics, such as TVs, stereos, and phones. Meanwhile, Philips/Sigmetics (and others under license) offer a plentiful variety of PC add-on chips including micros, EEPROMS, real-time clocks, ADC, DAC, and so forth.

PC is surprisingly sophisticated, despite its low chip cost, simple wiring, and a simple clocked serial port basis where data (SDA) is sampled when the clock (SCL) is high. On top of the basic communication mechanism, I²C layers a message format consisting of the destination address, a read/write flag, and the data framed by start and stop conditions. Furthermore, each byte transferred requires an ACK

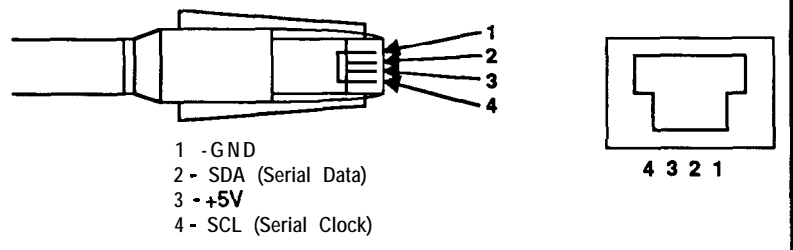
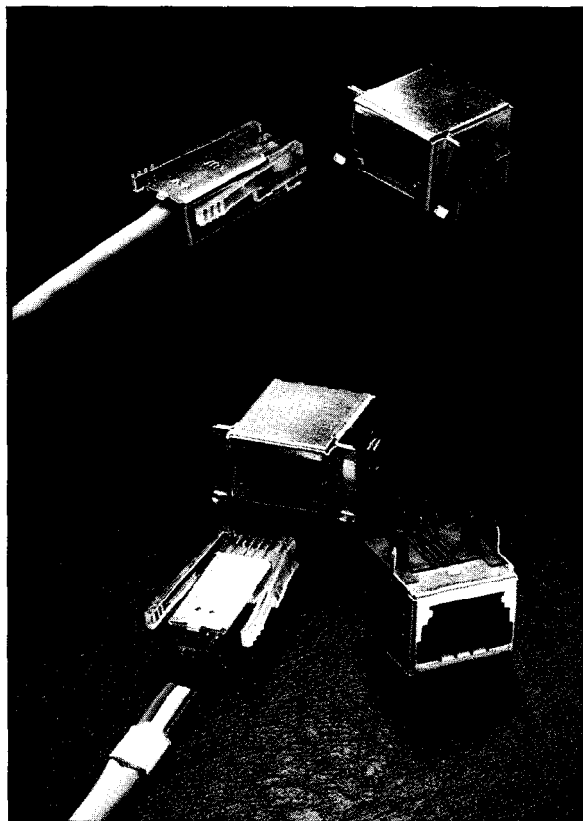


Figure 2—Based on PC, ACCESS.bus uses a simple four-wire interface that provides not only a data channel, but also power to peripherals. The proposed modular connector locks in place and eliminates orientation confusion.

or NAK from the recipient [see Figure 3].

PC is smart when generating the clock and when dealing with varying speed devices in particular. Relying on the use of open-collector drivers, slow devices can request the equivalent of wait states by holding the clock low. Because the clock synchronization is automatically handled by the PC hardware, you don't have to change DIP switches or software settings if you add a slow device.

As a multimaster bus, PC must face arbitrating between simultaneous data transfers. Most serial networks adopt a variant of "collision detection" in which each potential master monitors its own transmission and everyone backs off if a collision (i.e., what's on the wire isn't what was sent) is detected. Once again, relying on the open-collector nature of the bus, PC adopts an interesting variation where at least one of the messages will get through. During a simultaneous data transfer, all masters continue to output as long as the data on the wire

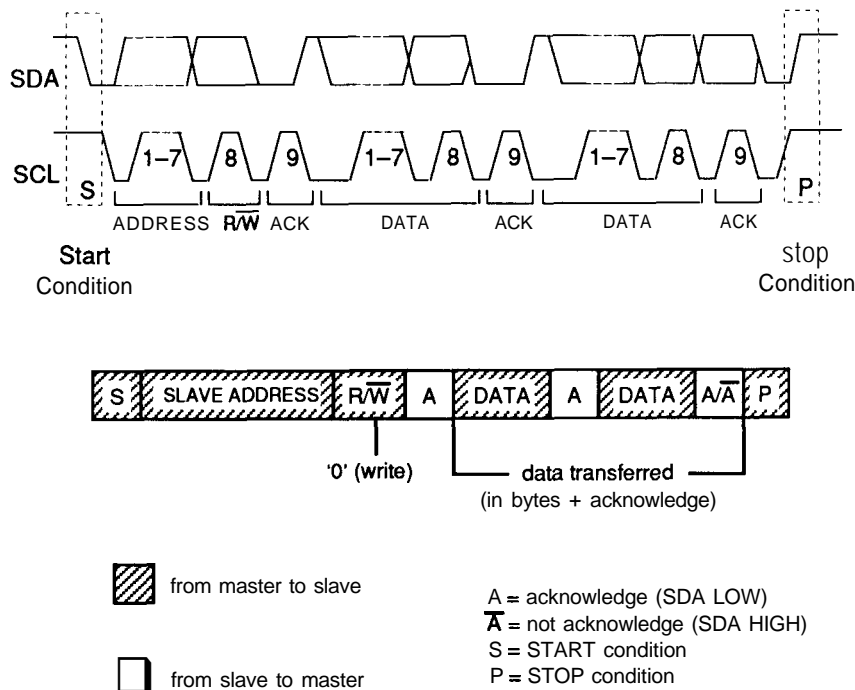


Figure 3—In PC, serial data (SDA) is sampled when the serial clock (SCL) is high. The basic packet of information consists of a destination address, a R/W flag, data, and start/stop framing. A simple ACK/NAK status is sent back by the destination device.

MOVE OVER INTEL MICROMINT SOURCES 80C52 CMOS BASIC CHIP

Micromint has a more efficient software-compatible successor to the power-hungry Intel 8052AH-BASIC chip. The 80C52-BASIC chip was designed for industrial use and operates beyond the limits of standard commercial-grade chips. Micromint's 80C52-BASIC chip is guaranteed to operate flawlessly at DC to 12 MHz over the entire industrial temperature range (-40°C to +85°C). Available in 40-pin DIP or PLCC

80C52-BASIC chip	\$25.00
OEM 100-Qty. Price	\$14.50
BASIC-52 Prog. manual	\$15.00

MICROMINT, INC.

4 PARK ST., VERNON, CT 06066
TO ORDER CALL

1-800-635-3355



EPROM PROGRAMMERS

Stand-Alone Gang Programmer

\$750⁰⁰



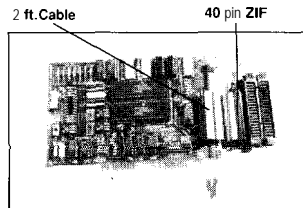
- Completely stand-alone or PC-driven
- Programs E(P)ROMs
- 1 Megabit of DRAM
- User upgradable to 32 Megabit
- 32K ZIF Sockets...IS-232, Parallel In and Out
- 32K Internal Flash EEPROM for easy firmware upgrades
- Quick Pulse Algorithm (27256 in 5 sec, 1 Megabit in 17 sec.)
- 2 year warranty
- Made in the U.S.A.
- Technical support by phone
- Complete manual and schematic
- Single Socket Programmer also available. \$550.00
- Split and Shuffle 16 & 32 bit
- 100 User Definable Macros. 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex, and Motorola S
- 2716 to 4 Megabit

Internal Programmer for PC

\$139⁹⁵

New Intelligent Averaging Algorithm Programs 64A in 10 sec. 256 in 1 min 1 Meg (27010,011) in 2 min 45 sec 2 Meg (2762001) in 5 min Internal card with external 40 pin ZIF

- Reads, Verifies, and programs 2716, 32, 32A, 64, 64A, 128, 128A, 256, 512, 513, 010, 011, 301, 27C2001, MCM 68764, 2532, 4 Megabits
- Automatically sets programming voltage
- Load and save buffer to disk
- Binary, Intel Hex, and Motorola S formats
- No personality modules required
- 1 Year warranty
- 10 days money back guarantee
- Adapters available for 6748, 49, 51, 751, 52, 55, TMS 7742, 27210, 57C1024, and memory cards
- Made in U.S.A.



