

CIRCUIT CELLAR **I N K**

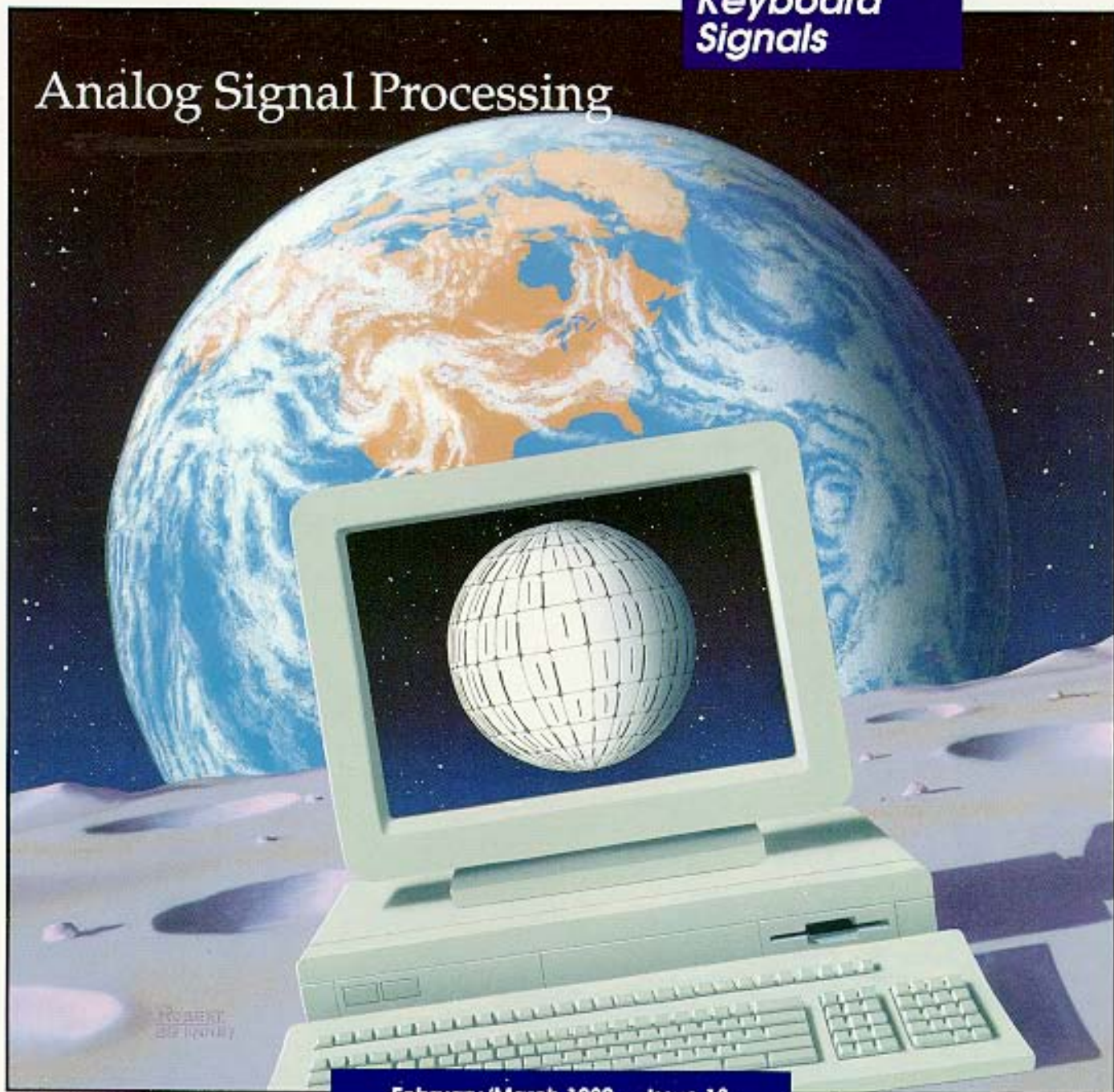
THE COMPUTER  
APPLICATIONS  
JOURNAL

**Working  
with DSPs**

**Build a  
25-MHz  
Digitizer**

**Translating  
Keyboard  
Signals**

Analog Signal Processing



February/March 1990 — Issue 13

\$3.95



# Dateline: Las Vegas

**W**ords are wonderful in their ability to form visions and emotions in our minds. Such a simple word as "home" can, with its associations, bring a tear to a weary traveler's eye. Other words, take COMDEX as an example, can produce fevered visions of technology oozing from the very pores of casinos while armies of disembodied feet plod through aisles stretching to infinity.. .

Can you tell that I just finished "doing" COMDEX? A week of strolling past the latest in microcomputers has left me tired and footsore, but encouraged about the future of the small computer industry.

## MS-DOS MARCHES ON...

...or more accurately, the Intel 80x86 family keeps rolling. The perception of the 80x86 as an architecture for desktop business machines only is in the process of radical change. While there were scads of computers for running spreadsheets and word processors, I saw considerable evidence of the 80x86 becoming a leader in the application and control markets. In one form or another, the Intel architecture was the dominant force at almost every level.

First, the high end: You couldn't swing a dead cat without hitting an 80486 motherboard at COMDEX. Of course, finding an 80486 microprocessor was another matter entirely, and all the 80486s to be found had major bugs of one flavor or another. No matter, the manufacturers have decided that 80486 is the way to go, so the clones were out in force. Most of the "clone" motherboards used the ISA (PC/AT) bus, and many proclaimed EISA compatibility. Since the EISA spec is still a darker secret than the avionics in the B-2, I don't know whether or not any of the EISA machines work as promised. In any event, the 80486 machines are being sold as the microcomputers that will finally bring about the demise of minicomputers. Sales folks talk about the 80486 as the perfect processor for LAN servers, but the winks and nudges on the side say that no high-powered engineer, programmer, or financial analyst is going to be satisfied until a fire-breathing 80486 computer with 600-MB hard disk, 16 MB RAM, 8514A graphics, and \$20,000+ price tag is enshrined on his or her desk.

The favorite pastime of computer industry pundits is proclaiming the death of the 80286. Since everyone knows how deficient the processor is, you wouldn't have expected to see any major moves in the 80286 market. In my view, though, one of the most exciting products of the show was the 80286 plug-in module from Mitsumi. The

module puts an 80286, 512K memory, and all of the "glue" chips for a working AT-type computer into a package only slightly larger than an 80486. The engineers at Mi tsumi see the 80286 playing an increasing role in embedded applications, and they are pushing the idea of the entire computer as a plug-in module. According to the engineers, fax machines have already been designed around their 80286, and more products are on the way. I think they may be on to something with this approach, and we're working on an article showing how to use the module in applications.

A side-effect of the 80286's move into embedded applications may be the demise of the 80186. The '186 has been a favorite of designers who wanted to develop software under MS-DOS but needed more oomph than an 8088 could muster. Products like the Mitsumi module make it much harder to justify the 80186 approach.

Finally, the time has arrived for XT-class machines to take a serious role in the control application market. I know: "Real" computer people hate the architecture, loath the operating system, and can't abide the lack of built-in I/O. None of that matters when you can buy a full IO-MHz XT motherboard for \$48 (quantity 1). At that price, economics begin to cast a rosy glow over most shortcomings.

Several months ago I wrote about the move toward using PC-type computers for control applications. The trends I saw at COMDEX tell me that the trend is accelerating. Good software development tools and low hardware costs are teaming up to make the Intel/MS-DOS combination irresistible to a lot of developers.

What does all of this mean for other microprocessors and controllers? Other architectures won't wither just because of new competition. We'll see a push to bring software development tools to a new level of functionality and friendliness. Engineers and programmers have tolerated obtuse commands and low functionality in microcontroller development tools because "that's the way it's always been done." Users accustomed to the "look and feel" of the best MS-DOS compilers and other development tools, will push the software vendors to give them better tools. If the vendors are smart, they won't argue.



FOUNDER/  
EDITORIAL DIRECTOR  
Steve Ciarcia

PUBLISHER  
Daniel Rodrigues

EDITOR-in-CHIEF  
Curtis Franklin, Jr

PUBLISHING  
CONSULTANT  
John Hayes

ENGINEERING STAFF  
Ken Davidson  
Jeff Bachiochi  
Edward Nisley

CONTRIBUTING  
EDITORS  
Thomas Cantrell  
Jack Ganssle

CONSULTING  
EDITORS  
Mark Dahmke  
Larry Loeb

CIRCULATION  
COORDINATOR  
Rose Manse/la

CIRCULATION  
CONSULTANT  
Gregory Spitzfaden

ART & PRODUCTION  
DIRECTOR  
Tricia Dziedzinski

PRODUCTION  
ARTIST/ILLUSTRATOR  
Lisa Ferry

BUSINESS  
MANAGER  
Jeannette Walters

NW PRODUCTS  
EDITOR  
Harv Weiner

STAFF RESEARCHERS

Northeast  
Eric Albert  
William Curle w  
Richard Sawyer  
Robert Stek  
Midwest  
Jon Elson  
Tim McDonough  
West Coast  
Frank Kuechmann  
Mark Voorhees

Cover Illustration  
by Robert Tinney

# CIRCUIT CELLAR **INK**

## THE COMPUTER APPLICATIONS JOURNAL

In This  
Issue...

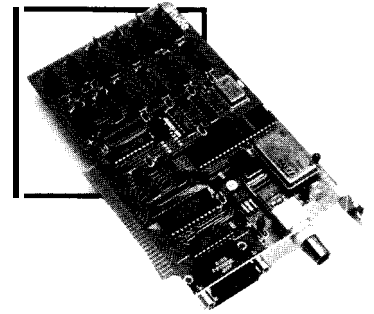
**21**

### Building etude:

#### Part 1 — A 25-MHz Analog-to-Digital Converter for the PC Bus

by J. Conrad Hubert and Dick Hubert

It's hard to imagine doing serious data acquisition without a serious digitizer. The etude 25-MHz digitizer board is just that. The first of two parts deals with the hardware side of the story.



**42**

### The BCCH 16

#### Part 2 — Porting a Multitasking BASIC to the H 76

by Jack Ganssle

The BCCH 16 is more than hardware: a multitasking BASIC compiler lets you put its power to use. Jack Ganssle looks at the process of porting a compiler between similar processors in the wrap-up article on the BCCH 16.

## DEPARTMENTS

### Editor's INK

Dateline: Las Vegas \_\_\_\_\_ 1  
by Curtis Franklin, Jr.

Reader's INK—Letters to the Editor \_\_\_\_\_ 5

NEW Product News \_\_\_\_\_ 8

Visible INK- Letters to the INK Research Staff \_\_\_\_\_ 12

### Firmware Furnace

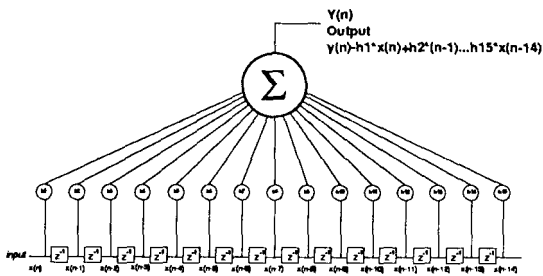
An Exercise for the Student \_\_\_\_\_ 52  
Building Software from the Ground Up  
by Ed Nisley

### From the Bench

ENTION . . . ATTENTION . . . ATTENTION ... ATT \_\_\_\_\_ 60  
Building an LED Moving Message Display  
by Jeff Bachiochi

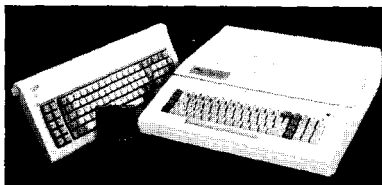
**30** Digital Signal Processing  
Part 1 — An Introduction  
by Dean McConnell

Digital Signal Processors (DSPs) have changed the way we look at analog signal problems. In the first of two parts, Dean McConnell discusses the most commonly used transforms and how they're applied.



**14** Building An IBM PC Keyboard Translator  
An 8031-based System for Code Translation  
by Bill Curlew

You know the problem: You have a favorite keyboard, but it only works with one of your systems. Wouldn't it be nice if you could take "old reliable" with you as you move from system to system? With the keyboard translator developed by Bill Curlew, you can!



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

The schematics provided in Circuit Cellar INK are drawn using Schema from Ovation Inc. All programs and schematics in Circuit Cellar INK have been carefully reviewed to ensure that 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 the 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.

CIRCUIT CELLAR INK (ISSN 08968985) is published bimonthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 875-2751. Second-class postage paid at Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions \$14.95, Canada \$17.95, all other countries \$26.95. 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 2099, Mahopac, NY 10541 or call (203) 875-2199.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 2099, Mahopac, NY 10541.

Entire contents copyright 1990 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.

Advertiser's Index	65
Silicon Update	
Earthshaking Chips	68
<i>A Report from the Second Microprocessor Forum</i>	
by Tom Cantrell	
Software by Design	
Memory Management on the HD64180	72
by Jack Ganssle	
ConnecTime—Excerpts from the Circuit Cellar BBS	76
Conducted by Ken Davidson	
Steve's Own INK	
An Analog State of Mind	80
by Steve Ciarcia	

# letters to the Editor

## TAKE CONTROL

I really enjoy reading *CIRCUIT CELLAR INK*. After dealing with multiprocessing super-mini control computers that only a rocket scientist would understand while at work, it's great to know there is still a forum for basic 8- and 16-bit applications that don't use megabytes of memory. I enjoy reading the tutorials on embedded systems, both hardware and software. The tutorials are of great value when trying to integrate systems and people at work.

After staying on the sidelines for quite some time, I now have something to contribute. This is a reply to Wilson Snyder concerning motor control and track switching ("ConnecTime," *CIRCUIT CELLAR INK* # 11).

Bruce Chubb has written a book called "Build Your Own Universal Computer Interface," ISBN O-8306-3122-4, published by Tab Books. I have seen it in several bookstores around the LA/Orange County area so I don't imagine it would be too hard to find.

The book describes (what else?) a universal interface that will connect almost any computer through a bus or serial port, analog and digital data acquisition and control, interface testing, and control software, and ties it all together with a project involving model railroad control.

Kevin K. Asato  
Gardena, CA

## EVEN FARTHER BEYOND ASICS

I appreciated Tom Cantrell's article on PLDs ("Beyond ASICs") in *CIRCUIT CELLAR INK* #10. Indeed, the billion-dollar PLD industry has grown dramatically since its humble beginnings barely a decade ago, and the growth is not slowing. The good news is that it is now becoming economically practical (and desirable) to design with PALs and other PLDs at home!

The programmable logic devices themselves are cheap, and PLD "starter kits" are now available for around \$50 that provide good PC-based development software (albeit limited to only a few device types). The final link—the device programmer—is also now available from several sources in the \$300 price range.

With a few PLDs around, and the appropriate development software and hardware, home designers are no longer at the mercy of their stock of "jellybean" parts laying around. Need a 4-bit counter or a 9-input AND gate with both inverted and noninverted outputs? Just program your PLD to do what you need! Wiring up projects is also much simpler with PLDs, since a single PLD can often functionally replace several 7400-series SSI/MSI devices. If your project doesn't work, don't change the circuit, just change the PLD programming.

For readers interested in more information about PLDs, I suggest they check out my new book from Howard W. Sams & Co., "Programmable Logic Designer's Guide," ISBN O-672-22575-1.

Roger C. Alford  
Dexter, MI

## MORE ON MILDEW

I read with interest the letter from Guyana and your response in *CIRCUIT CELLAR INK* #11. In June I attended the first International Conference on Computing and Missions at Taylor University in Upland, Indiana, where the same problem was discussed. The missionaries reported that Tupperware makes an ideal container for preventing formation of mildew on diskettes, especially with the addition of silica gel as you suggested.

That conference was the first place I had heard of the problem, so I'm not surprised at the discussion you had.

Duane Vosburg  
Binghamton, NY

## ...AND FROM THE OTHER SIDE

I wish to protest in the strongest and most vociferous terms the editorial focus and content of *CIRCUIT CELLAR INK* #11, for which I have been forced to pay and accept, as a subscriber.

With issue 11's "Build Your Own 386 Clone" (yeah, I'll run right out and do it), its "32-bit Multitasking Control-

ler" (surely what every reader needs), and its "Design Your Own 32-Node Network for Your EGA AT" (clearly everyone's goal), issue 11 is a serious breach of the spirit, essence, and soul of the Circuit Cellar articles originally appearing in BYTE.

Sir, it was the minimalist nature of the Circuit Cellar articles which made the hardware they described charming, practical, comprehensible, affordable (in both time and money), and, in a word, accessible.

That is what I imagined I was buying when I subscribed to CIRCUIT CELLAR INK. Please get back on track.

M. Edward  
Madison Heights, WI

P.S. I understand you are seeking authors. I believe a buddy of mine has an article entitled "How I Networked my Dozen VAX 11/780s to my Cray in my Spare Time For Only \$560,000." Would you be interested in seeing it?

Thank you for *taking* the time to write with *your concerns* over the content of CIRCUIT CELLAR INK #11. It's always helpful to *hear* from our readers.

We try, here at CIRCUIT CELLAR INK, to listen to our readers. When a number of readers write with the same request, that request receives additional priority. So if was with the 80386SX project, the single most requested topic over the last 18 months.

I assume, from *certain* subtle clues in your letter, that you appreciate projects based on the 8031 and 8052 microcontrollers. There have been a number of projects using these controllers, and rest assured there will be more such projects in the future. (I might mention that the control network article in #11 was based on the 80C52-BASIC microcontroller.)

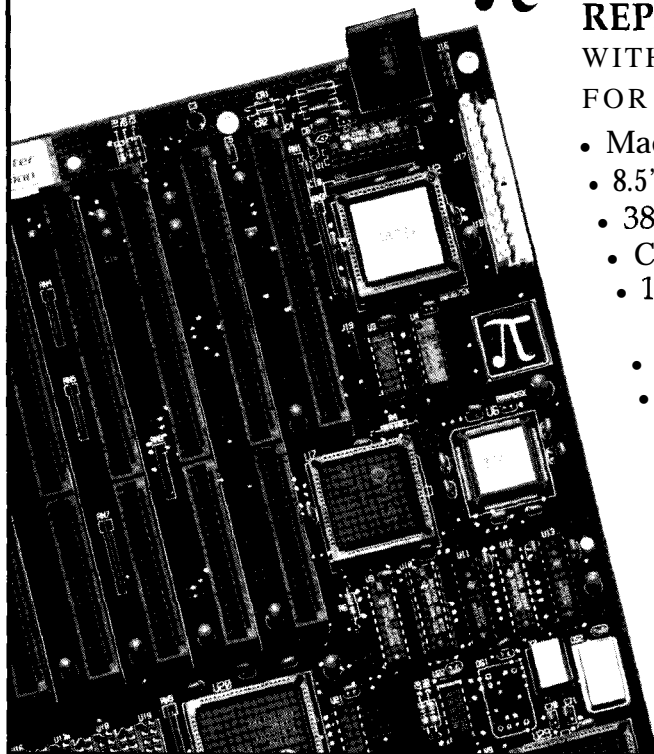
Our goal is to present a variety of projects balanced in complexity and expense. Just as the Circuit Cellar column in BYTE presented both large and small projects (see the projects for the SB180, CCAT, and Mandelbrot Engine for examples of "large" projects), CIRCUIT CELLAR INK will present projects based on processors and controllers ranging from the cheap and "accessible" to the challenging. There are readers who need both, and we will try to keep as many readers happy as is possible.

The "spirit, essence, and soul" of CIRCUIT CELLAR INK are defined by an attitude toward presenting working, practical projects and tutorials, regardless of the processor, controller, operating system, or bus used. While your letter makes it sound as though you will enjoy our planned projects based on the 8031, 80C52-BASIC, and 68HC11 microprocessors, I should probably warn you that we are also planning projects based on the 68000, 80286/32532, and other "large" processors and controllers.

Curtis Franklin, Jr.

P.S. Please have your buddy forward an outline of his project. If we can work a hack to dangle an 8033 controller off one of the nodes, we might just have a winner.

## NOW 386™ PERFORMANCE IS EASY AS $\pi$



REPLACE YOUR XT/AT MOTHERBOARD  
WITH A 2 MB PI 386SX™  
FOR ONLY \$699!

- Made in America
- 8.5" x 13" — Fits XT, Baby or Full AT & Tower cases
- 387SX™ and Shadow BIOS Support
- Choice of Award™ or Phoenix™ BIOS
- 16MHz or 20MHz SX™ models available, up to 8MB RAM onboard
- Three years parts and labor Warranty
- 25-33 MHz 386DX™ motherboards also available with/without SRAM cache

 **Pi Computer  
Systems, Inc.**

1030 Earlysville Forest Drive  
Charlottesville, VA 22936

Orders only 800-666-9248



Information and **804-978-3917**  
Tech Support; **Fax: 804-978-3906**



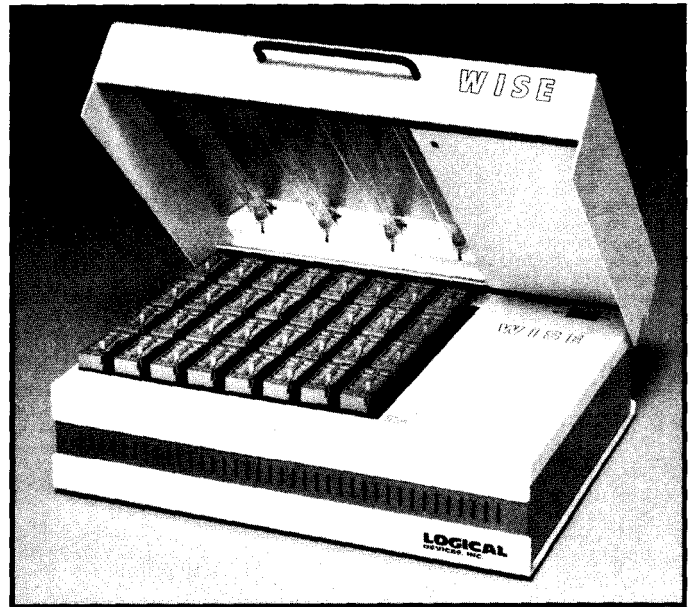
## EPROM ERASER/PROGRAMMER SYSTEM

The need for a separate gang programmer and eraser lamp has been eliminated with the announcement of the WISE EPROM **Erased Programmer** system by Logical Devices Inc. WISE allows the user to insert EPROMs into the programming sockets and initiate an erase/program cycle with a single push button. EPROMs are first erased by exposure to an ultraviolet (UV) light source located in the top lid of the unit. A patented Proprietary Intelligent Erasing Algorithm determines a safe erasure time. After erasure, the programmer automatically deactivates the UV light and begins the programming cycle.

With conventional programming systems, EPROM erasure is performed as a separate step. Devices are placed under an eraser lamp for an estimated erasing time. The devices are then removed from the eraser and placed into a programmer. The programming system must check to see if the devices are blank to avoid errors. If they are not, the devices must undergo another erasure and check cycle.

While the UV erasing system is operating, the programming section of the unit will read the data in each one of the 32 chips on a repetitive basis to monitor the exact time to erase all of the EPROMs. The system will abandon faulty chips that will not erase within a specified time in order to avoid system hangups. Erasure time is done at Vcc margins and a safety erasing time is allowed to avoid fading of EPROMs.

As a further time saver, the WISE system also uses a solenoid-activated, auto-load Zero Insertion Force (ZIF) socket handle flipper. This eliminates the need to individually lift



and close the ZIF socket handles. The WISE **system** is priced at \$9995.

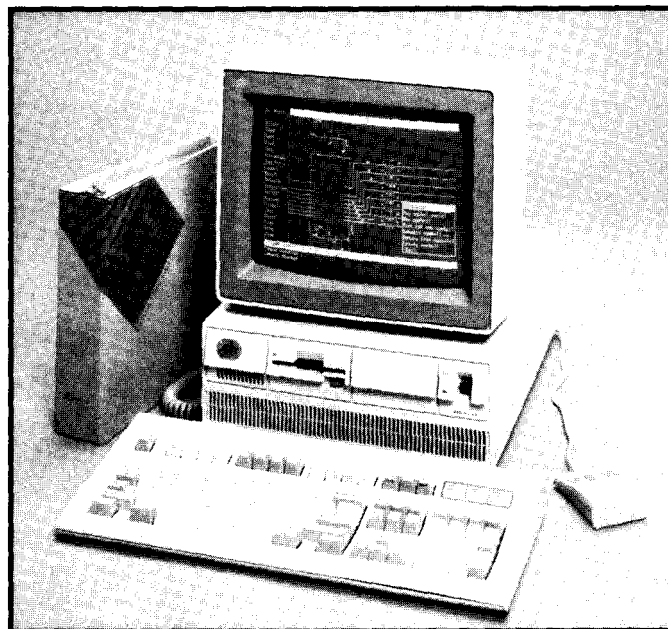
Logical Devices, Inc.  
1201 NW 65th Place  
Ft. Lauderdale, FL 33309  
(305) 974-0967

## LOW-COST SCHEMATIC DESIGN PACKAGE

Phase Three Logic Inc. has announced a low-cost schematic design package developed for the broad electronic-design automation market. The **CapFast CF640 Schematic Design Package** runs on the 640K-byte main memory of PC/AT, PS/2 and compatible systems.

The CF640 uses an incremental design concept combined with an overlay memory management system to handle large designs with multiple pages. It can handle an individual A- through E-size schematic page and extract a net list for a SO-page design with C-size pages (approximately 10,000 design elements).

In addition to the incremental net list extractor,



the CF640 Schematic Design Package includes an intelligent packaging program that automatically assigns reference designators and pin numbers to physical packages, Xilinx and Abel interfaces and symbol libraries,

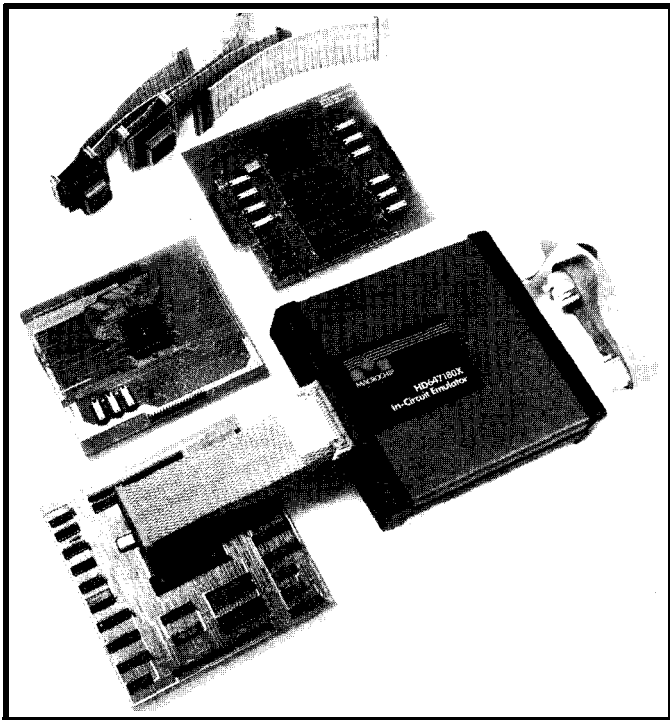
and a PADS-PCB back-annotator. It also includes the symbol creation editor, an enhanced symbol library composed of more than 5500 parts, an incremental part list program, Spice interface tools, Susie digital simulator

interface, and interfaces to major PCB-CAD systems.

The CF640 provides up to four split-screen windows for simultaneous global and local viewing and features infinite zoom levels. Hardcopy printer and plotter support is also provided. The system may be upgraded to other CapFast software products because all of their design databases are 100% compatible. This growth path extends to Sun Workstation Unix versions.

The CF640 Schematic Design Package is priced at \$295 and includes a 30day money back guarantee. One year of software updates and support is included at no charge.

Phase Three Logic, Inc.  
1600 N.W. 167th Pl.  
Beaverton, OR 97006  
(503) 645-0313



## IN-CIRCUIT EMULATORS SUPPORT ENTIRE Z180 AND HD64180 FAMILY

Macrochip Research is now offering low-cost in-circuit emulators that support the newest members of the Z180 and HD64180 family. The **MR7180X** emulates Hitachi's HD647180X ZTAT (Zero Turn Around Time) device, and the **MRZ180S** emulates Hitachi's new HD64180S HDLC serial controller device. Both emulators connect to the PC's serial port and provide real-time, nonintrusive in-circuit emulation with no wait states at clock speeds up to 10 MHz. On-chip refresh is maintained at all times. Both emulators feature 64K bytes of emulation overlay memory, mappable on any 8K boundary within the first 64K physical address space.

A complete monitor/debugger, provided in firmware, allows the user to assemble and disassemble code; examine registers, memory, and I/O; set breakpoints; and single step the target processor ROM or RAM without the use of special driver software on the host. Intel hex or straight binary files can be downloaded from the host directly into target memory or overlay memory at serial rates up to 38.4 kbps.

The emulator allows up to four hardware breakpoints that can be placed anywhere in the target system's 1 megabyte physical address space. Additionally, each breakpoint can be qualified to break only on memory read, memory write, I/O read, I/O write, opcode fetch, or interrupt acknowledge cycles.

The **MR7180X** and **MRZ180S** emulators are shipped with an 84-pin PLCC emulation plug, 6' RS-232 cable, Z180 macro assembler and Development Environment software (MS-DOS), and user's manuals. Options include HD64180 cross-assemblers for the Macintosh and Amiga computers, and an 84-pin PLCC to 28-pin DIP adaptor for programming the HD647180X's internal EPROM using an ordinary EPROM programmer. Each of the emulators can be field converted to emulate another HD64180 family member at a relatively low price.

The **MR7180X** and **MRZ180S** list price is \$1995 each.

Macrochip Research, inc.  
1301 N. Denton Dr., Suite 204  
Carrollton, TX 75006  
(214) 242 0454

## 68000 SIMULATOR ALLOWS FULL MEMORY ADDRESSING

Testing and debugging Motorola 68000 programs before hardware is available is now possible with the **PseudoMax** 68k simulator from Pseudo Corp. The developer can watch the program execution via machine windows as the simulator single steps or free runs through the program code. Each register, the stack, I/O ports, and blocks of memory can be monitored.

User-definable screens enable the designer to customize the simulator. Each screen can contain up to 40 machine-specific windows. The simulator uses an internal demand-page virtual memory addressing algorithm that allows simulation of the full 16-Mbyte addressing space. Simulations include input/output

interrupts, traps, and exceptions. Other features include unlimited breakpoints, memory mapping, and a trace file feature that allows selective recording of the simulator session for later analysis.

**PseudoMax** requires a 512K IBM or compatible PC with mono, CGA, EGA, Hercules mono display and MS-DOS 2.1 or greater. The introductory price of the **PseudoMax** 68k is \$100. The **68k** cross-assembler is \$50, and a **68k disassembler** is available for \$100. The Developer **Pack** consisting of all three products is \$200.

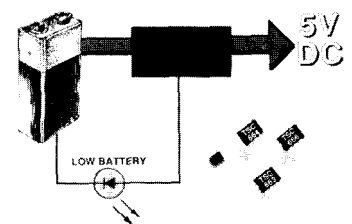
Pseudo Corp  
P.O. Box 1423  
Newport News, VA  
23601-0423  
(804) 595-3703

## CMOS MICROPOWER VOLTAGE REGULATORS

A new family of low-cost micropower voltage regulators, featuring an optimized wafer fabrication process, has been announced by Teledyne Semiconductor. The TSC663, 664, and 666 are lower cost pin and functional replacements for similar devices from other manufacturers.

The TSC663 positive voltage regulator, and TSC664 negative voltage regulator are designed particularly for battery-powered applications. They feature low standby current for low quiescent power, and a shutdown pin for external control. In the shutdown-mode, quiescent current is less than 12 microamps. The regulators are designed to be used either as fixed 5-V regulators with no external components, or as adjustable regulators with output voltages from 1.3 V to 16 V

### CMOS MICROPOWER LINEAR VOLTAGE REGULATOR



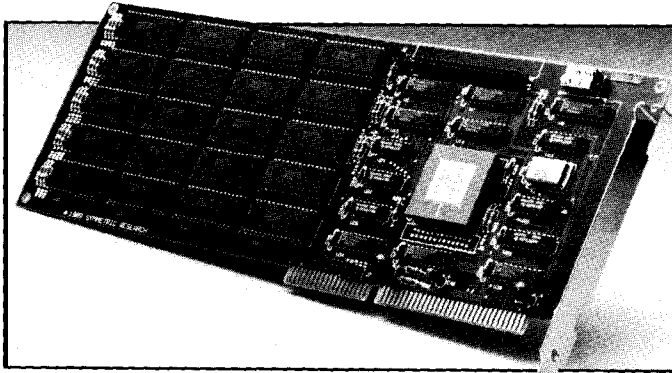
TELEDYNE SEMICONDUCTOR

set with just two resistors. The input voltage can be from 2 V to 16.5 V, and the output current capability is 40 mA. These CMOS ICs are very low noise, wide bandwidth linear regulators, and have output current limiting built in.

The TSC666 is a positive voltage regulator with a low-battery detection circuit to detect the input voltage dropping below specification and provide a signal to the system to warn of impending power loss.

Teledyne Semiconductor  
1300 Terra Bella Avenue  
Mountain View, CA  
94039-7267  
(415) 968-9241





## PC/AT DIGITAL SIGNAL PROCESSOR

Thirty-two-bit Digital Signal Processing development is now available with the Symmetric Research **DSP32C PC/AT coprocessor board**. The board features the DSP32C chip from AT&T and performs 25 million floating-point operations per second (MFLOPS). It provides state-of-the-art real-time performance for filtering and numerical calculations.

The on-board memory, which is socketed and can be populated to 640K bytes, is dual ported for simultaneous access by the DSP32C and the PC/AT bus. These features allow the user to run deep buffers and save them to disk while the DSP32C continues running.

For interfacing to external devices, the board features a 32-bit bidirectional parallel port that can be accessed at full processor speed. A header provides connections to the high-speed serial port of the DSP32C for interfacing to CODECs and other serial acquisition devices.

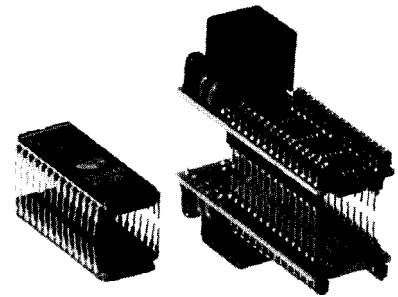
Software included with the board consists of an assembler, monitor debugger, and math libraries callable from C and FORTRAN. The source code for all software is included, along with numerous example programs demonstrating the use of the board. Fractal computations and Fast Fourier Transforms (FFTs) are included among the demos. The benchmark for a 1024-point complex FFT, including data on and off the board, is 15 milliseconds with 100-nanosecond static RAM chips.

The base coprocessor board including all software is **\$950**. A full 640K of 100-ns memory is an additional **\$400**.

Symmetric Research  
15 Central Way, Suite #9  
Kirkland, WA 98033  
(206) **828-6560**

## 2764 ROM EMULATOR

A 2764 ROM **Emulator** for use with the IBM PC and compatibles has been introduced by Parallax Inc. The device emulates most of the popular 8K x 8 EPROMs including the 2764, 2764A, 27C64, and Intel 87C64. The emulator saves considerable time in developing ROM code. Instead of being programmed into EPROMs, software is downloaded to the emulator and run on the target system within seconds to allow quick code development.



Surface-mount technology makes the 2764 ROM Emulator highly reliable and only slightly larger than the 2764 that it replaces. The emulator accepts SRAM for normal operation, or EEPROM for nonvolatile use. It connects to the parallel port of the PC via a modular telephone cable and downloads 8 Kbytes in 2-9 seconds. A tristate reset output restarts the target system after downloading. HC/HCT logic is used for compatibility with CMOS and TTL systems.

The supplied software includes a command-line downloader and a full-screen editor. The command-line downloader provides a way to download files directly from DOS and may be called from within batch files for automated assembly and downloading. For more involved work, the full-screen editor may be used to edit and download files. The emulator software accepts Intel Hex, Motorola S, hex, and binary files, and is written entirely in 8086 assembly language for quick, dependable operation.

The Parallax 2764 ROM Emulator is available without memory for \$129, with 100-ns SRAM for \$149, and with 250-ns EEPROM for \$159. An adapter for the 87C64 is available for \$49.

Parallax, Inc.  
6200 Desimone Lane, #69A  
Citrus Heights, CA 95621  
(916) 721-8217

## PORTABLE CD-ROM DRIVE

The portable office becomes a step closer with CD Technology's introduction of a portable CD-ROM drive with a 600-megabyte storage capacity. The **Porta-Drive** facilitates the portability of massive amounts of information. It allows instant access to large databases such as legal and medical libraries, parts catalogs, and marketing data. With its optional battery pack, the Porta-Drive can be used anywhere with any IBM or Apple Macintosh portable computer.

The Porta-Drive, a high-performance CD-ROM drive manufactured by Toshiba, measures 2" high by 5.8" wide by 9.5" deep and weighs only 4 pounds. It uses industry-standard media and cartridges for universal hardware compatibility. Its total capacity is 683 megabytes with an average access time of

350 ms. The PC and Macintosh software driver allows compatibility with all CD-ROM products. A SCSI interface (required if there are no other SCSI devices installed) allows seven drives to be linked, and they can be stacked to minimize desktop space. The battery pack offers an estimated computer time (intermittent access) of 4 hours with a 6-hour charge time.

The price of the **Porta-Drive** for either the IBM PC or Apple Macintosh is \$895. **The optional battery pack is \$200.**

Custom Design Technology, Inc.  
780 Montague Expressway, Suite 407  
San Jose, CA 95131  
(408) 432-8698

# Letters to the INK Research Staff

## VISIBLE INK

answers:  
clear and simple

### GETTING STARTED WITH THE 8031

I recently picked up *CIRCUIT CELLAR INK #8* and really enjoyed it. I particularly enjoyed the articles on the 8031/8052 embedded controller and working with LCD displays. I have been working with the 80C52-BASIC because I have not yet learned to program in assembly language. Can you recommend a good starter for the 8031?

Jon Williams  
Riverside, CA

Thanks for the nice comments about *CIRCUIT CELLAR INK*. We are glad that you enjoy it, and hopeful that it will continue to fuel your interest in embedded controller design and programming.

There is not the wealth of programming tutorials available for the 8052 family of microcontrollers (8031, 8032, 8051, 8052) that exists for other processors such as the Z80 or the 8086. You are probably aware that Intel, the original manufacturer of the 8031, publishes the "8-Bit Embedded Controller Handbook." This book is a reference manual to the 8051 family and is valuable information to have, but not necessarily the greatest for the beginning programmer. Signefics also publishes the "8051 Microcontroller Users Guide" which some people feel is more readable than the Intel version. Signefics is a second source for the 8052 family products. Also take a look at the DDT-51 that was published as a *Circuit Cellar* project in the August and September 1988 issues of *BYTE*.