EMPDEMO.EXE available BBS (916) 972-8042

NEEDHAM'S ELECTRONICS

1539 Orange Grove Ave. Sacramento, CA 95841
Monday-Friday 8 am-5 pm PST

C.O.D.

Call for more information
(916) 924-8037
FAX (916) 972-9960

matches what they are sending. Eventually, a particular master will output a high, but the output from another master will be holding the wire low. At that point, the loser (the master with the high output) detects a collision, quits transmitting, and has to try again. The process continues until a single winner, one whose message gets through, remains.

ACCESS.bus makes a few changes to the PC protocol. First, of the 128 addresses (7-bit address) defined by PC, 16 are allocated for **ACCESS.bus** devices as follows:

- . 50H: Host computer
- . 6EH: Default power-up address
- . 52-6CH (even addresses): 14 assignable device addresses

An **ACCESS.bus** message superimposes more information on top of the basic PC packet. As shown in Figure 4, this message consists of destination and source addresses, a byte count, the data bytes, and a checksum. Notice how the LSB of the addresses must be

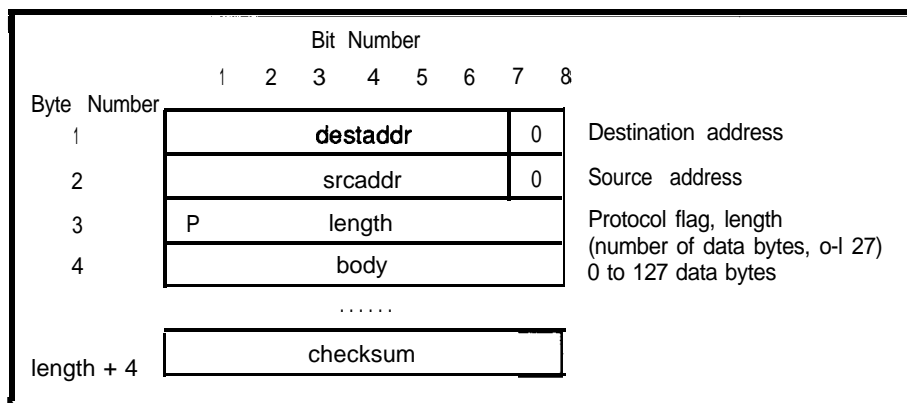


Figure 4—The standard **ACCESS.bus** message packet is based on PC, but **includes** additional information. The packet **consists** of a **destination** address, source address, length, data bytes, and a **simple** checksum.

a 0. For PC, this bit functions as a R/W direction flag, allowing both masters and slaves to function as transmitters and receivers. **ACCESS.bus** is more restrictive: only a master is allowed to transmit and only a slave may receive. The two-way communication is possible with **ACCESS.bus** because each device can be a master [when transmitting] or a slave (when receiving) at any given moment.

For example, the host computer is a master when transmitting to a device

and a slave when receiving data from that device. So, all messages on the **ACCESS.bus** are writes from masters to slaves, which is why the R/W flag is fixed at 0 [write].

Packed with the 7-bit data length specifier [i.e., message length = 0 to 127 bytes] the P [Protocol] bit specifies whether the message is a data or status/control transfer. For the latter, **ACCESS.bus** defines eight messages (four each for computers and devices) as shown in Figure 5. Of these, the

8051 EMBEDDED CONTROLLERS WITH ALL THE EXTRAS!

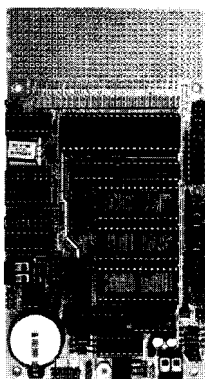
We offer a full line of low cost 80C32 embedded controllers and software tools. They are ideal for developing products, test fixtures and prototypes.

Features include:

- . Low power CMOS design
- . RS-232 and RS-485 ports
- . Up to 60K of code space
- . Up to 60K of data space
- . 5 to 15 volt operation
- . Small form factor (3.5" * 6.5")
- . System diskette includes application notes and an assembler.
- . Start at \$100

Available Options:

- . Real Time Clock
- . Watch Dog Timer
- . Multifunction Board adds LCD, Keypad, UART, A/D and 24 I/O lines.
- . BASIC-52 or Monitor/Debugger in EPROM
- . C Compiler, \$100



Iota Systems, inc.

POB 8987 • Incline Village, NV 89452
PH: 702-831-6302 • FAX: 702 831-4629

REMOTE POWER CARD!

3 VERSIONS:

RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

PHONE

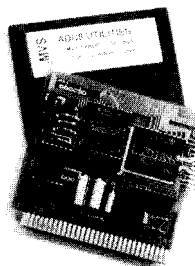
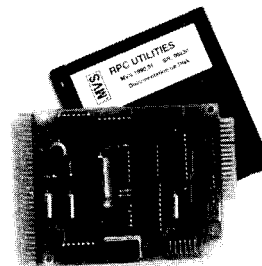
TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER

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

95\$ 27\$ OEM

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



8 CHAN ADC

DATA ACQUISITION, SERVO CTL, AUDIO
8-BIT RESOLUTION 220KHZ SAMPLE RATE
SHARP CUTOFF ANTI-ALIAS FILTER
CREATE STEREO BLASTER (VOC) FILES

95\$

2 CHAN DAC

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

75\$



MVS BOX 850
MERRIMACK, NH
(508) 742-9507

5 YEAR LIMITED WARRANTY
PING IN USA

most interesting are the commands that pass “capabilities” information from devices to computers.

The capabilities scheme is part of an effort by ACCESS.bus to introduce a measure of device and software independence. ACCESS.bus groups devices into three generic classes: Keyboard, Locator (e.g., mouse), and Text (e.g., bar-code reader).

The capabilities for a typical mouse might be defined as follows:

```

PROT(LOCATOR)
TYPE(MOUSE)
BUTTONS(1(L)2(R)3(M))
DIM(2) REL RES(200 INCH)
RANGE (-127 127)
DO(DNAME(X))
D1(DNAME(Y))

```

This information describes a device of type “mouse,” which uses the generic “locator” device protocol. The 2-D mouse [the dimensions are named X and Y] outputs relative movement between -127 and +127, with a resolution of 200 counts per inch, and has three buttons named L, R, and M.

ACCESS.bus makes a final step back from PC’s LAN-like pretensions by allowing transfers only between computers and devices and not among devices. This restriction is reasonable because having your keyboard and mouse talking to each other behind your back seems rather risky.

WHO’S ON FIRST

ACCESS.bus, like all LANs, faces the classic problem of uniquely identifying each node. At power up (or in response to a **RESET** () command) each ACCESS.bus device reverts to the default address. Next, the computer sends an **Identification Request** command to the default address (therefore, to all devices on the bus).

At this point, every device will attempt to reply with their 32-bit ID. The ID can be a unique serial number embedded in each device’s ROM. However, because this practice adds cost, the protocol also allows devices to generate their own ID, typically via a counter cleared at reset and incremented by the device’s internal

clock. The result is two of the same devices will usually come up with a different ID thanks to a slight difference in circuit timing.

That all the devices are trying to respond at once is resolved by the previously mentioned multimaster arbitration mechanism. As each ID message gets through, the computer sends an **Assign Address** command based on the ID. Once the device receives this command, it will assign itself the address specified. Once a device knows its address, it can commence sending and receiving data.

You probably have noticed this procedure has a small, but potentially fatal, loophole: the rare case that two devices report the same ID in the **Identification Request** phase. This subsequent **Assign Address** command will assign both devices the same ACCESS.bus address, which will surely cause problems.

To cinch this loophole shut, ACCESS.bus adopts one final trick. After receiving an address, but prior to first data transmission, each device sends a reset message to its own

address. The device sending the message is not itself reset, but any other devices at the same address are. Those devices that are reset will reenter initialization phases in order to receive new addresses.

TIMING IS EVERYTHING

The proponents of ACCESS.bus are careful to keep reminding us that it is mainly designed for low-speed and low-frequency (i.e., human) input devices. There is a danger of users and suppliers of other I/O devices boarding the bus without a ticket.