We know of one book that describes designing projects around the 8031 and developing the embedded software to run them. "Mastering Digital Device Control," ISBN 0-89588-346-5 by William G. Houghton, is published by Sybex Books. This book presents a number of examples of how to interface various devices to an 8031 and includes the complete design, hardware and software, for a stand-alone EPROM programmer that communicates with a host computer via an RS-232 serial port.

coprocessor chips? My particular interest is in a coprocessor board for a PC/XT that can be reprogrammed more easily than the chips in the Mandelbrot engine and provide more processing power for numerical calculations.

I am aware of the special coprocessor boards, such as the Transputer-based boards, but their prices place them beyond serious consideration. It seems to me that with some attention to software development, it would be possible to achieve the same processing power with a much lower investment in the hardware.

Patrick L. Durusau  
Jena, LA

We do not know of any commercial boards that use multiple 8087s as parallel numeric processors. One reason for this is due to the nature of the way an 8087 works: It really is best when it is tightly coupled to an 8088. You can think of the 8087 as the floating-point silicon that didn't fit in the 8086's case. As such, you would have to emulate the 8088's signals and you might as well use an 8088 rather than go to the added expense of other hardware.

Actually, while software design is critical to the mission, hardware is usually far cheaper than software. This is true across just about all computer types and sizes. Witness the difference between the Macintosh and the IBM PC: The Mac has fewer chips on the logic board than an IBM PC has on the CGA graphics board alone. On the other hand, the software involved in running a Mac is far more complex than DOS for the PC. You know which machine is more expensive, right?

For now, the coprocessor boards you mention are the main choices for numerical computation. Of course, given the rate of hardware improvement, today's supercomputer will be available in just a few years at the local department store.

### COPROCESSOR HELP

Are you aware of any commercially available designs for a coprocessor board using multiple 8087-1s or similar

### IRS

201 Very Useful  
202 Moderately Useful  
203 Not Useful

# Building An IBM PC Keyboard Translator

*An 603 7 -based System for Code Translation*

Yes, I admit it. I'm a tag sale junky. Take me out back and shoot me, 'cause I don't want to be cured. And to make matters worse, I love mucking about with micros (as long as I can get them cheap, of course). So I have about 12 different systems hanging around by basement, from my first system—a hand-wrapped Altair 8800A look-alike—to my latest: a Colony Data Systems XT clone with 30-MB hard drive.

Unfortunately, these systems have something in common besides the (cheap) price I paid for them. Every one of them uses a different keyboard, with various layouts and "feels"! ARGHH, I hate having to remember stuff like what set of arcane keystrokes is required to create a control-Q, or whether the key marked "Del" is really a 7FH, 08H, or FFH. Lord knows I have better things to remember, like what time the new Star Trek comes on in my area on Saturday, and did I feed the cat at all this week? Besides not wanting to deal with the key differences, I do get used to a certain touch, regardless of whether it's pounding or stroking the keys.

Recently this problem came to a head when I tried to configure some word processing software for my daughter. She has an old Apple II+, and I was using my Franklin Ace 100 (yes folks, the infamous 100) to set up

the software for her. After an extended period spent struggling to recall what was where on that blankety-blank keyboard, I gave up in disgust. I had become addicted to the IBM-style keyboard attached to my clone, with its typeamatic action, typewriter feel, function keys, special numeric pad, and cursor controls.



Now to be honest, the Franklin keyboard on the Ace 100 has a very nice layout, but the feel isn't the same, and it's kind of dangerous to pull the thing out of the computer and put it in your lap. Besides, why should I have to worry about junk like keyboards in this wonderful computer age? That's when I decided to let a computer solve the keyboard problem that the multitude of computer keyboard styles had created in the first place.

## ...AND ONE FOR ALL

I set out to create an interface that would allow me to use my IBM PC-style keyboard with my other com-

puters, mainly the Apple, Franklin, and Altair machines. The IBM PC interface uses a serial-type of data stream with special keyboard codes, which I'll talk more about in a bit. The Apple and Franklin use an ASCII-encoded, parallel-port-attached keyboard, and my Altairs can use either ASCII parallel or RS-232 serial keyboards.

The interface converter presented here, which I call TRANSKEY, converts the encoded IBM keyboard code output into ASCII bytes. The ASCII bytes can then be sent to a host computer through either an 8-bit parallel interface (for use with, say, an Apple II), an RS-232

serial interface (for use with my Altairs), or both.

Using this converter allows me to define each key code from the keyboard as an 8-bit byte to be presented to the target computer. Thus, other character codes besides ASCII could be supported. A relatively straightforward software change would allow you to get multiple ASCII codes from a single IBM key code. An example of why you might want this would be to provide multiple-byte cursor control sequences.

The TRANSKEY system was developed on an RTC31 microcontroller, though an RTC52 may also be used. The final unit could even be built into a single 8751 processor with the con-

trol program and tables in the internal EPROM. [Editor's Note: For more information about the RTC31 and RTC52, see "From the Bench" in issue #8 of CIRCUIT CELLAR INK.1

The combination of serial and parallel ports available on the 8031 makes it a good choice for this type of system, but other controllers like the Zilog Z8 could have been used. In fact, I got the idea for this project from some work I had done with the Z8, using it in another device related to IBM-style keyboards. The Z8 system provided key code inputs to the IBM PC based on input from a slide projector hand control or an infrared transmitter. The device allowed the normal keyboard to remain attached and available at the same time. The research for the Z8-based project was used in the development of the TRANSKEY system presented here.

## THE IBM KEYBOARD

IBM, in its infinite wisdom, has not one, but at least three unique kinds of keyboard interfaces. These change as you move from PC/XT to PC/AT to PS/2. The main differences between the keyboard types are the serial data byte format and the way key codes are identified. TRANSKEY is designed to work with the PC/XT- and AT-compatible keyboards which are most prevalent at (you guessed it) PC faires and flea markets. These keyboards use two different signaling and code identification standards, known in the IBM lingo as Mode 1 and Mode 2. Both modes use a special bidirectional serial interface composed of a clock line and a data line. I will not be discussing the bidirectional operation of the interface in this article, since it is not implemented by the TRANSKEY system.

## BASIC CODE TRANSMISSION

To transmit a bit, the keyboard places the appropriate logic level on its data line, then signals that the data line should be sampled by moving the clock line from a logic 1 to a logic 0. This is very similar in operation to the way a synchronous modem interface

operates. The data line is valid from before the falling edge until after the rising edge of the clock signal.

Bit transmission continues until all the transmitted bits have been sent to the processor. The transmission of individual bits is the same regardless of the mode in which the keyboard is operating. Only the number of bits in a byte and the sequence of codes differ.

## KEYBOARD MODE 1

In a Mode 1 transmission, each keyboard code is made up of nine bits. The first bit is called the start bit, and is always a logic 1. Eight data bits follow. Simple keyboard interfaces use an 8-bit serial-in, parallel-out latch with an overflow line to interface the keyboard with the computer. The overflow line on the latch is set to a

Knowing whether a key has been released is how the shift, control, and alternate functions are handled. The receiving system might see the left shift "make" key code, then the "A" make and break codes, the "B" make and break codes, and the "C" make and break codes. Finally the break code for the left shift key would come in. It is up to the receiving system to remember that the shift key was being pressed while the other key codes came in.

Typeamatic action is simply the transmission of the "make" code over and over again, and works the same way in Mode 2.

## KEYBOARD MODE 2

Keyboard Mode 2 data transmission aligns much more closely with an RS-232-like serial interface's method

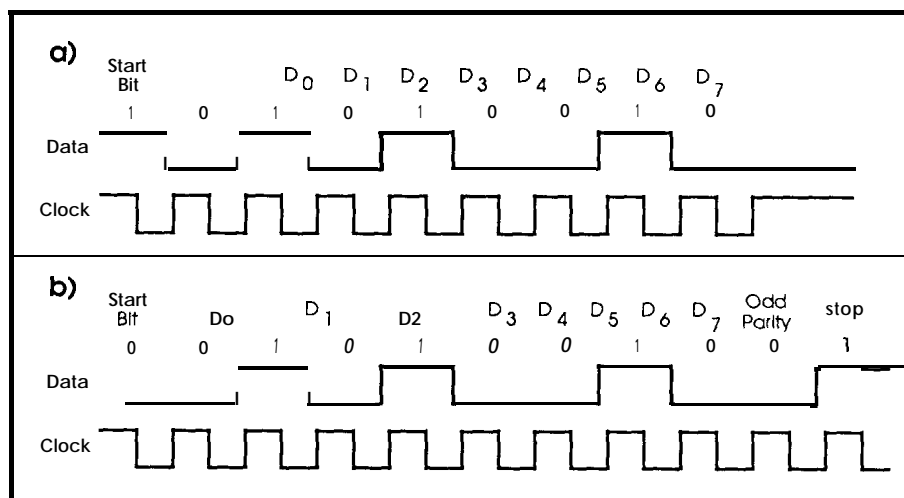


Figure 1—*a)* Key codes in mode 1 keyboard transmissions include a start bit and eight data bits. *b)* Mode 2 transmission adds a parity bit and stop bit to the end.

logic 1 when the start bit has been shifted through the latch. At this point, the eight bits of data are on the latch's parallel output lines. The low-to-high transition of the overflow bit triggers an interrupt, which causes the computer to read the 8-bit byte, and then the latch is reset.

The Mode 1 keyboard can transmit a maximum of 128 distinct codes. Any byte with a value from 0 to 127 is considered a "make" code (i.e., a key has just been pressed). Setting the high-order data bit indicates a "break" code, which means that the key has been released.

of transmitting bytes, and the make/break signaling has been changed to allow for more possible key codes. In Mode 2, signaling each key code is done with an 11-bit word, as opposed to the 9-bit word used in Mode 1. The timing differences between the two modes are shown in Figure 1.

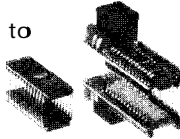
Mode 2 transmission begins with a logic 0 start bit. The start bit is followed by the eight data bits that comprise the actual code being sent. The data bits are followed by an odd parity bit to provide some level of data integrity. Last and least comes the stop bit.

# DEVELOPMENT TOOLS

For the IBM PC and Compatibles

## 2764 ROM EMULATOR

Appears as 2764 to target system. Connects to PC parallel port.



- Only slightly larger than an actual 2764
- Plugs into target ROM socket and connects to PC parallel port via modular telephone cable
- Accepts 8K x 8 SRAM or EEPROM (non-volatile)
- 3-state reset restarts target after downloading
- Uses HC/HCT logic for CMOS & TTL compatibility
- Loads Intel Hex, Motorola S, hex, and binary files
- Command line software can be run from batch files for automatic downloading after assembly

\$129 (with 128K memory)    \$149 (with 120ns SRAM)    \$159 (with 250ns EPROM)

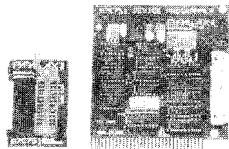
## 8051 FAMILY ASSEMBLER

- Works on all 8051 derivatives
- Supports standard Intel syntax
- Allows local labels and include files
- Labels may be up to 32 characters
- Generates assembly listings
- Outputs Intel Hex

\$95

Quick • Dependable • Clean Operation

## 87C751 DEVELOPMENT PACKAGE



### 87C751 Programmer & 8051 Family Assembler

- Programs Signetics 87C751 and 87C752 micro-controllers
- Handles EPROM, encryption key, and security bits
- Loads and saves Intel Hex, ASCII hex, and binary file formats

Dev. Package-DIP \$259 (includes choice of DIP adapter)	87C751 DIP adapter \$38
	87C751 PLCC adapter \$79
Dev. Package - PLCC \$299 (includes choice of PLCC adapter)	87C752 DIP adapter \$38
	87C752 PLCC adapter \$79

# PARALLAX

(916) 721-8217

Parallax, Inc.  
3200 Desimone Lane, #69A  
Citrus Heights, CA 95621



California residents add 6.5% sales tax  
Shipping: No charge for UPS ground,  
\$7.00 for UPS 2nd day, \$15.00 for UPS next day

Reader Service # 144

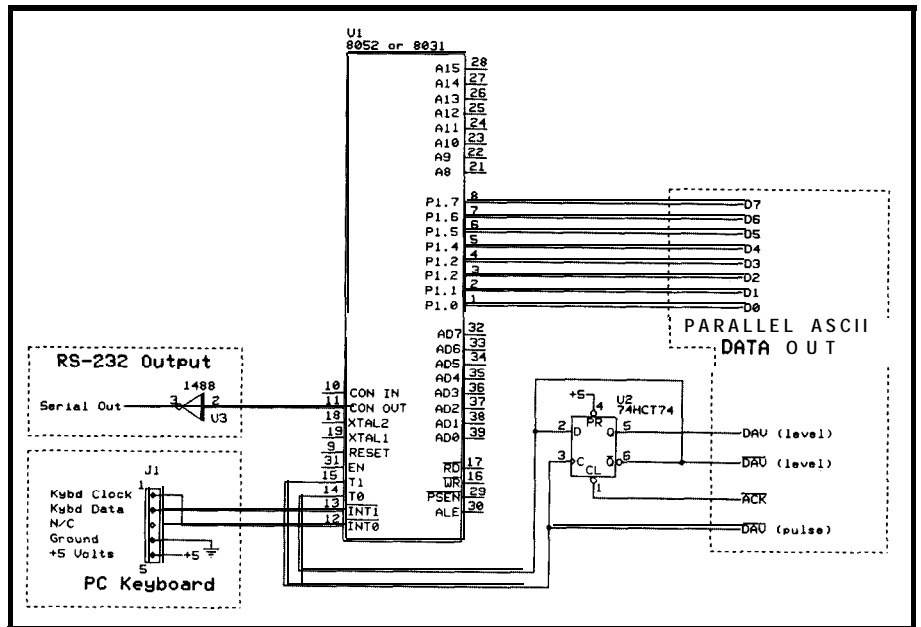


Figure 2—The keyboard translator was designed around a typical 8031-based microcontroller with the additional interface hardware shown here. See the article cited in the text which describes the RTC31 for a more complete 8031 design.

Another change in Mode 2 was the inclusion of a timeout feature on the AT keyboard interface controller. The controller starts a timeout clock at the beginning of each code being received. If the entire code is not received within 2 ms, the receive operation is aborted, and an error bit is set in the keyboard controller's status word.

The final difference we care about in Mode 2 deals with how the key code data is represented to the receiving system. The high-order bit is no longer used to indicate "make" versus "break" codes. Instead, a unique code, FOH, indicates the release of a key. Thus, while make codes can be sent in one byte, a break code requires two: the break indicator (FOH) and the code for the key being released.

The use of this coding scheme allows a greater number of keys to be supported, and gave IBM an opportunity to enhance the keyboard interface command set. An example of a keyboard command would be the code that turns the Num Lock light on or off at the keyboard. This command would be sent to the keyboard by the AT after BIOS decides how the reception of the Num Lock key code should affect the LED. Other commands include retrying the last transmission, invoking keyboard self-tests, and setting the repeat rate for the typematic keys.

## THE TRANSKEY SYSTEM

TRANSKEY performs data reception, byte translation, and retransmission. There are three interfaces, one for the keyboard input, and two for ASCII output. The ASCII output can be picked up over the serial port, which is the internal serial register of the 8031 microprocessor, or as an 8-bit parallel byte output on port 1. Some lines from port 3 are used along with a flip-flop to provide either level- or pulse-type data-available signals to the receiving device, and to allow the 8031 to know when it is OK to transmit another byte.

## KEYBOARD INTERFACE

Keyboard input is brought into the 8031 through two bits of port 3, as shown in Figure 2. The falling edge of the keyboard clock line drives an interrupt line in the processor, asking it to sample the data line. All keyboard input is handled in an interrupt service subroutine, illustrated in Figure 3. The software keeps track of what stage in byte assembly we are in based on the keyboard mode in use, and enforces timeouts and validity checking on the incoming bits. After a valid key code has been assembled, it is moved into a 256-byte ring buffer for process-

ing by the mainline routines. If the buffer is full at the time a code is received, the code is discarded.

## SERIAL INTERFACE

Serial output of the translated ASCII codes is accomplished through the 8031 internal serial register (SBUF). This register's output can be sent out to a bit on port 3, and is taken through a TTL-to-RS-232 converter IC.

While the serial port has the capability of running as an interrupt-oriented device, I have chosen to run it in the polled mode in this system. The two reasons for that were: it reduces the complexity of the system software, and since I require that both interfaces be clear before sending another byte, I would end up polling a status indicator somewhere anyway. The TI interrupt indicator is used as a polled TBMT (Transmit Buffer empty) line, which the main routine checks before attempting to write to the serial output register.

The speed of the serial transmission is controlled by an on-board timer. The initialization routine selects timer values based on the lower four bits of the parameter byte passed into the system at startup time.

## PARALLEL INTERFACE

The parallel interface handles transmission of the S-bit ASCII code to systems that expect all eight bits at once. The "port" is really made up of all of port 1, which is used to output the data, and three bits of port 3, which are used to control handshaking of the data to the outside world.

An external latch (flip-flop) is used to provide high or low levels of data available (DAV or DAV\). The latch is driven by three bits of port 3. Port 3 provides a bit which gives a high-to-low pulse when the data at port 1 is valid for input. The flip-flop changes state at this point, and provides the level-oriented data-available lines. In addition, the DAV\ line is brought back into the 8031, where it is used as a transmit buffer empty indicator. This allows the 8031 to wait for the attached computer to acknowledge the

reception of the data before it is overwritten by the next byte.

Because of the way the Q\ line of the flip-flop is run back into the D input, the 8031 can reset the flop to a known state by outputting **one** or two pulses on the DAV\ strobe line.

## MAIN SOFTWARE ROUTINE

The main driver for the system has several responsibilities. First, the serial, parallel, and keyboard interfaces are configured and initialized.

At this point, the two buffers used to hold data (key codes received and ASCII codes awaiting transmission) are empty, and the system loops around looking for work to do. The major structure of the software is shown in Figure 4. [Editor's Note: Software for this article is available for downloading from the Circuit Cellar BBS, or on Software On Disk #13. See page 78 for downloading and purchasing information.]

Outputting data to the ASCII interfaces is given top priority, which

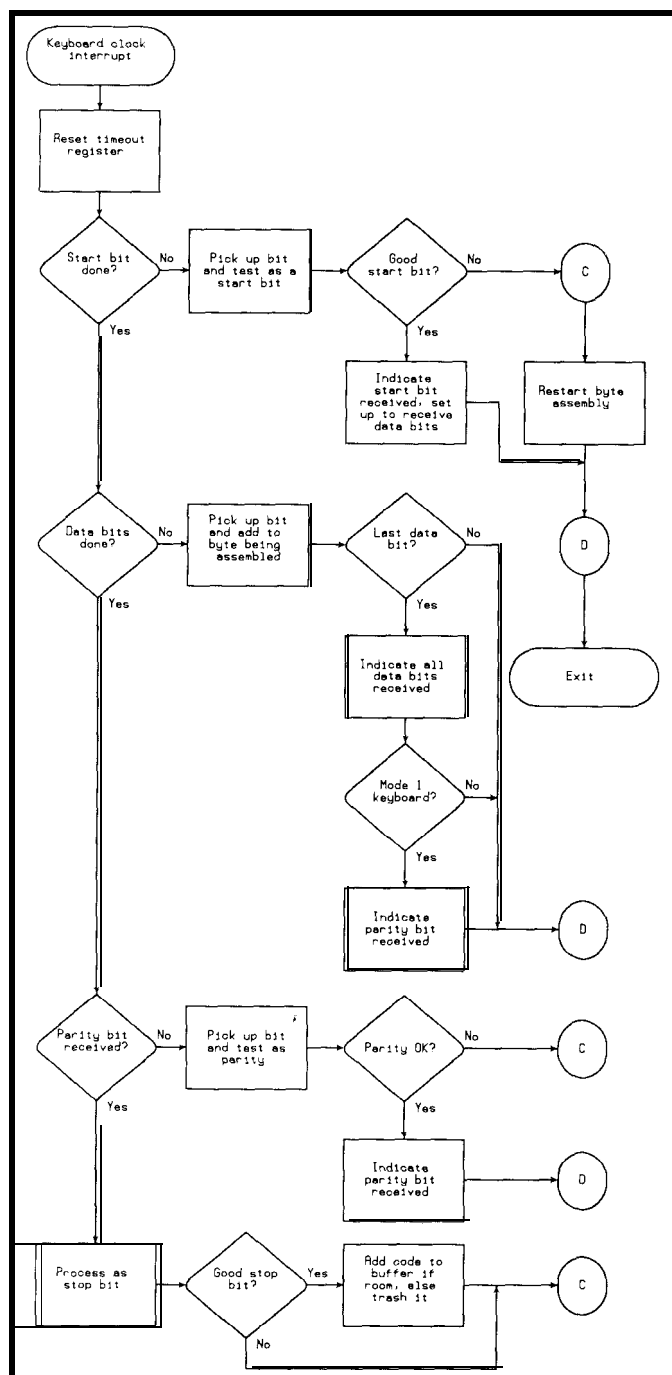


Figure 3—The interrupt service routine for the keyboard clock interrupt is primarily responsible for assembling key codes as they come in from the keyboard.

should make the receiving system very happy. One, both, or neither of the ASCII interfaces may be selected. This is determined by the bit settings in the parameter byte passed to the system at startup time. While selecting neither of the interfaces may not make much sense, it is nevertheless possible.

When the ASCII interfaces are idle, the ASCII code ring buffer is checked to see if more data is available to be transmitted. The ASCII interfaces are

always loaded together if both are selected, and each must wait for the other to be finished transmitting before the next byte will be loaded. This simplifies system design, and will not usually be a performance problem, since one of the two interfaces will normally not be selected.

If either of the selected interfaces is busy, or there are no ASCII codes in the ASCII ring buffer, the key code ring buffer is checked for bytes to be translated. As mentioned above, the

keyboard routines are interrupt driven, and are invisible as far as the main routine is concerned. Data appears in the key code buffer "under the covers," and the main routine only worries about translating key codes once they appear.

## TRANSLATING KEY CODES TO ASCII BYTES

There are several things that may happen when a particular key code is translated. First, the key code might be converted directly to an ASCII-equivalent code and put in the ASCII code ring buffer.

Another thing that might happen is that the key code will cause a status indicator to be set. Alt, Ctrl, either Shift key, Caps Lock, and Num Lock are examples of this kind of activity. The translator routine must remember when these keys are "made," since their settings influence the translation of the codes that follow.

The final translate option is that the code will be ignored, which is the case for most "break-type codes. Exceptions to this would be the break codes for either Shift key, Alt, or Ctrl.

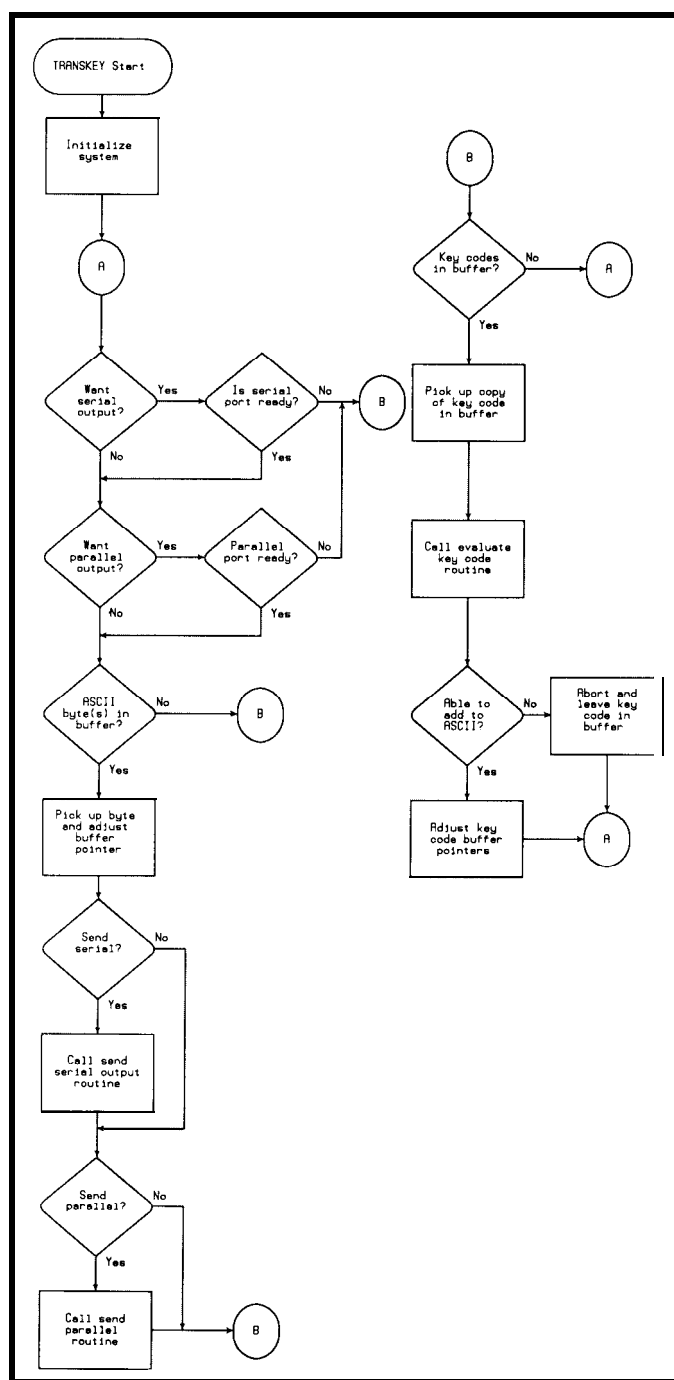
## TRANSLATION TABLES

While the translator tables and software are currently set up to do a one-to-one translation of key codes to ASCII codes, it would be possible to set up a one-to-many system. There are several simple approaches to this kind of translation problem. One would be to use translate tables that had a byte count first, and then the data to send.

A slightly different approach would be to use a special code on the one-to-one table to indicate when multiple bytes need to be sent. The multiple byte codes would then be implemented on a different table, using the tables that have a byte count with ASCII data following as described above.

The modified approach would cut down on the amount of memory needed for translation, since only the multiple codes required would be on the one-to-many table. Most codes

Figure 4—The main driver for the keyboard translator is responsible for initializing the serial, parallel, and keyboard interfaces, and for buffer maintenance and character flow control.



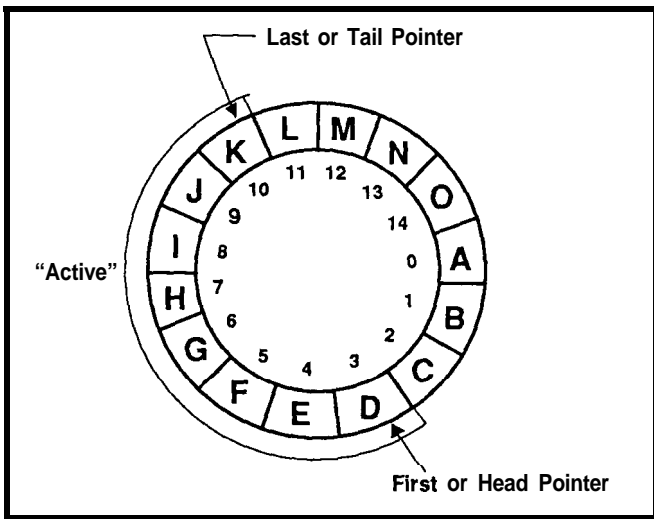


Figure 5-A ring buffer uses a head pointer and a tail pointer to keep track of the active entries in the buffer. Here, entries D-K are active.

would be single bytes, and would fall on the "regular" one-to-one table. This approach also has the advantage of being easily implemented on top of the one-to-one table with little change to the original table or its driver software.

RING BUFFERS

As a final note, I would like to say a few words about the ring buffers, such as that shown in Figure 5, used to hold key codes and ASCII bytes. A "ring" buffer is one where two pointers are used to point into a table of values. The pointers into the buffer are called the first and last, or sometimes the head and tail, pointers. In this system, the buffers are 256 bytes long, and the pointer values can be from 00H to FFH. The pointers are always incremented, and never decremented. Thus, they cycle from 0 to 255 and back around to 0 again, hence the name "ring" buffer.

There are two important things to know about any buffer: When it is empty and when it is full. This is easily done with a ring buffer by comparing the values of the pointers. When FIRST equals LAST, the buffer is empty. When LAST+1 equals FIRST, the buffer is full.

If there is data in the buffer, but the buffer isn't full, then LAST won't equal FIRST, and LAST+1 also won't equal FIRST.

In this system, the interrupt routine is the only routine allowed to modify the LAST pointer for the key

code buffer. The EVAL routine is the only entity which can modify the LAST pointer for the ASCII code buffer, while MAIN may modify the FIRST pointer for either buffer.

THE CAT'S REVENGE

If you read the code listing, you may notice some discrepancies in coding style along the way. That's because this system evolved over the course of a full year, and was done in dribs and drabs. Some days I was partial to the asterisk for comments, other days it was the semicolon. Several PCs were used that all had the same program development software, but, of course, the tabs were off by a few columns between them (sigh).

As a final note, before I had completed my tests on this system and its software, my Franklin keyboard bit the dust (one bit wouldn't go high in the parallel interface). I think the cat danced on it in revenge for my erratic feeding schedule. Oh well, anybody out there know of a good computer tag sale coming up? 🐾

Bill Curlew is a data processing manager for a major insurance firm. When not working with mainframe installations, he enjoys developing microcomputer applications and riding motorcycles.

IRS

- 204 Very Useful
- 205 Moderately Useful
- 206 Not Useful

Menu-driven software to monitor, display, and control your home or production system on site or from a Remote location.

uControl Features

- Display:
    - up to 16 analog inputs
    - up to 32 discrete inputs
    - up to 32 discrete outputs
  - Sample Rate:
    - update all within 1 sec.
  - Alarms
    - Switch Discrete Output on:
      - analog threshold
      - 'trip' of discrete line
  - Password Protection
    - 4 Priorities
  - MS-Windows based display
    - customization program.
- Price: \$175.

Control-Remote ADDS

- Dial up from remote to access all features
- Automatic dial-out on Fault Condition
- Price: \$95. (requires uCONTROL)

uControl-History ADDS

- Historical Plotting of any input or output vs. time.
- Price: \$95. (requires uCONTROL)

Requirements:

uControl. IBM PC/XT/AT (compatible) system with 512k memory; compatible data acquisition card (inquire about boards and systems supported.) Microsoft Windows Release 2.1 or later required to run configuration program. -Remote. also Hayes compatible modems at each end.

Commands: operator Logon immediate Print pick Transmitt Out  
 Logged on with Priority 4  
 Commands: Immediate mode Calibrations set autoPrint Save Data Logoff  
 Transmitter Console Keyboard in Cor

Analog Input	A Digital Output	A Digital Input
Temp Out : 2048.00	Sprinklers : ON	Front Door : CLOS
Temp 2nd Fl : 2048.00	Burglar Alarm : OFF	Window Interior : TRIP
Temp Burnt : 2048.00	Digital Out 2 : ON	Digital In 2 : OFF
Analog In 3 : 2048.00	Digital Out 3 : ON	Digital In 3 : OFF
Analog In 4 : 2048.00	Digital Out 4 : ON	Digital In 4 : OFF
Analog In 5 : 2048.00	Digital Out 5 : ON	Digital In 5 : OFF
Analog In 6 : 2048.00	Digital Out 6 : ON	Digital In 6 : OFF
Analog In 7 : 2048.00	Digital Out 7 : OFF	Digital In 7 : OFF
Analog In 8 : 2048.00	Digital Out 8 : ON	Digital In 8 : OFF
Analog In 9 : 2048.00	Digital Out 9 : ON	Digital In 9 : OFF
Analog Input : 2048.00	Digital Out 10 : ON	Digital In 10 : OFF
Analog Input : 2048.00	Digital Out 11 : ON	Digital In 11 : OFF
Analog Input : 2048.00	Digital Out 12 : ON	Digital In 12 : OFF
Analog Input : 2048.00	Digital Out 13 : ON	Digital In 13 : OFF
Analog Input : 2048.00	Digital Out 14 : ON	Digital In 14 : OFF
Analog Input : 2048.00	Digital Out 15 : ON	Digital In 15 : OFF

10 JUN 1989 8:28

30-day money-back guarantee

(617) 861-0181

FAX (617) 861-1850

**Unkel Software Inc.**

62 Bridge Street, Lexington, MA 02173

Reader Service #158



# FEATURE ARTICLE Part 1

J. Conrad Hubert  
Dick Hubert

## Building étude

A 25-MHz Analog-to-Digital Converter for the PC Bus

The list of applications for flash ADCs is long indeed: Medical instrumentation, RADAR, spectrum and transient analysis, test systems, digital oscilloscopes, and soon. However, one drawback to developing a system which solves one of these "application specific" tasks is the loss of generality inherent in most digital computers.

A short time ago, we were involved in a project to build a low-cost spectrum analyzer for nuclear magnetic resonance research, which traded reduced analog circuit complexity for increased digital signal processing burden. One requirement of that instrument was an analog-to-digital section which would accurately quantize signals containing 10-MHz frequency components. From that work, we learned how to build *étude*, but more importantly, we learned that "application generalized" design allows maximum flexibility in the configuration of both hardware and software.

*étude* is an example of an "application generalized" design. Its specifications include:

- 25-MHz maximum sampling rate
- 10-MHz full-power bandwidth
- \*Guaranteed 40 dB SNR (42.5 dB typical)
- 8-bit resolution (256 quantization levels)
- \*Sensitivity of 3.92 millivolts per bit
- Accepts bipolar or unipolar analog input
- All analog circuitry contained in a single hybrid package

- \*Interleaved 4-KB cache (allows relaxed memory speed requirements)
- Cache-to-PC RAM transfer by DMA or I/O port read
- "Noncached" mode (ADC-to-PC RAM as converted)
- \*Jumper-selected base address (allows up to 16 *études* in one PC)
- Jumper-selected DMA channel (1 or 3)
- Software-selected output coding format
- Turbo Pascal drivers and demonstration software

The hardware design for the converter breaks down into six sections, as shown in Figure 1. We'll take each section in turn, looking at the components and their relationships.

### ADC AND POWER SUPPLY REGULATOR

The heart of the board is TRW's THC1068 hybrid flash converter. The ADC combines all circuitry required to convert analog signals into 8-bit digital data at 25 megasamples per second. It consists of a wide-band analog input amplifier, precision voltage reference, and a TTL-compatible three-state output buffer as well as zero-scale and full-scale flags.

The input impedance of the ADC itself is 1 k $\Omega$ . A lower impedance results from installing R1 in parallel with the 1-k $\Omega$  ADC. The closest standard 1% resistors to terminate 50- and 75- $\Omega$  coaxial cables are 52.3  $\Omega$  and 80.6  $\Omega$ , respectively.

Potentiometer R3 allows adjustment of the input offset voltage over a

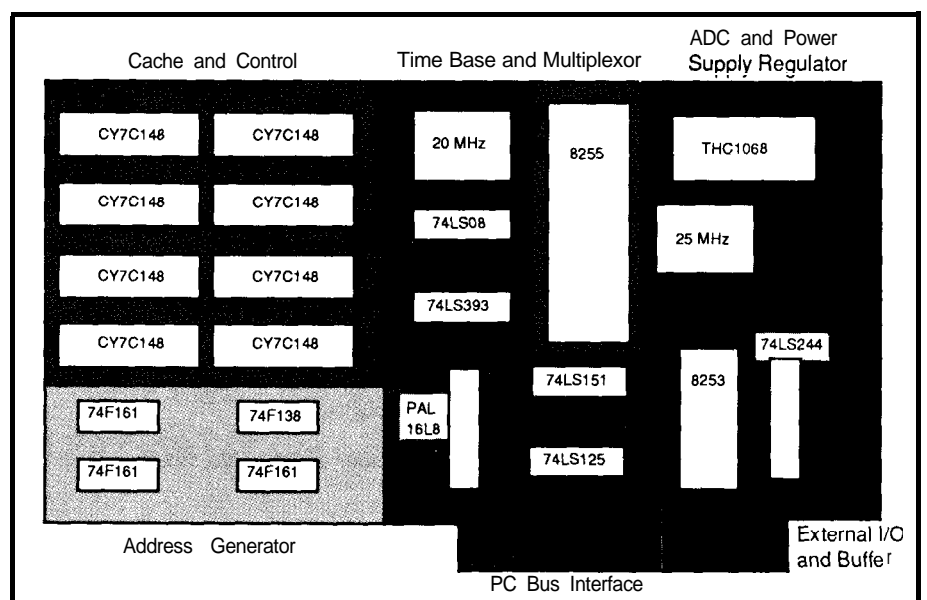


Figure 1 — The six major sections of *étude* manage to fit a full 25-MHz digitizer into an IBM PC-bus half-length card.

±0.5-volt range. This potentiometer is typically set to provide a 1-volt window centered at 0 volts.

When R3 is set at midpoint, the ADC will accept bipolar input signals. As R3 is moved to one extreme, the input range becomes unipolar positive, while the opposite extreme allows for unipolar negative inputs.

We **recommend** adjustment of the optional gain potentiometer (R5) not result in more than a 10 percent gain change. Also, it is more practical to use R5 solely for gain increases since this reduces power dissipation in the ADC's resistor ladder. Gain decreases are better achieved with a resistive divider at the analog input.

The ADC's specifications call for it to withstand voltages between +5 and -5 volts. Overload protection for the ADC is accomplished by zener diodes D1 and D2 which act as clamps to protect the ADC's internal amplifier. If the voltage into the BNC is greater than the zener voltage (±3.3 volts), they short-circuit the source driving étude.

U31 is a  $\mu$ A7952 voltage regulator which produces -5.2 VDC from the PC's -12-volt line. The ECL circuitry inside the ADC draws about 250 mA at 25°C. Since a standard PC power supply is rated to deliver only 300 mA at -12 V, external power may be supplied via the DB-15 if multiple études are installed in the same computer. To do so, remove jumper W2 and connect power to pin 9 of the DB-15. The positive terminal of the supply must be connected to the BNC common. This is not necessary in all cases, since we have drawn over 2000 mA from a 220-watt AT supply without problems.

## FULL-POWER BANDWIDTH

*All Flash Converters are Not Created Equal*

We'll examine how full-power bandwidth (FPBW) applies to étude, but first a bit of background. The most common type of ADC is one which uses a successive approximation register (SAR). SAR converters work by first generating a reference voltage, then using a technique similar to a binary search algorithm to make increasingly better approximations to the actual value of the analog signal. They use a comparator to answer the question, "Is the input greater than or less than the reference?"