Witness the case with the PC printer port that has been hooked to just about every kind of I/O device including hard disks. The problem is that hooking high-speed block I/O devices could result in a compromised response. Devices like keyboards or mice may not generate a lot of data, but users won’t be happy if they don’t perceive these devices’ responses as instantaneous.

To this end, the specification imposes a number of limits on the amount of traffic or delays any device

Computer-to-Device Messages	Purpose
Reset()	Force device to power-up state and default I ² C address.
Identification Request()	Ask device for its “identification string.”
Assign Address (ID string, new addr)	Tell device with matching “identification string” to change its address to “new address.”
Capabilities Request (offset)	Ask device to send the fragment of its capabilities information that starts at “offset.”
Device-to-Computer Messages	
Attention(status)	Inform computer that a device has finished its power-up/reset test and needs to be configured; “status” shall be the test result.
Identification Reply(ID string)	Reply to Identification Request with device’s unique “identification string.”
Capabilities Reply(offset, data hag)	Reply to Capabilities Request with “data fragment,” a fragment of the device’s capabilities string; the computer uses “offset” to reassemble the fragments.
Interface Error ()	Invalid checksum or premature end of message detected.

Figure 5—An ACCESS.bus message may be either data or status/control. Eight status/control messages are defined, four in each direction.

can impose. For instance, a so-called noninteractive device like a laser printer can only occupy the bus for 5 ms at a time, which limits the maximum data block size to 50 bytes or so (even though the protocol allows up to 127 bytes of data in a message). Furthermore, the device must delay for at least 12 ms after transferring a message before starting another transfer. Finally, abuse of the clock synchronization scheme is prohibited; a device may hold SCL low only for a maximum of 2 ms.

Assuming noninteractive devices obey the rules, interactive devices [i.e., mouse, keyboard, etc.] have plenty of bus available, enough to guarantee 60 Hz (16.6 ms) response. This amount of time is essentially the CRT frame rate, so screen response is fast and smooth.

YAWN?

Is ACCESS.bus a YAWN [Yet Another Wiring and Networking scheme]? The backers of ACCESS.bus are to be commended for avoiding NIH pretensions. PC is quite suitable for


the task and clearly has a good laundry list of technical features.

The technical stuff is nice, but it shouldn't be made the focus of too much attention. The fact is, the main strength of ACCESS.bus is that it is a single, open standard offering a solution to PC cable chaos. Energy spent arguing the bits and bytes will only detract from the true battle: overcoming the elephantlike inertia that characterizes the PC market.

The ACCESS.bus proponents have lined up quite a list of suppliers that comprise the infrastructure—cables, connectors, keyboards, mice, and chips—that must underlie any attempt to overcome the powers that be.

Likely, the next step will be the development of PCs that simultaneously support the old interfaces and ACCESS.bus, first with add-in cards and later on the PC motherboards themselves. From there, ACCESS.bus-only systems are just a short hop away.

I hope this move happens soon. When it comes to adding a port or swapping a cable, these old bones are

getting pretty tired of “assuming the position.” PC owners, rise up off your knees! Here's a chance to go one up on the Mac. 

Tom Cantrell has been in Silicon Valley for more than ten years working on chip, board, and systems design and marketing. He can be reached at (510) 657-0264 or by fax at (510) 657-5441.

CONTACT

Philips/Signetics Company
8 11 East Arques Ave.
Sunnyvale, CA 94088-3409
(800) 227-1817

For an ACCESS.bus developers kit, contact Sharon Baker at (408) 991-3518.

IRS

425 Very Useful
426 Moderately Useful
427 Not Useful

MORE CIRCUIT CELLAR PROJECTS!

GET YOURS TODAY!



The Circuit Cellar Project File, Volume 1 has over 200 pages of new and expanded hands-on projects and tutorials. The Computer Applications Journal's editors have chosen a dozen of the top projects from the Circuit Cellar Design Contest, independent submissions, and topresponse articles to make a book with something for every interest!

now only \$17.95! (includes domestic delivery)

Order your copy today!

*\$17.95 Visa, Mastercard, Check, or Money Order (US Funds Only)
(Add \$2.00 for delivery to Canada or Mexico, \$4.00 for delivery to other non-U.S. addresses)

The Circuit Cellar Project File... Volume 1

4 Park Street, Vernon, CT 06066 • Tel: (203) 8752199 • Fax: (203) 872-2204



**PCB ARTWORK
MADE EASY!**

"Are you tired of laying down tape or wire wrapping to make printed circuit art? End the nightmare with the PCBboards Circuit Layout Program for just \$99.00! Easily create single or double-sided circuit boards with professional results."

- Herc, CGA, EGA, VGA, & SuperVGA
- Use mouse and/or keyboard
- Built-in help screens
- Dip and Sip footprint libraries
- Build custom parts easily
- Requirements: DOS 3.0 or later, 1 disk drive, 384k of RAM, IBM-PC or compatible
- Copper flood for building ground planes
- Block move, copy, rotate, save, etc.
- Comes with dot-matrix & laser output
- Make your own films with 1x laser art
- Gerber and Excellon output available

Download Demos from our 24hr BBS at (205) 933-2954

PCBoards Software Prices: Add on Software:

PCBoards Layout	99.00	PCRoute Autrouter	99.00
Gerber and Excellon Drivers	49.00	PCRoute+ Autrouter	149.00
Plotter Drivers	49.00	SuperCAD Schematic	99.00
Demo Package (all software)	10.00	SuperCAD+ Schematic	199.00

Use PCBoards, PCRoute, and SuperCAD as a "system" to create pcb art from a schematic!
PCboards - 2110 14th Avenue South, Birmingham, AL 35205 / Ph: (800) 473-PCBS Fax (205) 933-2954

Power Code

PRACTICAL ALGORITHMS

John Dybowski

One fashion of the day is **power**—power CPUs, the power user, and power code to name a few. As a designer of embedded systems, I quickly gained an appreciation for the littlest things, and I hold the capacity for low-power operation in high esteem. Far from being merely intellectually agreeable, this attraction stems from the special needs of the equipment and instruments I scheme to resolve. With this preference in mind, I will now discuss power code-low-power code.

In embedded designs, low-power operation is often required. This need exists not because of a desire to reduce the electric bill, but often it is the result of the special demands made by remotely powered systems or systems that operate from battery power as either their primary or backup power source.

When power dissipation becomes a problem, replacing the power-hungry and often-used NMOS and LS circuit elements with CMOS parts is the simplest way to bring this predicament under control. The result of such a retrofit is a significant reduction in the power requirements of the system; however, further steps can be taken to gain additional power savings.

Despite the current emphasis on speed and power, many applications can be served using slower 8- and 16-bit processing units, which usually consume less power. Furthermore, even when a high volume of processing capability is required, necessitating a fast CPU, most programs are found to spend much of their time waiting for some event to trigger the heavy-

duty processing. The balance of the time is spent literally idling.

A direct correlation between the processing burden and the amount of power dissipated in a carefully orchestrated CMOS system exists. The trick here is to differentiate meaningful processing from a tight idle loop because the system consumes power based primarily on the frequency of logic transitions that the circuit components encounter. That is to say, if the system is running full steam doing nothing, power consumption will remain at a level commensurate with meaningful processing.

Before I describe some ways to accomplish this differentiation, briefly examining the nature of static and dynamic power dissipation in CMOS circuitry will be useful.

CMOS BASICS REVISITED

When a CMOS device is not switching and in a stable state, the *p*- and *n*-channel transistors don't conduct at the same time, so the conduction from *V*_{cc} to *V*_{ss} is pointless. Power dissipation in this state is extremely low because the leakage current typically amounts to several nanoamperes.

When clocked, CMOS circuits draw current in sharp spikes made up of two components. One is due to the charging and discharging of on-chip parasitic and load capacitances. The other component is the current that flows at the moment when both the *p*- and *n*-channel transistors are partially conducting. This component is further swayed by slow rise and fall times.

For one-shot circuits and gates configured as oscillators, additional dissipation is caused by the actual operation of these signals in a linear mode. The resulting "through current" is in addition to the normal supply current and causes power dissipation to increase linearly with the input rise or fall time. While a sinusoid is not the optimal waveform for a CMOS circuit, it is what crystal oscillators generate. In the case of on-chip oscillators, such as those commonly integrated onto microcontrollers and microprocessors, there is a crossover point at which reducing the operating frequency

In battery-powered systems, power consumption is of the utmost importance. It's not always necessary to build in complicated power-control circuits to reduce current consumption, though.