At the start of the conversion process, the most-significant bit (MSB) in the SAR is set (turned on). If the reference voltage is too big, the MSB is cleared (turned off). The process is repeated with the next-most-significant bit, and so on down to the least-significant bit (LSB). This procedure is valid only if the input remains constant over the entire conversion. A sample-and-hold amplifier is necessary for signals which vary more than 1 LSB during the conversion. An n-bit converter requires, at most,  $n$  comparisons.

The speed of SAR-based converters did not meet our requirements, so we selected an ADC called a flash converter. Flash converters use  $2^{n-1}$  comparators (where  $n$  is the number of bits of resolution). They also require a  $2^n$ -resistor ladder network to divide a precision voltage reference into equally spaced voltages. This gives the comparators something to compare against. A priority encoder makes an n-bit binary number out of the  $2^n$  comparator outputs. This is all accomplished in tens of nanoseconds, hence the name *flash* converter. We chose a flash converter for étude which, when operating at its maximum sampling rate, guaranteed its binary output will accurately reflect the analog input to within a known SNR specification rather than a converter that will **simply** not yield gross inaccuracies at more than some specific rate.

**Exactly** what are gross inaccuracies? One of the first applications of flash ADCs was to digitize video images in real time.

You may even hear flash ADCs referred to as video ADCs. In NTSC video, the digitizing rate is just over 14 MHz, so everyone's flash converter had to operate at least that fast. One manufacturer addressed the question of whether the digitized information is always accurate by introducing a specification which indicated how few bright-white dots appeared per reconstructed frame of video information. These bright-white dots or "sparkles" are due to spurious codes.

A spurious code is a grossly inaccurate datum such as a **midscale** input signal resulting in a full-scale output code, whereas a missing code is defined as an output that has a value less than the lowest possible quantization level. The bandwidth specification for an analog device (in which attenuation is the primary concern) and the bandwidth specification for a flash ADC are defined differently. In the latter, FPBW is derived from the method of testing for spurious codes, which involves applying a full-scale sine wave of known frequency to the ADC's input, and examining the output for spurious or missing codes. FPBW is then defined as the frequency just below the point at which spurious or missing codes begin to exist.

An FPBW specification of less than one-half the maximum sampling rate indicates that performance degrades before reaching the theoretical Nyquist limit. This problem is manifest in wide-band systems when components of the input signal approach one-half of the sampling frequency, while low-frequency multiplexed systems exhibit degraded performance if the multiplexer must switch between adjacent channels which differ significantly in potential. Both of these problems are related to the input signal's slew rate (the maximum rate of change of amplitude with time).

The source of these problems becomes apparent when one realizes that flash converters' comparators determine all bits in an output code in parallel by decoding an intermediate result known as a thermometer code. A thermometer code is a condition in which a logical one is generated by all comparators below a specific voltage, and a logical zero is generated by those comparators above that voltage. If the input signal slew rate is great enough, there will be more than one transition from ones to zeros in the comparator's outputs, thus causing spurious or missing codes.

Perhaps a more useful definition of power bandwidth would establish the full-scale input frequency which produced differential-linearity errors greater than 1LSB. Using this definition of power bandwidth, we selected the TRW TDC1068 for étude. It is capable of full-power Nyquist-frequency operation at 20 million samples per second.

PB7 S0	PB6 S1	PB5 S2	
0	0	0	One-shot from 8253 (also off)
0	0	1	5 MHz
0	1	0	External timebase input
0	1	1	20 MHz
1	0	0	Variable frequency from U23, counter 1
1	0	1	10 MHz
1	1	0	DMA
1	1	1	25 MHz

Table 1 - The three ports of U21, PB5-PB7, select one of eight *input sources* for the 74LS 151 multiplexor.

## EXTERNAL I/O AND BUFFER

The DB15 connector facilitates external I/O, while a 74LS244 Schmitt-trigger buffer cleans up ugly signals. All signals are TTL compatible and run adjacent to a digital ground. The following are pin assignments and explanations:

**Pin 1—+5VDC Output:** This is provided to power external circuitry. Maximum current draw is not specified.

**Pin 2—Done\ Output:** This output reflects the state of the sample counter. It goes high when the board begins acquiring data. When the preset number of data points have been acquired, the line goes low.

**Pin 3—Full-Scale\ Flag Output:** This output is not latched and indicates the status of the sampled analog data one clock cycle before it appears as output from the ADC. The Full-Scale\ flag indicates that the output bits of the converter are all high. You may infer a full-scale input only when true binary output coding has been selected.

**Pin 4—Zero-Scale Flag Output:** This output is not latched and indicates the status of the sampled analog data one clock cycle before it appears as output from the ADC. The Zero-Scale flag indicates that the output bits of the converter are all low. Again, you may infer a zero-scale input only when true binary output coding is selected.

**Pin 5—TimeBase Output:** This output is the clock signal from which conversion and storage sequencing is derived. All of etude's actions are

## Parts list for *étude*

Quantity	Type	Value	Ref. Des.
1	1 -of-8 decoder	74F 138	U1
1	-5.2-V regulator	$\mu$ A7952C	u31
2	0.25-W resistor	10k	R2,R4
1	0.25-W resistor	52.30R80.6 1%	R1 (for Zin=50 or 75)
8	1 Kx4 SRAM	CY7C148	U9,U10,U11,U12, U13,U14,U15,U16
3	4-bit counter	74F161	U2,U3,U4
1	8-to-1mux	74LS151	U24
1	25-MM 8-bit ADC	TRW THC 1068	u22
1	Address decoder	PAL16L8A	U26
2	Berg strips	2x3	J3,J4
4	Berg jumper		(none)
1	Card bracket		(none)
1	Circuit board	ADC.4.25.PC	PCB
1	Counter/timer	8253-5	U23
1	DB-15 female		J2
1	Dual 4-bit counter	74LS393	U25
1	Ferrite on lead	F/R2743009 111	L1
1	insulator mica		(for U3 1)
2	Jack screw assy		(for J2)
15	Monolithic	0.047 $\mu$ F	C5-C19
1	Multiturn pot	2k	R3
1	Octal buffer	74LS244	U27
1	PC-mount BNC		J1
1	PPI	8255	u21
1	Quad 2-input AND	74LS08	U30
1	Quad buffer	74LS125	U29
1	Screw & nut	#4-40x 1/4	(for U31)
1	Socket Augat	24-pin	(for U22)
4	Tantalum	10 $\mu$ F 16V	C1,C2,C3,C4
1	Xtal oscillator	20 MHz	osc 1
1	Xtal oscillator	25 MHz	osc2
2	Zener	1 N4728A	D1,D2

referenced to the rising edge of this signal. It reflects the software-selected acquisition rate only when the cache is filling; otherwise it is low.

**Pin 6—External TimeBase Input:** This input is selected via software and becomes the master clock signal for *étude*. It may be used to acquire data at a rate unavailable internally or to utilize a higher accuracy/stability clock than the on-board oscillators.

**Pin 7—Timebase\ Output:** This is the same signal as Timebase Output, only 180 degrees out of phase. It provides the capability of running two boards simultaneously on the same input data. In this way, data can be acquired by the second board in between the points collected by the first board. This doubles the effective sampling rate when the data is combined.

listing 1 — The PAL equations are used for decoding PC-bus I/O space signals during transfer to and from the host computer.

```
TITLE      étude ADDRESS DECODER
PATTERN    ETUDE.PDS
REVISION   (PRODUCTION VERSION)
AUTHOR     J. CONRAD HUBERT
DATE       APRIL 7, 1989
```

```
CHIP ADDRESS-DECODE PAL16L8
```

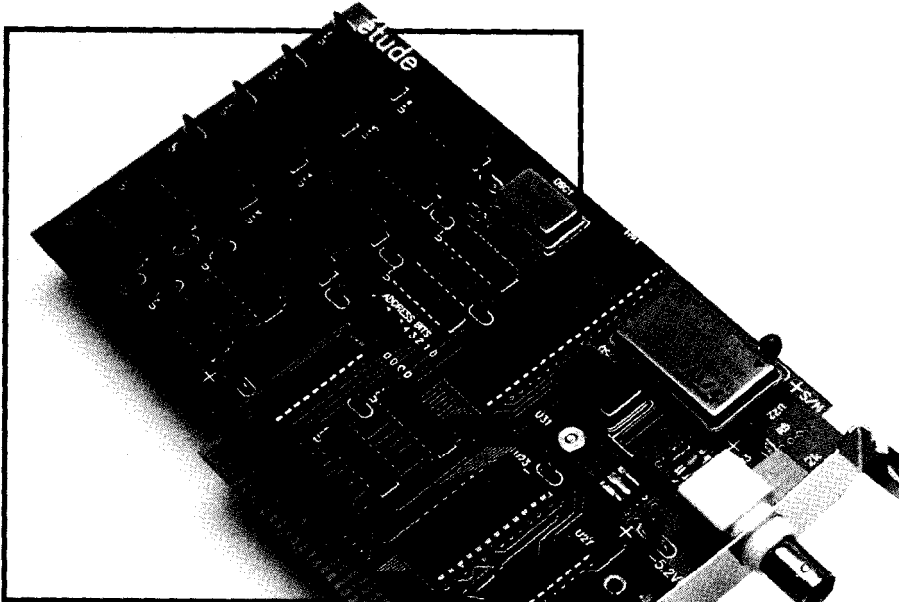
```

;PINS      1      2      3      4      5      6      7      8      9      10
           A9     A8     A7     A6     A5     B4     B5     B6     B7     GND

;PINS      11     12     13     14     15     16     17     18     19     20
           AEN    E8253 A4     A2     IOW    IOR    COMP  E8255 DMA   vcc
```

```
EQUATIONS
```

```
COMP      = /(A7 :t: B7) * /(A6 :+: B6) * (/IOR t /IOW)
/E8253    = A9 * /A8 * COMP * /(A5 :+: B5) * /(A4 :t: B4) * /A2 * /AEN
/E8255    = A9 * /A8 * COMP * /(A5 :+: B5) * /(A4 :+: B4) * A2 * /AEN
/DMA      = (/IOR + /IOW) t AEN
```



**Pin 8 —Inhibit\**  
**Release:** This input is tied high through a 10k-ohm resistor and also serves as a hardware trigger. If this line is pulled low, the ADC continues to run, but the address generator and sample counter are inhibited and data storage is halted.

Such control allows synchronizing the data acquisition with external events, as well as accommodating arbitrary pauses. Efficient use of the cache is important since 25-MHz

operation will fill a 4-KB cache in 163  $\mu$ s.

**Pin 9— -5.2 VDC Input/Output:** The direction of this supply is jumper selected via W2. It is designed to provide a -5.2-VDC output at 50 mA. By cutting jumper W2, etude's negative supply requirement may be provided by an external -5.2-VDC source.

**Pins 10-15—Digital Ground.**

### TIMEBASE AND MULTIPLEXOR

OSC1 and OSC2 form etude's internal timebase. OSC1 produces a 20-MHz square wave, while OSC2 produces a 25-MHz square wave. Another oscillator may be substituted for OSC2 to obtain a single sample frequency not available elsewhere.

U25 is a 74LS393, dual 4-bit ripple counter. One half of it is unused. The other half is clocked from OSC1, so its outputs are 10-MHz, 5-MHz, and 2.5-MHz square waves.

U23, an Intel 8253, contains three independent 16-bit counter/timers. Counter 2 is clocked at 2.5 MHz and

## Introducing the MICROMINT T-286 CPU

Micromint announces the latest addition to its line of high-quality CPUs for industrial and OEM applications: The Micromint T-266. This IBM PC/AT-compatible computer packs the features our customers requested most into the single expansion card format made popular by the OEM-286. Add the quality and support that have made Micromint famous, and you have a CPU that can't be matched for total price, features, and performance!

The T-266 is 100% AT-compatible, with clock speeds switchable (via jumpers or keyboard) between 6 and 12 MHz. Among the features offered on the T-286 are an on-board real-time clock, socket for an 80287 numeric coprocessor, on-board keyboard connector, and 4 megabytes of on-board RAM capacity. The T-266 features the industry-standard Award BIOS, and offers OEM customers the advantage of a 32K ROM, expandable to 64K for custom applications. ISA Bus compatibility is assured, with the T-266 taking up a single slot in an ISA Bus passive backplane, and requiring only +5V power for operation.

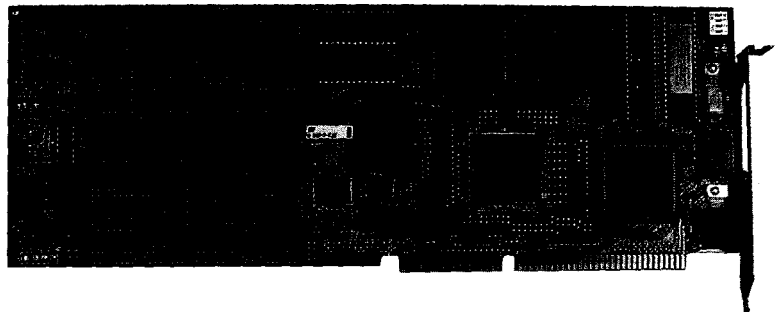
Low-power operation; speed; maximum configurability—the T-286 is the perfect choice for critical applications where ISA Bus expansion or MS-DOS software development are specified. Call Micromint today for more information on the T-266 CPU and custom system availability.

**T-286/0— 12 MHz 80286 CPU w/0K RAM** \$399.00  
**T-286/4 — 12 MHz 80286 CPU w/ 4MB 80-ns, 0-wait-state RAM** Call for Prices



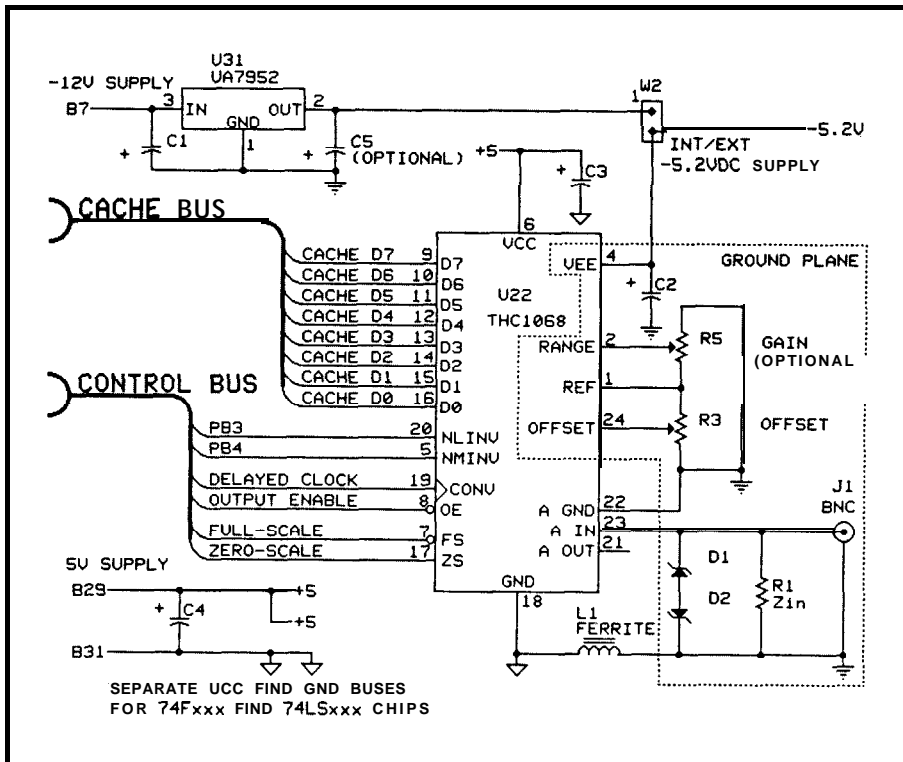
**MICROMINT, INC.**

4 Park Street, Vernon, CT 06066  
 Tel: (203) 871-6170 . Fax: (203) 872-2204



### T-286 CPU Technical Specifications

- 100% IBM PC/AT compatible
- 80286-12 processor, operating at 6/12 MHz, 0 or 1 wait state
- Speed selectable with keyboard or jumpers
- 256K, 1 meg. or 4 megs of on-board RAM
  - can use 4164, 41256, or 411000 DRAMs
- Battery-backed real-time clock (DS1287) and configuration RAM
  - 10-year on-board battery
- Socket for 80287 Numeric Coprocessor
- 32K ROM, expandable to 64K
- Award BIOS
  - Configuration and setup routines in ROM
- On-board keyboard connector (5-pin DIN)
- On-board tone beeper
- Needs only +5V for operation
- Single-slot, full-length expansion board form-factor
- ISA Bus interface



**Figure 2**—The IHC 7068 hybrid flash converter is the heart of the board. It contains the circuitry needed to convert analog signals into d-bit digital data at 25 megasamples per second.

provides the capability to increment the address generator via a one-shot. Counter 1 is clocked from the 2.5-MHz output of U25. It provides frequencies which can be obtained by dividing 2.5 MHz by a 16-bit binary integer. (Dividing by two yields 1.25 MHz, while dividing by 64K yields approximately 38 Hz. Division by one is not possible.) Counter 0 is clocked from U3 which is the timebase output divided by 16. The input to counter 0 is divided by 16 for two reasons:

1) the 1-MB version of étude requires a 20-bit address generator, and the 8253 counter/ timer has only a 64-KB range, therefore we must scale the count by a factor of 16. (1 MB divided by 64 KB = 16.)

2) 25 MHz is too fast for an 8253 to count. Since counter 0 receives every sixteenth clock pulse, this slows the maximum count rate to 25 MHz divided by 16, or 1.5625 MHz.

Originally, the sample counter was intended to halt data acquisition after a preset number of points had been acquired. Unfortunately, at the highest sample rates, the 8253's delay between reaching terminal count and its output pin changing states was

long enough to allow the cache to overflow. Now the terminal count signal comes directly from the address generator when the cache is full. Of course, at slower acquisition rates it is still possible to poll the 8253 in real time to determine how much of the cache is filled, and then halt data acquisition via software. If partial filling of the cache is required at the highest rates, there are two alternatives:

1) Invert the Done\ flag and connect it to the Inhibit\Release line. This will allow some overrun from the 8253's preset value. However, the address generator will never "wrap around" and overwrite the first data points because the terminal count signal from the address generator still disables the multiplexer.

2) If no overrun can be tolerated when partially filling the cache at the highest sampling rates, a fast external counter could be employed to generate an Inhibit\ signal when the preset number of clock pulses have emanated from Timebase Out.

U24 is a 74LS151, 1-of-8 multiplexer. One of eight input sources is selected by a 3-bit input on S0, S1, and S2, which is software-generated by U21 ports PB7, PB6, PB5, according to Table 1.

### ADDRESS GENERATOR

The address generator consists of three 74F161, 4-bit synchronous binary counters U2, U3, and U4. These chips form a fast 12-bit counter with look-ahead carry. The circuit is built in conjunction with U1, a 74F138, 1-of-8 decoder, which generates cache chip selects.

### PC BUS INTERFACE

We use the PC's I/O space as one way to communicate with étude. It is

Port	Name	Function
PB0	Reserved	
PB1	Reserved	
PB2	DMA\	DMA enable
PB3	NLINV\	ADC output coding format
PB4	MNINV\	ADC output coding format
PB5	S2	MUX control bit 2
PB6	S1	MUX control bit 1
PB7	S0	MUX control bit 0
PC0	ENIO\	Enable 74LS244 I/O buffer
PC1	Reserved	
PC2	CLR	Zero address generator
PC3	Reserved	
PC4	INHIBIT\	Halt address generator
PC5	WR\	Cache write
PC6	RD\	Cache read (not used in 4KB version)
PC7	OE\	ADC output enable

**Table 2**— Control for the chips in the cache section of the design are latched through ports B and C of the 8255, using the individual bits shown here.

# INTROL CROSS DEVELOPMENT SYSTEMS

- INTRC-C Cross-Compilers
  - INTRC-Modula-2 Cross-Compilers
  - INTRC-Macro Cross-Assemblers
- Provide cost and time efficiency in development and debugging of embedded microprocessor systems

All compiler systems include:  
Compiler • Cross-assembler • Support utilities • Runtime library, including multi-tasking executive • Linker • One year maintenance • User's manual, etc.

TARGETS SUPPORTED:  
6301/03 • 6801/03 • 6804 • 6805 • 6809  
• 68HC11 • 68000/08/10/12 • 32000/  
32/81/82 • 68020/030/881/851

AVAILABLE FOR FOLLOWING HOSTS  
VAX & MicroVAX; Apollo; SUN; Hewlett-Packard; Gould PowerNode; Macintosh; IBM-PC, XT, AT and compatibles

INTRC CROSS-DEVELOPMENT SYSTEMS are proven, accepted, and will save you time, money, effort with your development. All INTRC products are backed by full technical support. CALL or WRITE for facts NOW:



**INTROL  
CORPORATION**

647 W. Virginia St., Milwaukee, WI 53204  
414/276-2937 FAX: 414/276-7026  
Quality Software Since 1979

Reader Service #127

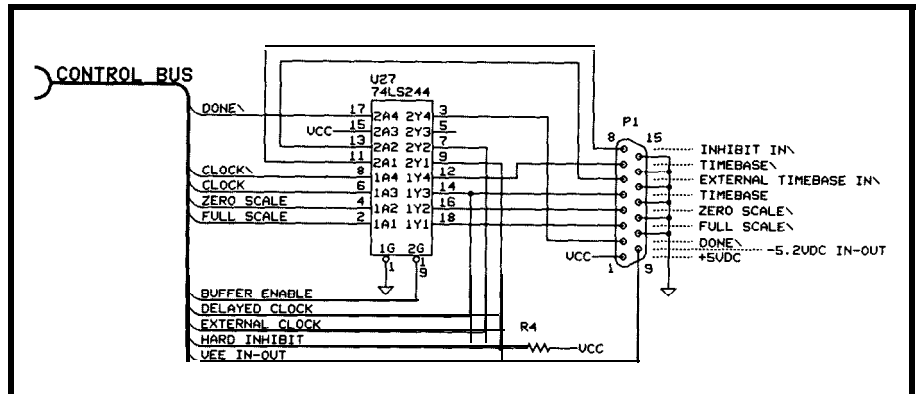


Figure 3-A DB-15 connector is the *étude's* link to the outside world. The 74LS244 buffers the incoming and outgoing signals for the entire board.

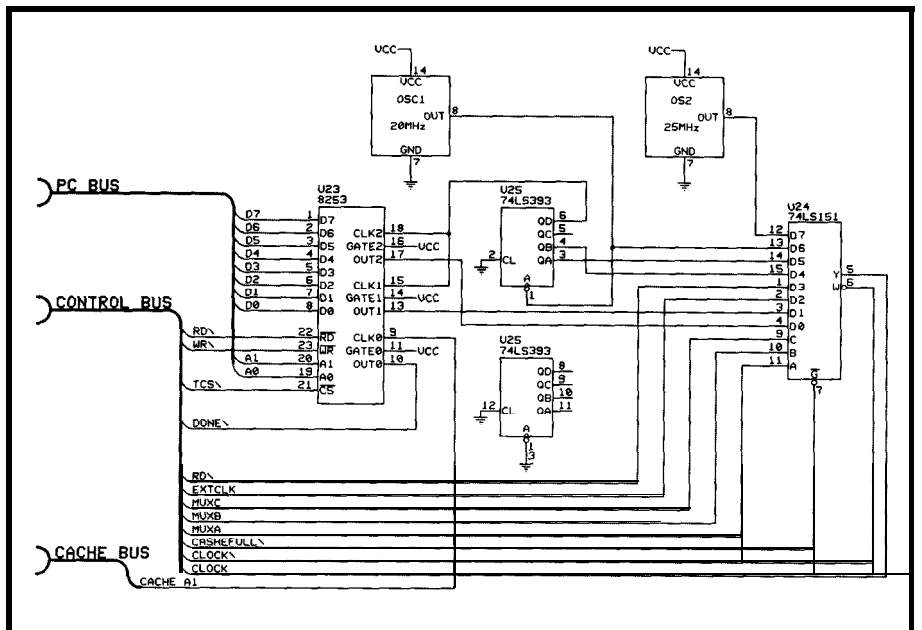


Figure 4—The basic signal for the digitizer's internal timebase is generated by a 20-MHz oscillator. The 8253 provides three independent 16-bit counter/timers for the address generator. U24 is a 1-of-8 multiplexer with eight inputs selected by software which corresponds to the ports shown in Table 1.

the only way to program specific parameters, like trigger source, acquisition rate, output coding format, and so on, and provides a way to read data from the cache. *étude* also allows a faster, albeit more complicated, way to read large blocks of the cache into main memory via either DMA channel 1 or 3. We will discuss DMA further in the next issue of *CIRCUIT CELLAR INK*.

U30, a 74LS08, is a quad AND gate. Three-fourths of it are used to generate conditioned AO, AI, and chip select signals for U21 when DMA is requested. One-fourth is used as an active-low OR gate for the address generator hold circuit.

U29 a 74LS125 is a quad buffer with enables. One-half of it buffers the PC's IOR\ and IOW\ signals. The other half is enabled during DMA request and acknowledge cycles.

U26 is a PAL16L8. Chip selects for U21 and U23 are derived by decoding the PC's I/O space signals A9 through A3, AO, AI, AEN, IOR\, and IOW\, according to the equations shown in Listing 1.

## CACHE AND CONTROL

The 8-bit, 4-way interleaved cache consists of eight 1K x 4 CMOS CY7C148L-45 Cypress chips. We used these chips because of their availabil-

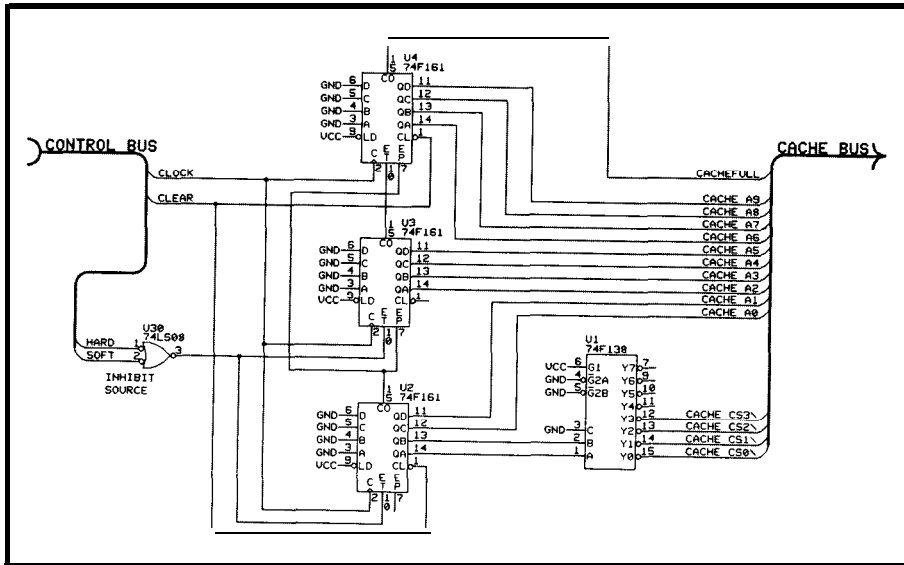


Figure 5—The Address Generator is three 74F161 4-bit synchronous binary counters in conjunction with a 74F138 1-of-8 decoder.

ity, since they are much faster than they need to be for this application.

Software-based control tasks are assigned to U21, an Intel 8255 programmable peripheral interface operating in mode 0. It latches control signals and provides a common port for both DMA and I/O port access to the cache. This yields flexibility of operation without any sacrifice in performance. Port A provides bidirectional I/O for cache-to-PC data transfers during both DMA and IOR\ cycles. Ports B and C latch control

signals for the other chips. Individual bits in ports B and C are shown in Table 2.

### CONSTRUCTION TIPS

When integrating a high-performance ADC into an existing system, some compromises are inevitable. We would have liked to keep the analog and digital grounds from the power supplies separate and used a single-point ground mecca directly under the ADC. Unfortunately, that

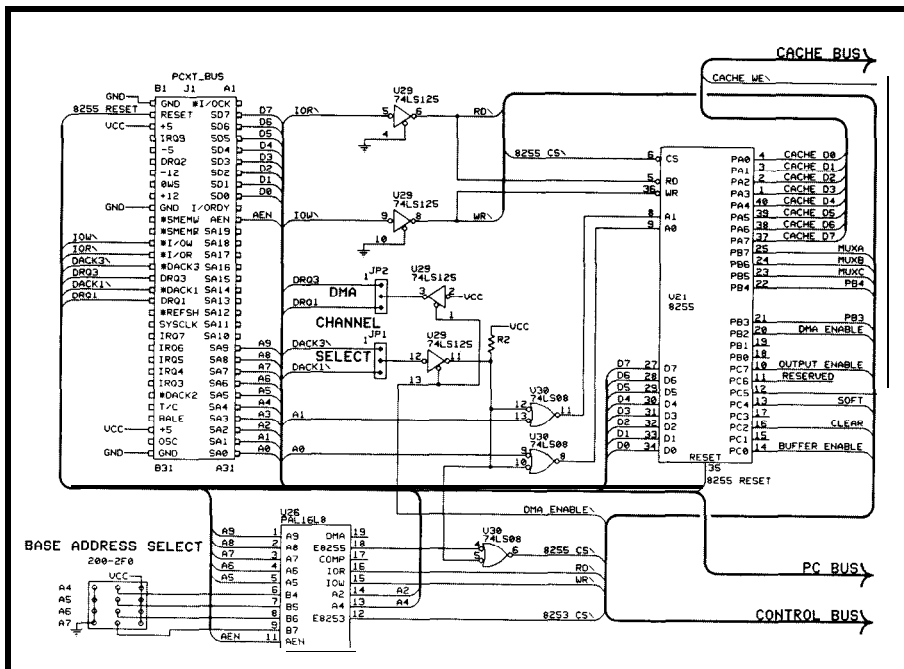


Figure 6—The PC's I/O space allows for programming of specific parameters such as trigger source, acquisition rate, and output coding format. DMA transfer is faster, but at a cost of increased complexity.

Develop powerful image applications quickly and easily

### Introducing VICTOR, the video capture and image processing library

VICTOR, the video capture and image processing library, is the latest product from the developers of ZIP Image Processing software. Victor is a library of functions for C programmers that simplifies development of scientific imaging, quality control, security, and image database software. Victor gives you complete control of your video frame grabber and also includes image processing, display, and TIFF/PCX file handling routines.

Application development is simplified because we've taken care of the details of device control and image processing. All the hard low level coding has been done -- and you can concentrate on your application.

Your software can have features such as live video on VGA, pan and zoom, display multiple images with text and graphics, pixel editing and image processing. And, to get you up and running quickly, we've included our popular Zip Image Processing software for rapid testing and prototyping of image processing and display functions.

Victor supports Microsoft C and Quick C, and includes over 100 functions, demonstration and prototyping software, full documentation, and source code for device control routines... all for only \$195.

Victor and Zip support ImageWise and other popular video digitizers.

VICTOR LIBRARY \$195 includes ZIP image Processing

ZIP Image Processing \$79 please specify digitizer

Video frame grabbers are also available.

Call (314) 962-7833 to order VISA/MC/COD

CATENARY SYSTEMS  
470 BELLEVIEW  
ST LOUIS MO 63119  
(314) 962-7633

was not possible since the PC has a single common ground for all its supply voltages. One way to circumvent this problem is to use an expensive DC-to-DC converter to produce -5.2 V from the +5-V line. A less elegant method calls for resistive isolation between the grounds, but this won't help much since TRW specifies that the analog and digital grounds must be within  $\pm 0.1$  V of one another. We used L1 as an effective low-frequency short, yet it attenuates HF switching noise as its impedance increases with frequency. It is much cheaper than a DC-to-DC converter and serves to keep digital ground noise out of the ADC's analog circuitry. If your application can tolerate a somewhat decreased SNR, you may find that a shorting wire works instead of L1.

Perhaps you've noticed the use of aluminum capacitors to replace tantalum. This is because the price of tantalum has increased dramatically over the last few years. Tantalum is still the choice for high-performance applications—use it! C5 is not required in the

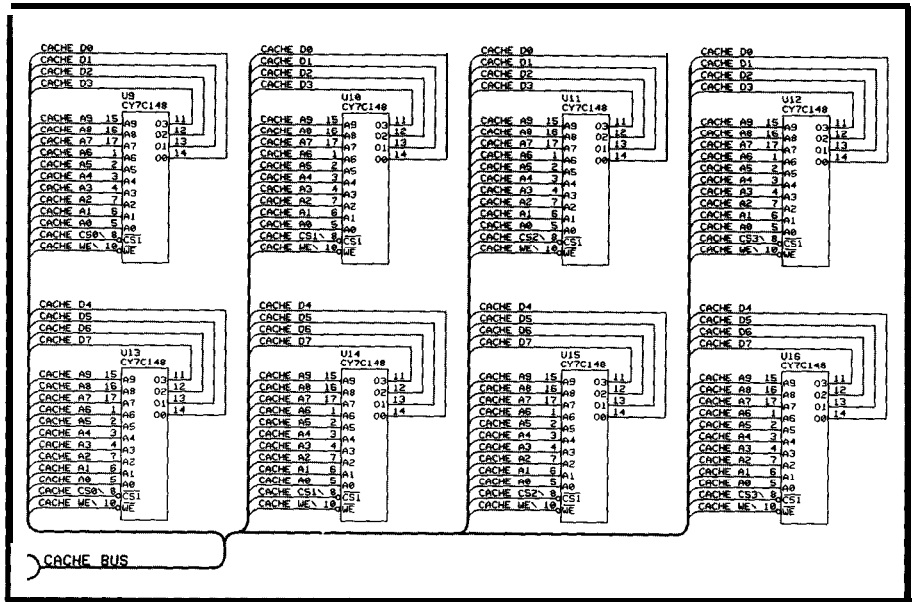


Figure 7—Eight MCM148-45 chips provide the b-bit, 4-way interleaved cache. They are under software control of the 8255 shared with the Timebase and Multiplexor section.

PCB version of the design because C2 was electrically close enough to U31. (C2 is intended to bypass U22.)

Just as power supply pins are implied in digital IC schematics, so are bypass capacitors. F-series logic

and memory chips each get their own bypass cap. Three other bypass capacitors are distributed amongst the other ICs. The ADC has its own internal bypasses because they are critical to the operation of that device and

**NEW WICEZ8**  
In-Circuit Emulator



wytec

This **NEW WICEZ8**, containing the new **Z86C12 CMOS ICE** chip, is a complete In-Circuit Emulator for the **entire Z8 microcontroller family**, including the **86C91** and the **86E21** with 256 internal registers.

- Comes with window-oriented, user-friendly PC driver software for the IBM PC, XT, AT or compatibles and permits real-time emulation up to 20 MHz.
  - Unique display windows monitor 30 programmable memory locations or registers, 17 stack locations and are automatically updated when the user program is stopped or single-stepped.
  - Addresses up to 128K bytes of memory and external data.
  - Provides 32K emulation RAM mappable in 2K, 4K, 8K, 16K, or 32K blocks.
  - Provides 8 hardware breakpoints which can always be displayed on the window.
  - Symbolic debugger reads symbol files in the 2500 A.D., Microtek, Zax and Micro Computer Control formats directly.
  - On-line assembler, disassembler, memory/register exam, compare, fill, move, search and modification.
- 038.4K baud rate for fast upload/download of files in Tektronix Hex, Intel Hex and Motorola S record formats. (Download an 8K hex file in 5 seconds.)

Regular Price: \$1415 Introductory Price: \$996  
**30 day Money Back Guarantee**

**wytec** (708) 894-1440  
**WYTEC COMPANY**  
Suite 140, 185C E. Lake Street, Bloomington, IL 60108

# Total control with LMI FORTH™

For Programming Professionals:  
*an expanding family of compatible, high-performance compilers for microcomputers*

- For Development:**  
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386
- 16-bit and 32-bit implementations
  - Full screen editor and assembler
  - Uses standard operating system files
  - 500 page manual written in plain English
  - Support for graphics, floating point, native code generation

- For Applications: Forth-83 Metacompiler**
- Unique table-driven multi-pass Forth compiler
  - Compiles compact ROMable or disk-based applications
  - Excellent error handling
  - Produces headerless code, compiles from intermediate states, and performs conditional compilation
  - Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11, 34010, V25, RTX-2000
  - No license fee or royalty for compiled applications

**LMI** Laboratory Microsystems Incorporated  
Post Office Box 10430, Marina del Rey, CA 90295  
Phone Credit Card Orders to: (213) 306-7412  
FAX: (213) 301-0761



external ones simply will not work. Also, according to good PCB layout practice, the VCC and GND traces of the F-series chips are separate from those of the LS-series chips. (F-series chips tend to produce more switching noise than LS-series chips on their supply leads.)

In general, it is poor practice to socket high-speed analog circuitry. The dielectric constant of the plastic socket is inevitably greater than that of air and the resulting interelectrode capacitance can often create problems, as does moving the IC away from the ground plane. We have used machine-tooled sockets for the ADC with no degradation in performance. This is due to the ADC's low input impedance and high (3.92 mV per LSB) signal levels.

This concludes the hardware outline. In the next issue we will examine the driver software.+

J. Conrad Hubert owns *Deus Ex Machina Engineering*, a St. Paul, Minn. consulting firm and is a partner in *Silicon Alley Inc.*, a Seattle-based manufacturer of DSP products. In his spare time he likes to sleep.

Dick Hubert is one of the founders of A.P.P.L.E. Coop, is a partner in *Silicon Alley*, and has been involved with microcomputers for ten-to-the-sixth years.

### *étude* [ay-tüüd] nf. study, research

*étude* is available from:

**Silicon Alley, Inc.**  
P.O. Box 59593  
Renton, WA 98058  
(206) 255-7410

both **in kit** and **finished** forms. The kit contains: PC board, PAL, 4 KB of SRAM, manual, and software for \$99.00 (**introductory price**). An assembled and tested version sells for \$495.00. PALs are available for \$5.00.

Extended cache versions (128, 256, 512 KB, and 1 MB) are available from

Rapid Systems, Inc.  
433 North 34th  
Seattle, WA 98103  
(206) 547-8311

### IRS

207 Very Useful  
208 Moderately Useful  
209 Not Useful

**Complete your reference library—  
You cannot afford to miss an issue of Circuit Cellar INK**

## AVAILABLE BACK ISSUES

January/February 1988—Issue #1	SOLD OUT
March/April 1988—Issue #2	SOLD OUT
May/June 1988—Issue #3	SOLD OUT
July/August 1988—Issue #4	SOLD OUT
September/October—Issue #5	
November/December—Issue #6	

**The above issues are available as a Bound Offset Reprint of the First Year of Circuit Cellar INK. Every printing of the Circuit Cellar INK Bound Reprint Book has been a sellout. Don't miss out on the current printing of this popular technical resource! To order your copy of the First Year of Circuit Cellar INK, send \$20.00 (\$24.00 foreign - U.S. funds only) to:**

Circuit Cellar INK First Year Reprint  
4 Park Street • Vernon, CT 06066  
or Call (203) 875-2199 for immediate delivery!