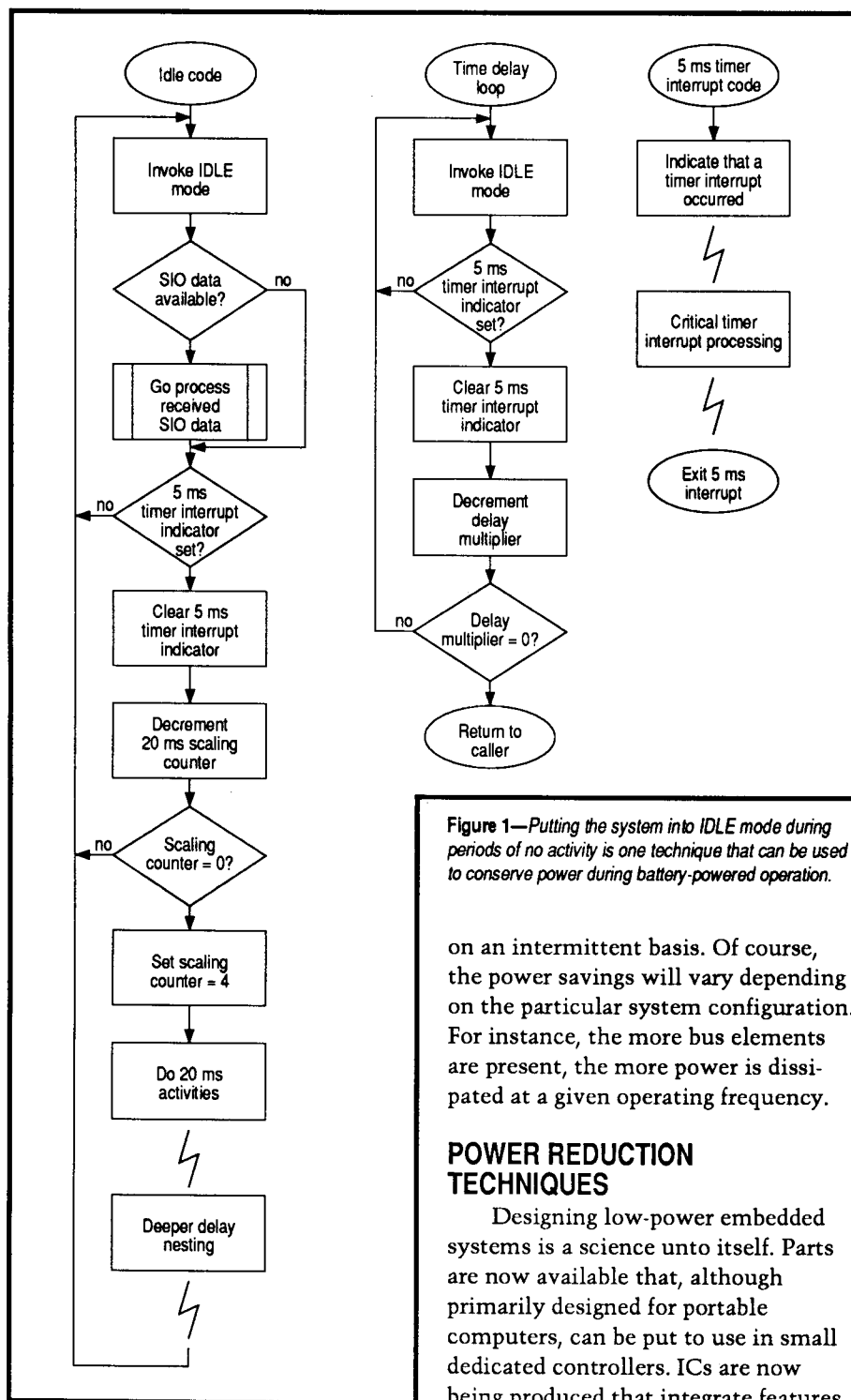


Figure 1—Putting the system into IDLE mode during periods of no activity is one technique that can be used to conserve power during battery-powered operation.

on an intermittent basis. Of course, the power savings will vary depending on the particular system configuration. For instance, the more bus elements are present, the more power is dissipated at a given operating frequency.

POWER REDUCTION TECHNIQUES

Designing low-power embedded systems is a science unto itself. Parts are now available that, although primarily designed for portable computers, can be put to use in small dedicated controllers. ICs are now being produced that integrate features on a single chip, such as synchronously rectified voltage regulators for logic power and logic-to-*n*-channel gate drive translators for controlling power to various electrical subsystems. Also in production are micropower standby voltage regulators for nonvolatile memory, comparators for detecting incoming power, and other desirable components needed to build a system capable of truly low-

power operation. Such ICs represent the ultimate foundation upon which to build a low-power system, implying a nontrivial design cycle.

Rather than get into the excruciating details of a complex design tailored for low-power operation, I'd like to describe some simple and easy-to-implement schemes that can be put to work immediately using standard hardware.

The essence of reducing power dissipation in a CMOS microprocessor-based design is limiting the number of transitions seen by the CMOS circuitry. You can make these limitations in a number of ways: electrical subsystems can be isolated (or perhaps shut down) when not needed, the basic CPU clock frequency can be varied in response to changing needs for processing power, or the processor can simply be operated on an intermittent basis. Intermittent operation is particularly attractive when running other processes dependent on the master clock, such as the system timer or serial communications. In this case, running the system in an intermittent mode circumvents the problems associated with running a variable frequency clock and its effect on the system peripherals.

The old adage that a typical program runs 10% of the code 90% of the time is true. It usually implies that a program spends much of its time idling, waiting for something significant to happen. During such moments is often the appropriate time to operate the system in a degraded mode in order to bring the power consumption down.

SYSTEM HALTING

This degraded mode of operation is achieved by suspending CPU operation, which then suspends external bus activity, thereby putting the system into a standby mode. This effect can easily be established by executing a HALT instruction. Most processors have such an instruction, though the mnemonics vary. For example, on the 80C31 the corresponding state is called *IDLE* and is entered by setting PCON.0. Let me describe *IDLE* mode briefly before moving along because it typifies the concept.

actually results in an increase of current consumption.

Following the logic of the preceding discussion, when using CMOS circuitry, power dissipation can be reduced by limiting the transitions seen by the logic circuits. This reduction can be accomplished by controlling the system operating frequency or by operating the system

In the *IDLE* mode, the CPU puts itself to sleep by gating off its own clock. It does not stop the oscillator; it just stops the internal clock signal to the CPU. Because the CPU draws 80 to 90% of the chip's power, this step represents a significant power savings in itself. Moreover, it results in the termination of all bus activity, further reducing power. The on-chip peripherals such as the timer, serial port, and interrupts continue to operate.

While the device is in the *IDLE* mode, ALE and \bullet PSEN emit logic high. If the device was executing out of internal program memory, ports 0 and 2 hold whatever is in the PO and P2 registers. When executing out of external memory, port 0 is left in a high-impedance state and port 2 continues to emit the high byte of the program counter. Because of this effect, care must be taken to avert a situation where the data bus lines are allowed to float or a chip-select line to an external RAM or PROM is left asserted, when using external memory components. Referring to the *IDLE*

mode signal states, this problem can be avoided by using ALE as a gating signal for the PROM chip-select signal or, in some cases, as the chip-enable signal itself.

There are two ways to terminate *IDLE*. Activation of an enabled interrupt will cause the hardware to clear PCON.0, terminating the *IDLE* mode. The other is by resetting the system.

In a system that is totally interrupt driven, shutdown is not a problem because an interrupt will result in an exit from the *IDLE* mode and will permit processing to resume. Often, such an approach is not feasible when some operations must be performed continually on a periodic basis. The trick here is to arrange the critical processes to be interrupt driven along with a free-running timer that triggers an interrupt service routine.

Using standard hardware running a 12-MHz 80C3 1 controller you can program one of the on-chip timers to generate an interrupt every 5 ms. Now, a 5-ms timebase may be a bit frequent

for the types of activities you may want to perform, but counting ticks and scaling this interval to some convenient timebase using software is not at all a problem. Say you want to scan a keypad and check some inputs, then 20 ms might be a good choice. Running an 80C3 1 at 12 MHz, servicing a timer interrupt, determining the interrupt event in the idle code, and decrementing and testing a counter should take on the order of 25 to 50 μ s, which adds little to the overall power consumption. Falling out of the *IDLE* state can be the result of any of the interrupts going off, so you need to include an indicator in the timer interrupt service routine to denote that event. If the occurrence of a timer interrupt is detected, service the scaling counter and proceed accordingly. If the interrupt was of some other type, then simply reenter the *IDLE* state.

On expiration of the scaling counter, the controller can do a keyboard scan, check the inputs, and generally look around before either

BCC52 BASIC-52 Computer/Controller

The BCC52 Computer/Controller is Micromint's hottest selling stand-alone single-board microcomputer. Its cost-effective architecture needs only a power supply and terminal to become a complete development or end-use system, programmable in BASIC or machine language. The BCC52 uses Micromint's 80C52-BASIC CMOS microprocessor which contains a ROM-resident 8K-byte floating-point BASIC-52 interpreter.

The BCC52 contains sockets for up to 48K bytes of RAM/EPROM, an "intelligent" 2764/128 EPROM programmer, three parallel ports, a serial terminal port with auto baud rate selection, a serial printer port, and is bus-compatible with the full line of BCC-bus expansion boards. BASIC-52's full floating-point BASIC is fast and efficient enough for the most complicated tasks, while its cost-effective design allows it to be considered for many new areas of implementation. It can be used both for development and end-use applications.

PROCESSOR

- 80C52-BASIC, 8-M CMOS microcomputer
- jumper-selectable conversion to 80C31/80C32 functionality
- 8K bytes ROM (MI BASIC interpreter)
- 256 bytes RAM
- three 16-bit counter/timers
- 32 I/O lines
- 11 MHz system dock
- 6 interrupts

MEMORY

- expandable to 62K bytes
- five on-board sockets
- up to four 6264 (8Kx8) static RAM
- either an 8K 2764 or 16K 27128 EPROM

Input/Output

- console I/O RS-232 serial port
- line printer RS-232 serial port
- three 8-bit programmable TTL-compatible parallel I/O ports using a 8255 PPI
- alternate console RS-422/RS-485

To Order Call

1-800-635-3355

Tel: (203) 871-6170

Fax: (203) 872-2204

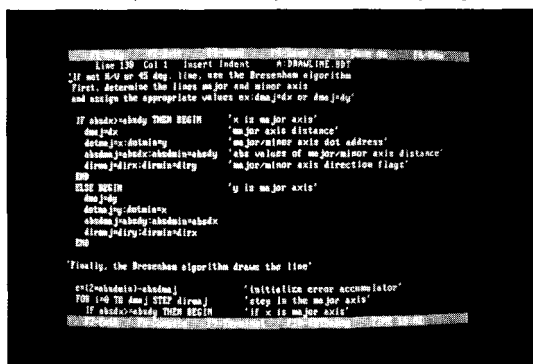
		Single Qty.	100 Qty.
BCC52	BASIC-52 Controller Board with 8K RAM	\$189.00	\$149.00
BCC52C	Lower-power all-CMOS version of the BCC52	\$199.00	\$159.00
BCC52I	Full industrial temperature range	\$294.00	\$220.00
BCC52CX	CMOS, Expanded BCC52 w/32K RAM	\$259.00	\$189.00

MICROMINT, INC. 4 Park Street, Vernon, CT 06066

BDT™ BASIC Developers Tool

ATTENTION: BASIC-52, BASIC-180 and BASIC-11 DEVELOPERS

Finally, an advanced development environment for BASIC single-board computers. BDT combines all the tools you need—including Editor, Compiler, Debugger and Terminal emulator in a powerful, fast, easy-to-use, and totally integrated package.



Editor

- Configurable keystrokes and colors • Memory resident text (FAST)
- Block move/copy/delete/read/write • Find & replace • Auto-indent

Compiler

- Structured programs: DO/UNTIL, WHILE/WEND, BEGIN/END • No line numbers
- Up to 20-character variable and label names • Subroutine LOCAL variables
- Five types of comments (including multiline) stripped during download

Debugger

- Up to 1000 BREAK/PASS points • Execution PROFILE • Up to 40 WATCH variables
- Integer variables WATCHable as DEC/HEX/BIN • All or partial array WATCH

Terminal

- Editor, file, and compile buffer download to SBC • File capture from SBC

Host PC Requirements

- 512K • One disk drive • One serial port • Mono.C/E/V/GA • DOS 3.x-5.0

Individual versions are available for BASIC-52 (BDT52), BASIC-180 (BDT180) and BASIC-11 (BDT11) SBCs @ \$199. SPECIAL OFFER—BDT-PAK™ combines all three versions @ \$489 for a savings of over \$100. Order today—offer expires soon...



40944 Cascado Place • Fremont, California 94539
(510) 657-0264 • FAX (510) 657-5441 • BBS (510) 657-5442



reentering the **IDLE** mode or doing some meaningful processing if some noteworthy event had been detected. This approach allows the other interrupt-driven processes to operate in a background mode on a continual basis, signaling the **IDLE** code only on completion of their assigned task. For example, an interrupt-driven SIO routine would assemble and acknowledge a data packet before indicating available data to the **IDLE** code.

Thus far, I've described running the system intermittently during idle moments to save power. This approach can be used for other wasteful functions like counting delay times as well. Delay loops can be structured to have a delay multiplier passed as an input argument. The delay code then disables the system by entering into an **IDLE** sequence. On emergence from **IDLE**, the timer-interrupt indicator is interrogated, and if it is set, the multiplier is decremented, otherwise **IDLE** is reentered. Once the multiplier counts down to zero, the delay is completed and the routine returns to

the caller. Figure 1 illustrates the basic premise behind this intermittent mode of operation.

SOFTWARE POWER DOWN

Although more limited in use and more radical in function, a mode of operation [or nonoperation] is featured on many microcontrollers that offers even greater power savings. On the 80C3 1 this software-invoked power-down feature is called **POWER DOWN** and is entered by setting PCON. 1. In the **POWER DOWN** mode, the CPU puts the whole chip to sleep by turning off the oscillator, so no internal clock is generated. The only exit from the **POWER DOWN** mode is via a reset.

In **POWER DOWN**, the on-chip RAM retains its data as long as Vcc is maintained. In this mode, the only current that flows is leakage, which is in the microampere range. Although the only way out of **POWER DOWN** is through a hardware reset, this mode may be appropriate if the system is to remain out of service for some time and is a simple alternative to adding

power-control circuitry. Of course, you'll have to provide some form of reset controller to the system, perhaps a push-button reset or some more sophisticated circuitry that can respond to an external stimulus. Operational status information can be held in RAM in this mode, so operating the system in a seemingly continuous manner is possible.

Fantastic solutions now exist to the power dilemma. However, examining the issues to be certain of your needs in order to apply the proper remedy is wise. Sometimes relief is only a hack away. □

John Dybowski has been involved in the design and manufacture of hardware and software for industrial data collection and communications equipment.

428 Very Useful
429 Moderately Useful
430 Not Useful

4 Configurations Available

The Heart of the Matter...

RTXC™

Real-Time Multitasking Executive

• HITACHI 6303 • INTEL 80x88/x86, 80x96, 80x51

• MOTOROLA 680x0, 683xx, 68HC11, 68HC16

• INMOS T400, T800 • ZILOG Z80/Z180

- Preemptive Scheduling
- Fixed or Dynamic Priorities
- Timeout on some services
- Configurable and ROMable
- **Intertask** Communications
 - Messages
 - Queues
 - Semaphores
- Memory Management
- Resource Manager
- Over 50 Executive Services Available
- System Level Debugging Utility
- System Generation Utility
- 450 Page User Manual

- Written in C
- Source Code Included
- No Royalties
- Technical Support
- Broad C Compiler Support
- Sensible License Agreement
- Most Popular C Compilers supported

Ask about ASSIST™
Our new PC Development Package Combination

One Time License Fee From \$995

Discounts for Multiple Licenses/Ports

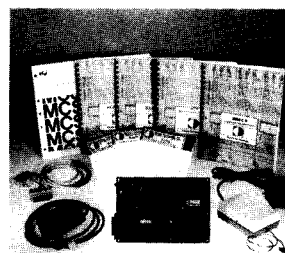
The only real-time kernel you'll ever need™

A.T. BARRETT & ASSOCIATES, INC.
111501 Chimney Rock, Houston, TX 77035
FAX 713/728-1049
Phone 800/525-4302 or 713/728-9688

FREE DEMO AVAILABLE

The \$595 Solution to 8051 System Development

PDK51



The PDK51 is a fully integrated hardware, firmware, and software system designed to help you develop your products quickly and cost effectively.