January/February 1989—Issue #7	
April/May 1989—Issue #8	
June/July 1989—Issue #9	SOLD OUT
August/September 1989—Issue #10	
October/November 1989—Issue #11	
December '89/January 1990—Issue #12	

**To order Circuit Cellar INK back issues, send \$4.00 check (Visa, Mastercard accepted) for each issue to:**

Circuit Cellar INK — Back Issues Sales  
4 Park Street • Vernon, CT 06066 or call (203) 875-2199

## The DA/M<sup>TM</sup>

### The Lowest Cost Data Acquisition System

Our DA/M can solve more of your data acquisition problems at a lower price than any other product on the market. DA/M's are used for:

- Military meteorological stations.
- Building management.
- Automated hydroponic farming.
- Industrial process control.
- Your application

In fact, DA/M's can be used whenever you can't afford to use any one else's product.

Made in North America, DA/Ms are available NOW!



DATA ACQUISITION AND MANAGEMENT CORPORATION LTD.

#140, 17303 - 102 Avenue

Edmonton, Alberta, Canada T5S 1J8

Phone 1-403-486-3534

Fax 1-403-486-3535

# Digital Signal Processing

-An Introduction

The term "Digital Signal Processing" for most individuals conjures up thoughts of exotic, complex mathematics and advanced electrical engineering theory. Digital signal processing, as a specialized field, is relatively new, developed and made popular over the last quarter century with the advent of solid-state electronics and the integrated circuit. During this period, DSP required expensive computing equipment in the form of mainframes, minicomputers, or specialized high-speed DSP hardware. Until just recently, the specialized hardware required placed DSP out of the reach of the serious experimenter. However, with the appearance and subsequent deflation of

"microprocessor-like" DSP chips, first unveiled in 1983, moderately priced DSP performance is now available. Many engineers, technicians, and computer programmers with programming and/or circuit design backgrounds would like to learn about and experiment with DSP but don't have access to DSP hardware. Since most do own or have access to PCs, graphics-based BASIC language programs will be provided in the first part of this article which will "visually" demonstrate various DSP fundamentals. For more advanced experiments requiring high-speed real-time performance, "DXP-25" will be offered. DXP-25 is a low-cost IBM-compatible PC card and support software package based on

the popular TMS320C25 DSP chip (second-generation Texas Instruments Digital Signal Processor). Applications utilizing the kit will be discussed in Part 2 of this series.

## WHAT IS DSP?

DSP is formally defined as any digital operation performed on an input sequence of numbers. The sequence of numbers may range from stock market data to digitized human speech. DSP can be applied to stock price data to find hidden periodicities just as it can be used to find periodicities (pitch period, formants, and voiced/unvoiced detection) in human speech.

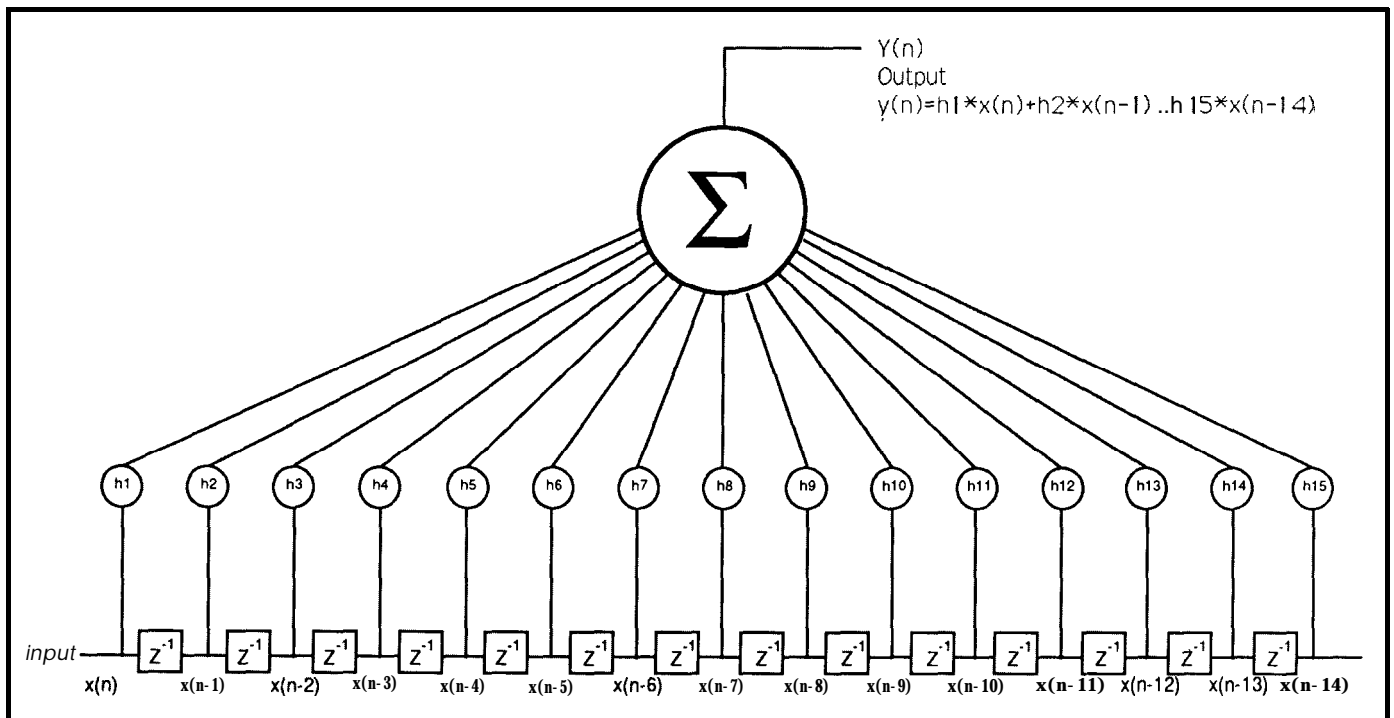


Figure 1 - A 15-tap Finite Impulse Response (FIR) digital filter. Each number in the output is equal to the sum of the inputs above it.

```

'Finite Impulse Response filter program
DIM X(256),H(15),Y(256),F1(256),F2(256)
PI = 3.14159265
DEFINT I,N
LENGTH = 15 'number of taps in digital filter
SCREEN 9,0 'IBM XT EGA Mode 640x350 with 16 colors
CLS
INPUT" SELECT H)ighpass or L)owpass filter: ",B$
IF B$ = "L" OR B$ = "l" THEN

*****
' FINITE IMPULSE RESPONSE FILTER COEFFICIENTS
' THESE COEFFICIENTS FORM A LOWPASS FILTER WHOSE CORNER
' FREQUENCY IS AT 0.03125*Fs (or 312.5 Hz for a 10-kHz Fs)
*****
h(1) = -8.996965363621712E-003 : h(2) = 7.864067330956459E-003
h(3) = 3.349926695227623E-002 : h(4) = 6.534869968891144E-002
h(5) = 9.889715164899826E-002 : h(6) = 0.1285655647516251
h(7) = 0.1489759534597397 : h(8) = 0.15625
h(9) = 0.1489759534597397 : h(10) = 0.1285655647516251
h(11) = 9.889715164899826E-002 : h(12) = 6.534869968891144E-002
h(13) = 3.349926695227623E-002 : h(14) = 7.864067330956459E-003
h(15) = -8.996965363621712E-003

ELSE

*****
' FINITE IMPULSE RESPONSE FILTER COEFFICIENTS
' THESE COEFFICIENTS FORM A HIGHPASS FILTER WHOSE CORNER
' FREQUENCY IS AT 0.1250*Fs (or 1250 Hz for a 10-kHz Fs)
*****
h(1) = 8.996965363621712E-003 : h(2) = -7.864067330956459E-003
h(3) = -3.349926695227623E-002 : h(4) = -6.534869968891144E-002
h(5) = -9.889715164899826E-002 : h(6) = -0.1285655647516251
h(7) = -0.1489759534597397 : h(8) = 0.84375
h(9) = -0.1489759534597397 : h(10) = -0.1285655647516251
h(11) = -9.889715164899826E-002 : h(12) = -6.534869968891144E-002
h(13) = -3.349926695227623E-002 : h(14) = -7.864067330956459E-003
h(15) = 8.996965363621712E-003

END IF

' BUILD WAVEFORM TO FEED INTO FILTER
' This waveform will consist of a 312.5-Hz + 1250-Hz sine wave.
' To provide a series of discrete values at the sampling
' rate (Fs), which is 10000 Hz for this example, we determine
' the number points per cycle associated with each sine wave.
' For example, we know that at 0.5 Fs, it takes 2 points/cycle.
' So, for 312.5 Hz, we have 10000/312.5 or 32 points/cycle,
' and at 1250 Hz, we have 10000/1250 or 8 points/cycle.

FOR i = 0 TO 255
  F1(i+1) = SIN(2*PI*I/32): ' generate 312.5-Hz signal
  F2(i+1) = SIN(2*PI*I/8): ' generate 1250 Hz signal
  x(i+1) = F1(i+1) + F2(i+1): ' mix the two together
NEXT

CLS
' PLOT WAVEFORM
FOR I = 1 TO 256
  LINE(50+I*2,75-25*X(I))-(50+I*2+1,75-25*X(I+1)),14
NEXT
LOCATE 6,2:PRINT"In";
LOCATE 15,2:PRINT"Out";

LINE(0,0)-(639,349),3,B
' SHOW ZERO VOLTAGE LINE
LINE(50,75)-(499+50,75),3,,&H8888
LINE(50,200)-(499+50,200),3,,&H8888
LINE(50,200)-(50,200),14

'PERFORM FIR FILTER FUNCTION
FOR N = 1 TO 256
  FOR I = 1 TO LENGTH
    Y(N) = Y(N) + H(I) * X(N-I)
  NEXT
  LINE-(N*2+59,200-25*Y(N)),14
NEXT

100 A$=INKEY$:IF A$="" THEN 100
STOP

```

Listing 1 — FIR.BAS is a Turbo BASIC implementation of a finite Impulse Response digital filter. The output of the program illustrates both lowpass and highpass F/R filter operation.

Where analog circuits operate on continuous signals, digital signal processing functions operate on a sequence of numbers. Most readers will be familiar with the fact that analog signals (whose formal signal classification is continuous-time, continuous-amplitude) require conversion to a digital representation (discrete-time, discrete-amplitude classification) prior to digital signal processing operations taking place. (See "Preparing Analog Signals for DSP" on page 36).

Classical DSP functions generally include:

- Digital Filtering (Lowpass, Highpass, Bandpass, Bandstop, and Multiple Band filters)
- Discrete Fourier Transforms (used to determine periodic frequency content of a signal frame)
- \*Signal Modulation (generation of sinusoidal waveforms),
- Autocorrelation (used to determine presence of periodic signals)
- \*Cross-correlation (used to determine presence of a periodic signal with known characteristics)

## THE COMMON FILTER

Digital filtering is probably the most common DSP application. The fascinating thing about digital filtering is that it can be accomplished with a simple "sum-of-products" operation using current and past input samples multiplied by coefficients. Equipped with only delays (which can be accomplished by storing samples in memory locations), multiplications, and additions, we can perform lowpass, highpass, bandpass, and bandstop filtering of digitized analog signals. The values of the coefficients determine the characteristics of the filter; that is, whether the filter passes high frequencies, low frequencies, or a band of frequencies. The number of delays (or how many past input samples we keep in memory) and the precision of the coefficients used determine the sharpness or performance of the filter (how quickly the filter rolls off in the transition band). The Finite

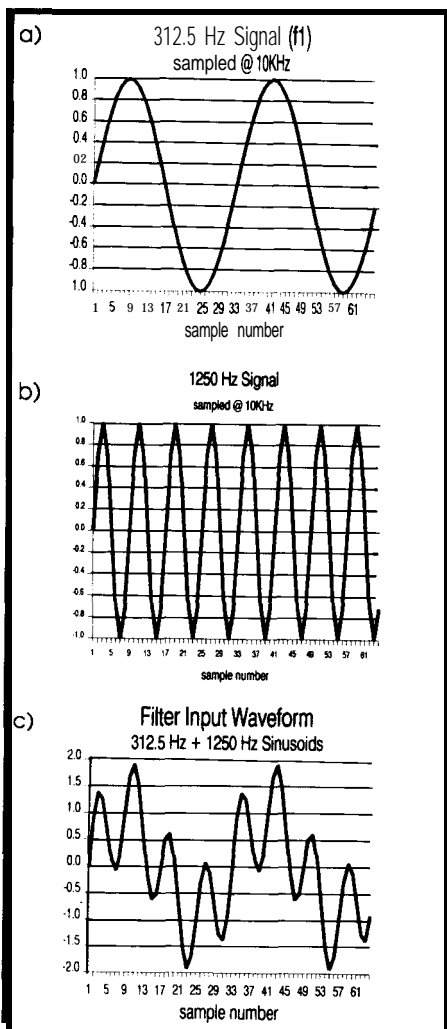


Figure 2—When the waveforms from (a) and (b) are added together, the result is the waveform shown in (c). This result can then be further manipulated through a highpass or lowpass filter with the characteristics listed in Listing 1.

Impulse Response filter, or FIR, is one of two major filter types which: uses only past and current inputs, provides unconditional stability, and does not have a counterpart in the analog world. The other type, Infinite Impulse Response (IIR), uses past outputs fed back into the input and sometimes suffers from instability. An unstable IIR filter can oscillate just like an unstable op-amp circuit. The term “finite impulse response” means that after an occurrence of a unit impulse at the input, the output eventually settles down to zero. An “infinite impulseresponse” filter can, however, theoretically go to infinity upon entering an unstable state.

Figure 1 illustrates a 15-tap FIR digital filter. Input samples enter at

the left end of the diagram and move to the right through each delay element as each new sample is ready. The newest input sample is  $x(n)$ . The previous input sample (delayed by one sample period) is  $x(n-1)$ . The sample before that is  $x(n-2)$  and so on until  $x(n-14)$  is reached. Input samples older than  $x(n-14)$  are simply discarded. With each new sample, a sum-of-products cycle is performed in which current and past inputs are multiplied by their respective coefficients. The summation symbol indicates that each number in the output sequence  $y(n)$ , is equal to  $x(n)*h1 + x(n-1)*h2 + x(n-2)*h3...x(n-14)*h15$ . To illustrate the FIR filter operation, we will introduce Listing 1, FIR.BAS, a Turbo BASIC program which graphically illustrates lowpass and highpass FIR filter operation. The program runs on an IBM PC-compatible equipped with EGA graphics. *[Editor's Note: Software for this article is available for downloading from the Circuit Cellar BBS or on Software On Disk #13. For downloading and purchasing information, see page 78.1]* The input waveform consists of two frequencies: 312.5 Hz ( $f1$ ) and 1250 Hz ( $f2$ ). Our hypothetical sampling rate will be 10 kHz. This means that each input sample will be spaced 100  $\mu$ s apart. In order to generate a 312.5-Hz signal, we need  $10\text{ kHz} \div 312.5\text{ Hz} = 32$  discrete points per cycle. To generate the 1250-Hz signal, we need  $10\text{ kHz} \div 1250\text{ Hz} = 8$  points per cycle. Notice that the denominators in the  $SIN(2*\pi*I/Cyc)$  and  $SIN(2*\pi*I/Cyc1)$  statements determine the number of points which define each cycle. Adding the waveforms shown in Figures 2a and 2b, we get the composite waveform shown in 2c. To filter out the higher of the two, leaving only the low frequency (312.5 Hz), we use a lowpass filter. The desired lowpass characteristics are determined by the values of the  $h(n)$  coefficients. The filter coefficients are generated using the Fourier Coefficient method. The resultant values are shown in the first block of parameters in Listing 1. Figure 3a illustrates a plot of the lowpass filter coefficients.

Inputting the waveform point by point into the lowpass filter and per-

forming a sum-of-products loop, Figure 3b results. Running a compiled version of FIR.BAS will demonstrate the operation of the lowpass filter. Notice that until at least 15 samples have entered the filter, the full effect of the lowpass filter has not occurred. The filter is 15 taps long. Therefore, the output lags the input in phase by 15 taps \* 100  $\mu$ s/tap or 1.5 milliseconds. The frequency response of this filter is shown in Figure 3c. By simply changing the coefficients to those shown in the second block of parameters in Listing 1, we can do the opposite: construct a highpass filter.

Performing the same process using different coefficients (plotted in Figure 4a) gives the results shown in Figure 4b. The high frequency (1250

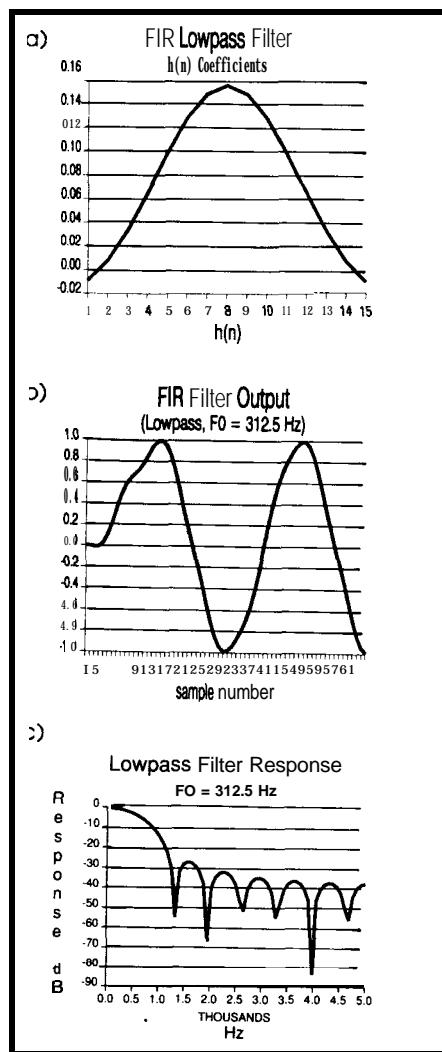


Figure 3—(a) is a plot of the lowpass coefficients listed in Listing 1. When a waveform is input and sum of products performed, (b) results. The filter response is shown in (c).

Hz) passes while the low frequency (312.5 Hz) is attenuated. Note also, the highpass filter frequency response in 4c. (For a brief explanation of DSP filter theory, see "How does digital filtering work?" on page 00.)

## A TRIO OF TRANSFORMS

Discrete Fourier Transforms (DFT) indicate the frequencies present within a frame of data. While the Fourier Transform operates on continuous signals (continuous in time and amplitude), the DFT operates on discrete samples. The DFT algorithm is a sum-of-products operation where each sample in the frame is multiplied by values of a stored sinusoidal wave. Each point in the DFT represents a frequency bin. For a 256-point DFT, we would store 256 points of a sine wave (from 0 to 360 degrees) in a table. To evaluate the content of frequency bin 1, we would multiply each frame sample by each point in the table. The resulting sum of products, divided by 256 and then squared, would represent the power spectrum for *bin 1*. To evaluate bin 2, each frame sample would be multiplied by every other value in the table modulo 256. To evaluate *bin 3*, each frame sample would be multiplied by every third table value modulo 256 and so on, until the last *bin* has been evaluated. One hundred twenty-eight frequency *bins* would result from a 256-point DFT. If the sampling frequency were 8,192 samples/second, each *bin* would represent 32-Hz resolution. The Fast Fourier Transform (FFT) is a computationally efficient version of the DFT. DFT.BAS, shown in Listing 2, is a Turbo BASIC program that demonstrates the DFT algorithm. The program asks for a relative percentage of noise which is added to a sine wave whose frequency is specified in terms of points/cycle. You could easily change the program to accept 512 points of data from a file instead of generating a noisy sine wave.