All you need to use the PDK51 is an IBM-PC/XT/AT or compatible. We supply the rest.

- SIBEC-II microcontroller board (8052AH-BASIC CPU) with 16K RAM
- BASIC interpreter source on disk
- SIBEC-II hardware manual with schematic
- Monitor/debugger ROM plus manual
- Monitor/debugger hardcopy source listing
- UL listed 5V power supply
- Programming power supply adjustable from 4V to 26V
- KERMIT communications program
- SXA51 8051/8052 cross assembler
- BTK52 BASIC programmers toolkit with tutorial
- 160 page BASIC Programming Manual by Systronics
- RS232 cable
- PDK51 tutorial
- Optional: aluminum case—\$45, monitor/debugger source—\$25, battery-backed real-time clock—\$39
- All components ATC manufactured to exacting standards and warranted for one year

PDK51 PLUS includes everything in the PDK51 plus Vers. 3 of our popular BXC51 8051/8052 BASIC compiler—\$800.

Call Now! 508-369-9556 or FAX 508-369-9549



Binary Technology, Inc.

P.O. Box 541 • Carlisle, MA 01741



CONNECTIME

conducted by Ken Davidson

The Circuit Cellar BBS
3001120012400 bps, 24 hours/7 days a week
(203) 871-1988—Four incoming lines
Vernon, Connecticut

On occasion, we've been known to have 'religious wars' on the Circuit Cellar BBS. In a religious war, the participants baffle over what they feel is the "best equipment" or the "best way of doing something" without much to base their stand upon except their own opinion or experiences. The old "PC versus Mac" battle is a classic example. Another classic example is what programming language is best. In the first discussion, we have such a war...

Msg#:54746

From: MARK DELAUNE To: ALL USERS

I am undertaking a project using the Siemens 80535 microcontroller (8051 derivative). The boss would like me to use BASIC so others could easily follow the code. I have never used BASIC, but am very pleased with assembly. Does anyone have any comments on why I should/should not use BASIC versus assembly language? I figured this would be the best place to find a few good tips.

Msg#:54757

From: ED NISLEY To: MARK DELAUNE

Well, I have an admitted bias _against_ BASIC, so take what I say with a moderate grain of salt...

The often-stated advantage of BASIC is that "anybody can follow the code," which is true for small programs and utterly false for anything nontrivial. There are several problems you run into when you build a big (meaning real-world) application in BASIC:

The biggest is the lack of local variables. All variables are global, which means you have to come up with increasingly bizarre names for the simplest routines; you _cannot_ assume that a name won't be reused by a caller because you'll be tinkering the code for quite a while and may well step on your own feet (a situation you won't detect for a _long_ time because the code _used_ to work just fine).

BASIC doesn't have a convenient way to structure the program; you don't have "real" subroutines that accept parameters and return a result. After a while the complexity of setting global variables before calling a routine and decoding the results gets old...

I suspect some of the BASIC compilers allow you to get around those issues, but at some point the "BASIC" stops being "BASIC" and becomes "just another block-structured language"—and you started out with BASIC because you _didn't_ want to use a block-structured language!

Although you can do it entirely in assembler, I'd suggest taking a look at any of the C compilers now available. There's a ton of C code lying around for my Firmware Furnace columns; take a look at it and see if it's completely illegible (even if you don't know C!)...that may be an indication of whether using C makes any sense.

What sort of project do you have in mind? There are some issues about timing and detailed control that are pretty straightforward in C/assembler that are pretty tough to pull off in BASIC...

Msg#:54819

From: MARK DELAUNE To: ED NISLEY

That's the kind of stuff that I want to hear—the pros and cons—so I can proceed in the "right" direction. The boss wants to network about 100 of the controllers together and I just don't know enough about it to say I can accomplish it all in BASIC. Any suggestions?!

Msg#:54871

From: ED NISLEY To: MARK DELAUNE

Now _that_ is a serious project...just out of curiosity, how are you going to handle the electrical connections? Good old RS-485 will handle 32 nodes, but going beyond that will take some custom hardware.

Msg#:55433

From: MARK DELAUNE To: ED NISLEY

For each 32 nodes there will be a "buffer" board to gather/poll info (says the consultant). So a system of many 32 node systems will be networked from this point to an IBM PC.

A discussion arose about not getting the info fast enough to the PC, so a coprocessor board was suggested to get the info as fast as possible from each of the "buffer" boards. Sound right? [Excuse my lack of knowledge.]

Is there an off-the-shelf package or some set protocol for micro networking that can be implemented quicker than writing one instead?

Msg#:55498

From: ED NISLEY To: MARK DELAUNE

Sounds like a big and complex system...good luck on getting it all working!

CONNECTIME

Given what you're trying to do, I suspect that there is no canned solution (particularly for 803 1s) that will meet all the design goals (fast, cheap, quick, easy...). You could use the INKnet interface (described in issues 10-12) if it suits your purposes; fact of the matter is that there aren't that many different ways to make it work.

I like the idea of a PC coprocessor card, but keep your eye on the ball...if it does nothing but collect serial data into a buffer, the task might be better done right on the PC so you have better control of what's going on. On the other hand, if the data rate is really high or you have multiple channels, the coprocessor will allow you to do useful work on the PC; it all depends on the details.

Msg#:54766

From: KEN DAVIDSON To: MARK DELAUNE

Such a problem. Most people are stuck with, "I know BASIC, but I need assembly to do ??? so need to learn it fast!" BASIC is much easier to learn than assembly.

If you're more comfortable with assembly, by all means write it in assembly. You'll only take a performance hit if you settle for BASIC. As Ed said, any program can be made readable or unreadable regardless of the language. A liberally commented assembly program with meaningful labels and a few macros can be far more readable than a BASIC program with a bunch of calls to absolute line numbers and variable names like "A," "QW," and "JP."

I wrote the entire HCS II Supervisory Controller in assembler with a multitasking kernel and I wouldn't have done it any other way. Anything else wouldn't have had the necessary speed and the executable code would certainly have been larger than the 8K it is now.

Tell your boss to leave the programming to the programmers.

Msg#:54830

From: KEN SIMMONS To: ED NISLEY

The newer BASICS for MS-DOS systems are more "structured" than regular interpreted BASICs. For instance, QuickBASIC allows local and global variables, separate procedures (SUBs and FUNCTIONs) that can be called recursively, user-defined variable types and other goodies, PLUS the advantage of "normal" BASIC syntax and coding. You can even directly run BASICA/GWBASIC programs [with minor alterations] you already have.

The only real drawbacks that I can see are difficulty in interfacing with assembly modules and the lack of "pointers" that C nuts love. However, the programs generated by QB, while not as small and "elegant" as C or Pascal, are nevertheless faster than if run on an interpreter.

I, personally, love QuickBASIC and think it's loads better than any interpreted BASICs out there.

Msg#:54872

From: ED NISLEY To: KEN SIMMONS

We're in violent agreement, but I submit that, to the extent that BASIC has turned into Yet Another Block Structured Language, you're giving up the immediacy of an interpreted environment without gaining much of the performance advantages of a truly compiled language.

Given the hyperthyroid PCs available nowadays, I think the performance issue is moot unless you're trying to use the thing as a controller instead of a computer. In that case you need every cycle you can get, so a simple way to bolt assembly language programs into your code is essential...

OK, lurkers, get out those torches...Flame On!

Msg#:54942

From: KEN SIMMONS To: ED NISLEY

I cannot disagree with you, Ed. Controller applications DO require that all times be as short as possible as well as every byte (bit?) of RAM being as precious as gold (remember the old CP/M days?).

I agree with you about today's PCs: they've followed the axiom of hard drives ("data will accumulate to fill available space") in that applications programmers have BECOME LAZY AND CARELESS in the coding! Why else do the "latest and greatest" require 33+ MHz 386s with 4+ megs of RAM?

I'd LOVE to see the "RAM crunch" of the early '80s happen again just to see if applications programmers come to their senses and realize that not everyone can afford the equipment to "support" their gluttonous software creations...

As for the "immediacy," interpretive BASICs are still EXCELLENT platforms for learning programming. Eventually, the person will "jump" to a compiled language IF his needs require it.

Yes, BASIC, like 01' DOS, is here to stay for a LONG time...

See, no flame from me... :-)

Msg#:54990

From: ED NISLEY To: KEN SIMMONS

Code does tend to fill the space available, but I'll put in a word for the defense: for tiny controller applications, you're going to have a 32K EPROM anyway...so it doesn't matter whether you have 12K or 31 K bytes of code!

Bottom line: if your code fits in the space available, meets the performance requirements, and can be maintained without too much hassle, why not use a high-level language and get the results out the door faster?

That said, I'm still in shock over the data in a recent PC Mag for Windows 3.1: something like 11 MB of disk

CONNECTIME

space. Their take on OS/2 is between 15 and 30 MB, but that includes the full-bore on-line **doc** for everything and a bunch of other programs and fonts...plus Windows!

Msg#:54835

From: GARY SMITH To: ED NISLEY

As a professional BASIC programmer, I would like to challenge any C or Pascal programmer in BOTH speed and code size. As a stand-alone compiler, the BASIC Professional Development System (BASIC 7.1) does take a lot of room. A simple PRINT statement will use about 20-30K. However, by using assembly language routines, this can be cut down to less than 1K! There is a very good assembly language library from Crescent Software that interfaces with the BASIC compiler.

The beauty of BASIC is it is fast and easy to learn (unlike C). It is structured, and I/O [i.e., screen, keyboard, etc.] is simple and can be very elegant(?). There is even support for **ISAM** files.

Just my two cents... ;->

Msg#:54873

From: ED NISLEY To: GARY SMITH

I'm tempted to take you up on that, but I'm already up to my eyeballs in Things To Do...

Actually, I think BASIC programs tend to grow at about the same rate as other **languages**; you get a big blob in the beginning and then incremental growth from there on out. C code can start out smaller for small functions, but probably grows at a faster rate as you add features. Might be a wash for programs of equivalent functionality...

Yo, Ken...this might make an interesting adjunct to the Design Contest. The challenge is to write a nontrivial controller program to defined specs of speed, size, processor, whatever...using whatever tools you think are appropriate to the task. Prizes for smallest, fastest, slickest...

Msg#:55035

From: JOHN CRUNK To: ED NISLEY

You're right on track, Ed. Microcontroller applications aren't that much different then their full-blown big-system cousins. The 80/20 rule usually applies, so I concentrate first on getting the application running then do some profiling and trim the 20% or so into assembler as required. Overall, I believe the time is well spent to concentrate on the big picture with a high-level language.

Msg#:55081

From: ED NISLEY To: JOHN CRUNK

I think experience counts for something, too. You have to pick the right "big picture" view so that the "little picture" is optimizable. Getting the overall algorithms right

counts for nearly everything: optimizing the daylights out of a bubble sort is the wrong way to go!

Msg#:54844

From: KENNETH SCHARF To: MARK DELAUNE

At the company where I used to work, we had quite a few microcontroller-based products using the 8073 tiny BASIC processor (National Semi., now an discontinued part). The time-critical stuff was written in assembly language and the routines were called from BASIC. In essence, the assembly routines were "functions" added to the BASIC language via "calls." The unfortunate part was that absolute addresses had to be used for these functions.

The assembler routines became a library and programs were built by stringing these calls together in BASIC. All menu functions (user interface) were built in BASIC, the guts and glory were done in assembler, math crunching [where speed didn't count] was done in BASIC [the 8073 had an integer math BASIC]. The same concept can be done a thousand times better on the 8052-BASIC chip. It has a much better version of BASIC, with floating point no less!

Most engineers who design computers and electronics have a very sparse chemical background. That doesn't stop the quest for knowledge, though, as you'll see in the next thread.

Msg#:52010

From: PAUL BOESE To: ALL USERS

I'm looking for information on fast oxygen sensing. I need a sensor that can respond quickly to the change in oxygen content of a person's breath. Chemical sensors are too slow. I've heard of paramagnetic O₂ sensors, but I am having difficulty locating some info. Any help would be appreciated. In medical terms, I need to know the End Tidal Oxygen content of a person's breath as compared to the inspired oxygen content.

Msg#:52483

From: DAVID PARRISH To: PAUL BOESE

We've been looking at oxygen sensors for calibrating incubators, so I might be able to help. First, you may want to check with a local anesthesiologist or anesthesiology or respiratory therapy department at a nearby medical school. They probably keep up with what's current in O₂ equipment. Second, call Ametek at (412) 828-9040 or HP's medical division for information on their systems. Third, be ready for sticker shock! Ametek's single-channel analyzer is almost \$7k and HP's multigas system is better than \$1 Sk!

CONNECTIME

Msg#:52758

From: PELLERVO KASKINEN To: PAUL BOESE

I do not know how fast the O_2 sensors used for car emission control systems are, but it looks like they would potentially meet your needs. If I understand, they still are "chemical" in nature, but not wet chemistry. Instead, they use a solid-state system based on some metal/oxide balance.

I think the first time I saw anything about these devices was over **10 years ago**. Zirconium-based design from Philips, the Dutch electronics giant. They have their semiconductor and passive components outlets here in U.S. nowadays collected under their parent name (used to be a bunch of different companies like Norelco and several others). I do not have the address here at home, will get back to you, if necessary.

Msg#:52887

From: ANDY SARNAT To: PAUL BOESE

When you say "oxygen content" (a term that usually refers to oxygen carried in blood) I presume you are talking about the ***concentration*** of oxygen in inspired and expired gas, which is expressed either as a partial pressure (in mmHg or other pressure units), or as a volume percent (dimensionless). For instance, at 1 atm = 760 mmHg the concentration of oxygen in dry room air is around 21% or 160 mmHg. (Sorry, I never got used to kiloPascals, the official SI unit.)

In operating rooms and intensive care units, inspired oxygen concentration is usually measured with polarographic O_2 analyzers. Oxygen is reduced at the cathode of an electrochemical cell known as a Clark electrode, generating a current proportional to O_2 concentration. The cell is isolated from the test medium by a membrane permeable only to gases. They're small and cheap, but not particularly fast: a step change in inspired O_2 takes several seconds to respond, maybe 30 seconds or a minute to reach a plateau. That's all you need for measuring inspired concentrations, but it would never do for tracking changes during the respiratory cycle.

Paramagnetic analyzers are based on the fact that O_2 , unlike other medical gases, is paramagnetic. That is, it is attracted into a magnetic field due to alignment of electron orbitals (atomic theory is not my thing-I better quit right there!) and thus tends to displace other gases present. There used to be an instrument called a Pauling meter, which placed a ball filled with nitrogen into a magnetic field, such that the sampled gas would displace the ball in proportion to its O_2 concentration, though I've only read about it. I don't think paramagnetic analyzers are used much now.

For FAST sensing of O_2 , the gold standard is still probably the mass spectrometer. They are used in some operating rooms for analysis of inspired and expired gases,

including O_2 , CO_2 , N_2 , N_2O , and volatile anesthetic agents, and they can give you a respiratory waveform. Big-time bucks, of course. They are typically washing-machine-size devices multiplexed to serve several ORs at once, though Ohmeda makes some "smaller" ones that will set you back only a few tens of kilobucks, and would only anchor a small yacht instead of the Queen Mary. Of course, if you can afford one, forget the research and just buy the yacht. :-)

Newer technologies for on-line gas analysis include (1) **Raman** scattering devices, such as the Rascal II monitor-I forget who makes it- which places the sample inside (yes, inside) a laser cavity and measures the scattering spectrum to identify molecular gases; and (2) photoacoustic devices, which illuminate the sample with various wavelengths chopped at audio frequencies and detect an acoustic signature based on absorption at those wavelengths. I'm not even sure if these are commercially available yet. And unfortunately infrared absorption devices, which are now the workhorses of fast CO_2 analysis, don't tell you much about oxygen.

Having said all that, are you sure you ***need*** a true end-tidal determination of O_2 in expired gas? Under many circumstances you can assume that end-tidal gas is in equilibrium with alveolar gas which is in equilibrium with mixed-venous blood which is in equilibrium with pulmonary artery blood, which is readily sampled in anybody with a PA catheter (i.e., your sicker ICU patients and all **open-heart** surgical patients). On the other hand, if you're after a measurement of oxygen extraction by the lungs (what goes in minus what comes out), you might be more interested in comparing inspired to 'average' expired gas concentration, which you get by collecting all the expired gas into a big bag over, say, one minute and simply measuring its final concentration, totally noninvasive.

I agree with contacting the Department of Anesthesiology at your nearest medical school for more info. Good luck!

Msg#:52901

From: SERGEI LUZHEFF To: ANDY SARNAT

I seem to remember an old way to measure CO_2 content in a person's breath that had a pretty fast response (1+ second). It might be what you're looking for. It's based on the fact that different gases have different thermal conductivities. Since you don't need to measure actual O_2 content, but only relative change after circulation through the lungs, you could do it indirectly by measuring the increase in CO_2 content (=DEcrease in O_2 content!), which is easier to do. CO_2 has a VERY different TC than air.

Such a method was formerly employed in machines used to measure human basal metabolism, derived from CO_2 production. In those machines, two tube sensors

CONNECTIME

formed two legs of a Wheatstone bridge, and the readings were made with galvanometers. They suffered from some drift (0.5% over a period of some 15 minutes). More modern measuring, compensation and power-supply systems might make the method more accurate and stable.

To make the sensors, take a glass tube, about 50 mm long and 0.6 mm in diameter. Stretch a thin platinum wire inside the tube so it lies coaxially to the tube, fairly well-centered and parallel to the tube walls. Fix the wire in this position, either by fusing it to the glass or securing it mechanically otherwise [conceivably, the sensor tube could be miniaturized].

In use, the platinum wire is gently heated by means of stabilized, calibrated, capacitor-coupled AC from a low-impedance source. There are also DC-coupled/AC-blocked/filtered sensing connections at the ends of the wire, used to measure the wire's resistance at any given instant by means of some sensitive instrument.

With a small air pump (aquarium pump?), take a continuous sample of the subject's exhaled breath, dry it through CaCl_2 , and circulate it through the tube sensor in a steady, calibrated stream.

The DC output of the sensor could be processed, digitized, or whatever to suit. A small, insulated thermocouple or other variation on the theme is also possible. Or the wire could ITSELF be a thermocouple. In fact, one of those old RMS thermocouple AC/DC converters could give you what you need if you can manage to attach (epoxy?) two small feed tubes without destroying it. Ha!

A second tube sensor is used for reference, to measure either ambient air, or subject's exhaled and chemically-processed (e.g., with CO, removed) breath. This is used for comparison with the unprocessed, exhaled gases circulating through the other tube. In both cases, moisture is removed prior to sending the gases through the tube sensors.

With such sensors and sufficiently stable, accurate, and sensitive electronics, it should be possible to make decent (1-2%) determinations of carbon dioxide (and, therefore, of O₂ consumption) in the subject's breath, based on the fact that CO₂ will conduct heat away from the wire at a different rate than air (oxygen/nitrogen), causing measurable change in the wire's temperature and, therefore, in its resistance.

While, as you say, chemical -determination_ (=reading/measuring) methods are usually slow, chemical _separation_ of gases is not slow at all-it can be nearly instantaneous! I wouldn't confuse the two.

Msg#:53352

From: PELLERVO KASKINEN To: SERGEI LUZHEFF

I can testify to the positive working of thermal conductivity analyzers. I did not think of the roundabout way of

using them for the subject on hand, when the original question was an oxygen analyzer. But I have been using the thermal-conductivity-based analyzer we bought from Matheson for argon and helium gas leak detection in our welding systems. While doing so, I have noticed what gases cause a detection and can verify that CO₂ does so, but that water does also. As you say, the prefiltering is absolutely necessary. I would add that there may be some substances created by anaerobic activities that might be found in the exhaled air that should also be checked before the CO₂ content would be trusted.

Unlike the platinum wire you suggest, I would suggest tiny NTC resistors. They are faster and more sensitive. That is what Matheson uses in their new, improved sensors. The one we have (was some \$850 at the time we bought it) has a response speed of a few seconds. Their new analyzer is specified to have a response time of less than 1 second.

For those people possibly interested (and most notably, the person starting this thread!), here is the address:

Matheson Gas Products
30 Seaview Drive
Secaucus, NJ 07096-1587
(201) 867-4100

By the way, the instrument is Model 8065 Hand Held Gas Analyzer.

We invite you 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 (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, or 2400 bps.

ARTICLE SOFTWARE

Software for the articles in this and past issues of ***The Computer Applications Journal*** may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only \$12.

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 875-2199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

I R S

431 Very Useful

432 Moderately Useful

433 Not Useful

STEVE'S OWN INK



Cost is in the Eye of the Beholder

As working engineers, we at the Computer *Applications* Journal are as interested in the applicability and appropriateness of the editorial content of this magazine as you are. When Jeff proposed investigating "instant PC boards" in his column, I felt the ramifications of his investigation might have more importance for our own efforts than just meeting his editorial obligations. Let me explain.

The circuits and projects presented here are not just cursory examples that are published and forgotten. They have to work because most are intended for commercial use. Unfortunately, the only way to properly evaluate a commercial design is to make a PC board (there can be considerable operating differences between hand-wired and PC board circuits). Finding a faster and more cost-effective way to make prototype PC boards would allow us to meet our expanding editorial needs.

But, what is fast and what is cost-effective?

The steps in making a PC board are fundamentally the same regardless of who makes the board. Only the actual fabrication of the board differs between "personal" and "commercial."

The schematic is first drawn on a CAD package, converted to a net list, and then transformed into a plot file using a PC layout program. We usually transmit the resulting file by modem to a photo shop where films are photoplotted. The films are then sent to a PC board fabricator who makes two prototypes and Federal Expresses them back to us. The time from the modem call to PC boards in hand is usually 10 days. The cost for two prototypes (-30 sq.in. each) is usually under \$500 and the quality is first rate.

In the case of Jeff's newly tried technique, the PC layouts are laser printed onto the PC board and then you just etch the boards in the usual way. Sounds great, doesn't it?

Well, unless you do a lot of board etching, the chemicals and trays have to be set up each time. The board has to be etched to find out whether the "iron on" laser print was properly aligned and transferred or the whole sequence has to start again. Finally, the board has to be manually drilled and the through-holes connected.

In my opinion, the new system Jeff explored made good editorial sense for us to present. There are many readers who are looking for low-cost PC prototypes who might be satisfied by this method. On the other hand, if I temporarily change hats and view the situation as a manufacturing manager, I'd have to reject hand-making prototypes except in cases where delivery time was the sole acceptance criterion.

Jeff is no stranger to PC board fabrication, having had a previous job in that field. By his own admission, it took him a whole week to get one good prototype that was drilled and through-hole connected. On a pure cost-per-man-hour engineering-time evaluation, any board that took more than 3 or 4 hours to produce exceeded the cost of going to our outside vendor. Jeff didn't think it would ever take less than a full day to make the kind of board we generally would need.

To properly evaluate "personal PC board prototyping," you have to first determine what value you place on time and quality. If time is of the essence (i.e., a new PC board has to be in this machine on the customer's site tomorrow at 8 A.M., or else), then the question is moot. The value of time supersedes all other valuations.

Similarly, if you place no value on your time (you're just experimenting or have nothing else productive on the agenda), then any board you produce is of positive value and less expensive than contracting a PC board house.

I classify our needs as middle of the road; we're often rushed, but quality and performance ultimately dictate procedure. The first time Jeff spends a week making a board this way it's written off as an editorial experiment. But unless the technique is improved significantly beyond present methods, claims of \$5 PC board etching kits have no validity because these components are not the main expense-producing ingredient. Except as noted, man-hour costs cannot be disregarded. A \$200 commercial prototype board is economical if compared to the cost of paying an engineer a week to do the same job. Until Jeff tells me he can make a board in 2-3 hours with plated-through holes, I know one outside vendor who shouldn't fear losing us as a customer.