## CORRELATING SIGNALS

Autocorrelation is a computationally intensive process of multiplying a

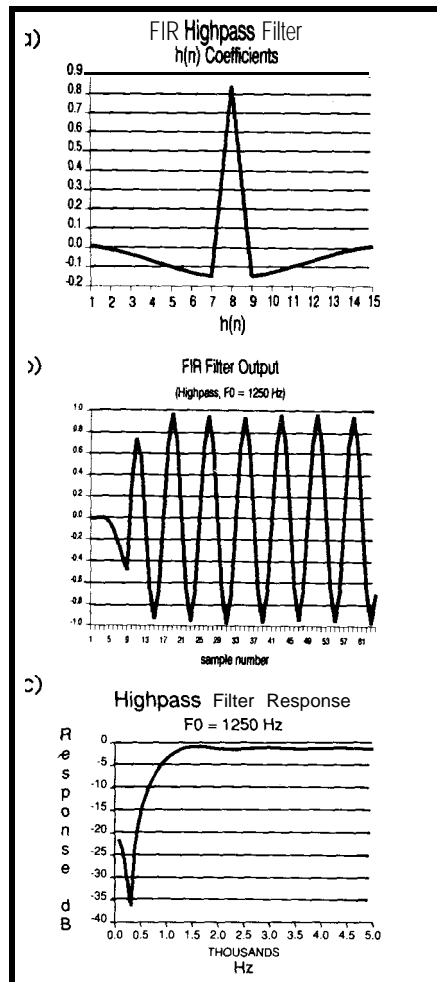


Figure 4—Using the coefficients listed in Listing 1 on the waveform shown in (a) results in the pattern pictured in (b), the highpass filter frequency response is shown in (c).

frame of samples with a shifted replica of the same frame. As shown in Figure 5, the duplicated frame is shifted over one point and multiplied by the corresponding points in the unshifted frame. The sum of products of each shift forms each point in the autocorrelation output. As an example, let's say we're looking for the presence of periodic signals buried in noise yet don't know anything about the characteristics of the signal. This is an ideal job for autocorrelation. A frame of received signal samples has been autocorrelated with a duplicated received frame as shown in Figure 6. Notice the improvement in signal-to-noise ratio between the original signal and the autocorrelated output signal. The spike preceding the periodic wave indicates a large amount of noise within the received signal. Autocor-

# Digital Signal Processing

DSP Development Tools and Standalone Systems from Ariel

## From the IBM PC:

**DSP-16** • A complete TMS320C2 or TMS320C25 Development System on a single board, with 16-bit 2 channel data acquisition of up to 50 kHz per channel.

**PC-C25** • The lowest cost full speed TMS320C25 oased card available. Just \$595 with parallel and serial I/O, 14 bit analog I/O is just \$95!

**DSP-56** • Integer DSP development system based on the Motorola 56000 DSP chip with two channel 16 bit analog I/O, compatible with Ariel's Bug-56.

**PC-56** • A new, low-cost DSP card based on Motorola's fast DSP56001. Full speed 24 bit DSP for \$595! Parallel and serial I/O standard. Available with 14-bit analog A/D, NeXT compatible DSP port and microphone preamp.

**BUG-56** • Fast, efficient symbolic debugger for the PC-56 and DSP-56. Macros, windows, the works. Also available: Assembler/Simulator, C Compiler and TMS320 Code Converter.

**DSP-32C** • Floating Point DSP development system with true 16 bit analog I/O based on AT&T's 32 bit DSP32C chip.

**PC-32C** • Low cost floating point coprocessor based on AT&T's DSP-32C standard with parallel and serial I/O.

**SDI** • A complete, 2 track 16 bit digital audio recorder with advanced editing capabilities. Real-time 50 kHz stereo I/O using any PC.

**SYSid** • Comprehensive acoustic test Instrument. Developed by Bell Labs for quick and accurate measurements.

**PC-FFT** • Fast FFT's on a single card.

**ASM-320** • The fastest TMS320 Assembler.

**PDS-320** • Deluxe TMS320 Program Development, FFT-320 • 256 and 1024 point TMS320 FFT Subroutines. Real-time demo program too.

**FDAS** • Digital FIR and IIR Filter Design with real time Implementation on the DSP-16.

## For You:

Ariel Corporation is dedicated to providing you with the best values in high performance DSP products. Our products are designed, built and maintained in the U.S. The best support in the industry is always at hand. Ariel's products are sold directly throughout North America, and are available worldwide, through our international dealer network.

**Ariel Corporation**  
**433 River Road**  
**Highland Park, NJ 08904**  
**Telephone: 201-249-2900**  
**Fax: 201-249-2123**  
**Telex: 4997279 ARIEL**  
**DSP BBS: 201-249-2124**

# Ariel

Reader Service #103

# How Does Digital Filtering Work?

In order to provide a very simple example of a filter function given a set of noisy data, we can do the following: Take a pair of points, add them together, divide the sum by two, replace the pair with the average of the two, and repeat steps 14 until the end of the data is reached.

What can be noticed from the diagram shown in Figure I, (which illustrates the averaging scheme), is the smoothing process. The simple averaging technique has, in effect, filtered out or attenuated the high-frequency noise. It is fairly easy to understand how this simple filter works. However, it is not intuitively obvious how the 15-tap FIR filter works. For this, we will delve into a little DSP theory. In Figure II, the signals used in the 15-tap FIR filter example are shown in (a), (b), and (c). A pure sine wave is represented in the frequency domain as a dirac pulse (or spike) placed at the associated frequency position on the x-axis. The vehicle which allows continuous time domain signals to be placed in the frequency domain is the Fourier Transform. To convert frequency domain signals to the time domain, an Inverse Fourier Transform is used. Notice, that adding the two frequencies together in (d) and (e) yields the result shown in (f). In order to low-pass filter the composite signal, the following steps can be taken: (refer to Figure III)

1. convert the time domain signal in (d) to the frequency domain in (a)
2. multiply the frequency domain signal in (a) by the desired frequency response shown in (b), (a lowpass filter which passes  $f_1$  and attenuates  $f_2$ )
3. the result of the frequency domain filtering leaves only the lower frequency,  $f_1$ , as shown in (c)
4. converting the frequency domain representation of  $f_1$  to the time domain yields the desired result, the 312.5-Hz sine wave.

This approach can be used to perform filtering, but is costly in computation and is not really very feasible. Let's look for a more efficient way of filtering. We can use a Fourier Transform trick from mathematics which states that "multiplication in the frequency domain is equal to convolution in the time domain." This means that by "convolving" the time domain waveform with a block of special coefficients, we can provide an efficient filter implementation. Notice the special operator,  $**$ , which indicates convolution. Convolution is simply a "sum-of-products" operation characteristic of all FIR filters.

The special coefficients actually represent a time domain version of the desired filter frequency response. These coefficients can be generated through various techniques which include the Fourier Coefficient and Inverse Discrete Fourier Transform (IDFT) methods. Convoluting the time domain signal in Figure III d by the waveform shown in Figure 3a yields the filtered lowpass result shown in Figure III f. By utilizing convolution, we are able to leave out the Fourier transform in steps (d) to (a), and the inverse Fourier Transform in steps (c) to (f) allowing for efficient digital filtering.

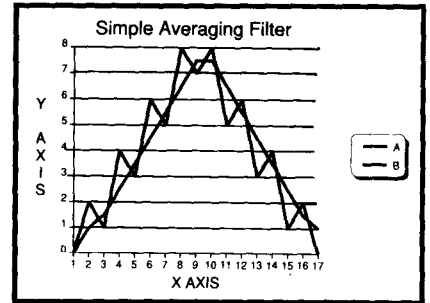


Figure I-A simple filtering technique, such as the averaging scheme shown, can effectively attenuate high-frequency noise.

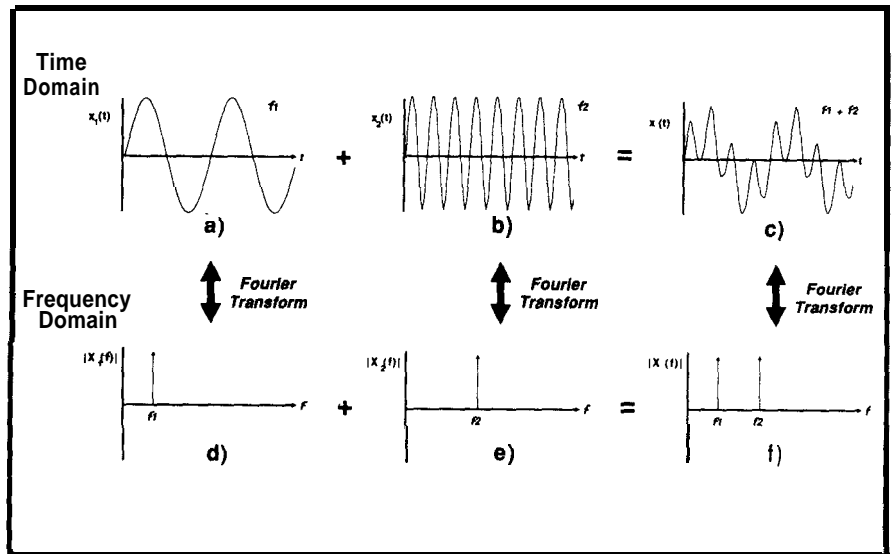


Figure II- (a), (b), and (c) are frequencies used in the 15-tap FIR filter. A sine wave is shown as a spike in the frequency domain. Continuous time domain signals are placed into the frequency domain via a Fourier Transform. The final signal in (f) is the result of adding the signals shown in (d) and (e).

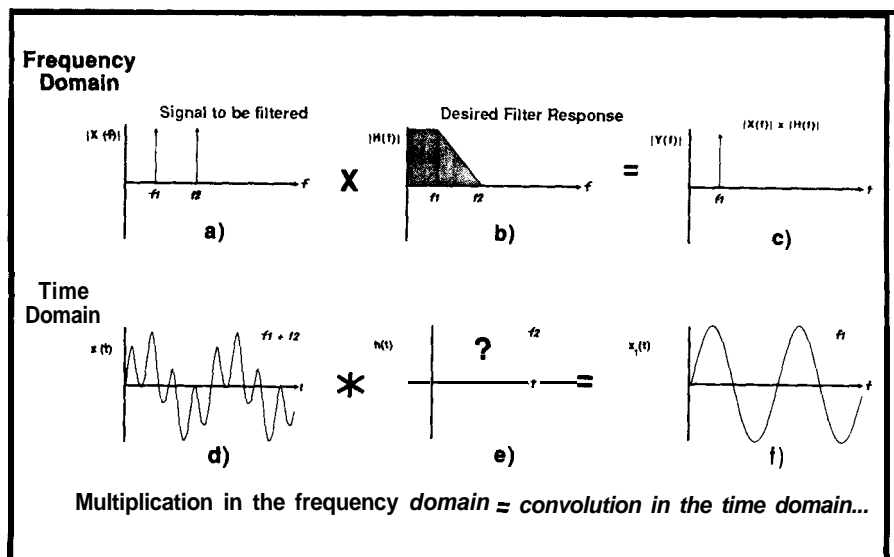


Figure III--The composite signal produced in Figure II can be lowpass filtered using the technique illustrated here, and described in the text.

```

DEFINT N,M,L,K,I,J,E,O:CLS
NSAMP=512:M=LOG(NSAMP)/LOG(2)
N=NSAMP/2:M=M-1
SCREEN 9,0
DIM X(1000):DIM S(63):PI=3.14159265
INPUT"please enter percent noise to be added to signal ",NS
INPUT"please enter points/cycle desired ",CY
'BUILD WAVEFORM
FOR I = 0 TO NSAMP
  sn = ((rnd(1)*2)-1)*NS/100
  X(I) = SIN(2*PI*I/CY)+sn
NEXT
CLS
LINE(0,0)-(639,349),3,B
LINE(50,75)-(512+50,75),3,,&H8888 : ' SHOW ZERO VOLTAGE LINE
'PLOT NOISY WAVEFORM
FOR I = 0 TO 512
  LINE(I+50,75-25*X(I))-((I+1)+50,75-25*X(I+1)),14
NEXT

LINE(45,300)-(512+50,300),3,,&H8888 : ' SHOW -100 DB LINE
LINE(45,280)-(512+50,280),3,,&H8888 : ' SHOW -80 DB LINE
LINE(45,260)-(512+50,260),3,,&H8888 : ' SHOW -60 DB LINE
LINE(45,240)-(512+50,240),3,,&H8888 : ' SHOW -40 DB LINE
LINE(45,220)-(512+50,220),3,,&H8888 : ' SHOW -20 DB LINE
LINE(45,200)-(512+50,200),3,,&H8888 : ' SHOW 0 DB LINE
LOCATE 22,2 : PRINT"-100"
LOCATE 19,2 : PRINT"-CO";
LOCATE 15,2 : PRINT" 0 dB";
LOCATE 24,6 : PRINT"0";
LOCATE 24,71 : PRINT".5 FS";
LOCATE 12,33 : PRINT"Power Spectrum";
X=50 : Y=303 : co=15 : GOSUB ARROW

GOSUB FFT : ' PERFORM FFT FUNCTION

' GET MAGNITUDE OF EACH FREQUENCY
FOR L=0 TO N-1 : X(L)=(X(2*L)^2+X(2*L+1)^2): NEXT
' FIND LARGEST MAGNITUDE
xMax=o
FOR L=0 TO N-1
  IF X(L) > XMAX THEN XMAX=X(L)
NEXT
' CONVERT TO DBs
FOR L=0 TO N-1
  X(L)=10*LOG10(X(L)/XMAX)
NEXT
' FIND SMALLEST VALUE
xMin=0
FOR L=0 TO N-1
  IF X(L) < XMIN THEN XMIN=X(L)
NEXT
XSCALE=512/N
' PLOT SPECTRUM
FOR L=0 TO N-1
  XP=L*XSCALE
  LINE(XP+50,300)-(XP+50,300-100*(1-(X(L)/-100))),14
NEXT
100 A$=INKEY$: IF A$="" THEN 300
IF LEN(A$)=2 THEN
  IF ASC(MID$(A$,2,1))=77 AND (X<=562) THEN
    CO=0
    GOSUB ARROW
    X=X+2
    CO=15
    GOSUB ARROW
  END IF
  IF ASC(MID$(A$,2,1))=75 AND (X>50) THEN
    CO=0
    GOSUB ARROW
    X=X-2
    CO=15
    GOSUB ARROW
  END IF
END IF
IF A$="Q" OR A$="q" THEN STOP
GOTO 100
STOP

```

(continued)

listing 2-A Discrete Fourier Transform is implemented in DFT.BAS.DFT.BAS generates a sine wave whose frequency is user specified, and adds to it a specified percentage of noise. The DFT then works to extract the relevant components of the signal from the noisy whole.

# étude™

## 25 MHz 8-bit ANALOG-TO-DIGITAL CONVERTER

Based on the TRW THC1068-1 hybrid flash converter, its high signal-to-noise ratio yields excellent accuracy at the Nyquist limit.

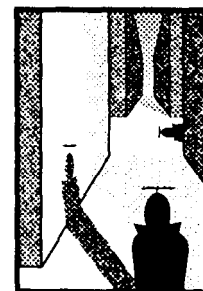
- 4 KB of cache SRAM or to host as converted at DMA speed
- I/O or DMA data transfer
- 10 MHz full-power bandwidth
- 3.92 mV resolution
- 16 jumper selectable base addresses
- External clock and trigger inputs TTL compatible
- Software source code included

### \$495<sup>00</sup>

ALSO AVAILABLE AS A KIT  
FOR \$99, INCLUDING:

- Printed Circuit board
- Software
- Manual & assembly instructions

Requires: PC compatible 1/2 length 8-bit expansion slot DOS 2.11 or greater. EGA, VGA or Hercules display needed for graphic representation of data.



**Silicon Alley Inc.**

P. O. BOX 59593  
RENTON, WA 98058  
206. 255. 7410

© 1989 Silicon Alley Inc. étude is a trademark of Silicon Alley Inc. Other brand or product names are trademarks or registered trademarks of their respective holders. Prices and specifications subject to change.

```

' DRAW ARROW
ARROW:
LOCATE 24,30:PRINT" ";
LOCATE 24,35
PRINT"F = ";
PRINT ((X-50)/2) * (.5/256);
PRINT" Fs";
LINE(X,Y)-(X,Y+10),CO
PSET(X+1,Y+1),CO
PSET(X-1,Y+1),CO
PSET(X+1,Y+2),CO
PSET(X-1,Y+2),CO
PSET(X+2,Y+3),CO
PSET(X-2,Y+3),CO
LINE(X,Y-116)-(X,Y-106),CO
PSET(X+1,Y-106-1),CO
PSET(X-1,Y-106-1),CO
PSET(X+1,Y-106-2),CO
PSET(X-1,Y-106-2),CO
PSET(X+2,Y-106-3),CO
PSET(X-2,Y-106-3),CO
RETURN

FFT:
' FFT Subroutine
PI=3.1415926535
' BEGIN COMPLEX INPUT FFT
K2=1
FOR K=1 TO M
K2=2*K2
K1=K2/2
TH=(2*PI/N)*K1
NK=N/K2
N1=N/K1
FOR I=1 TO NK
AG=(I-1)*TH
WR=COS(AG)
WI=-SIN(AG)
FOR J=N1 TO N STEP N1
J1=J-N1+(I-1)
J2=J1+NK
E1=2*J1
E2=2*J2
O1=E1+1
O2=E2+1
DR=X(E1)-X(E2)
DI=X(O1)-X(O2)
X(E1)=X(E1)+X(E2)
X(O1)=X(O1)+X(O2)
X(E2)=WR*DR-WI*DI
X(O2)=WI*DR+WR*DI
NEXT
NEXT
NEXT
' END OF COMPLEX INPUT FFT

```

```

' PERFORM BIT REVERSAL
J=0:
FOR I=0 TO N-2
IF I<J THEN
EI=2*I
EJ=2*J
T=X(EI)
X(EI)=X(EJ)
X(EJ)=T
T=X(EI+1)
X(EI+1)=X(EJ+1)
X(EJ+1)=T
END IF
K=N/2
AGAIN:
IF K>J THEN
J=J+K
ELSE
J=J-K
K=K/2
GOTO AGAIN
END IF
NEXT
' UNPACK ARRAY TO GET POSITIVE
' FREQUENCY SPECTRUM
T=X(0)+X(1)
X(1)=X(0)-X(1)
X(0)=T
X(N+1)=-X(N+1)
N2=2*N
FOR K=1 TO N/2-1
K2=2*K
NK=N2-K2
AR=X(K2)+X(NK)
AI=X(K2+1)-X(NK+1)
BR=X(NK)-X(K2)
BI=-X(NK+1)-X(K2+1)
S=SIN(K*PI/N)
C=COS(K*PI/N)
CR=BR*S-BI*C
CI=BR*C+BI*S
X(K2)=(AR+CR)/2
X(K2+1)=(AI+CI)/2
X(NK)=(AR-CR)/2
X(NK+1)=(CI-AI)/2
NEXT
RETURN
' END FFT SUBROUTINE

```

## Preparing Analog Signals for DSP

The sequence of numbers (or data) which DSP algorithms operate on may originate from daily stock market statistics or weekly sales figures for a consumer product. For continuous analog signals, however, which are found in seismic, EKG, radar, speech, and digital communications applications (to name a few), conversion to discrete amplitudes within specified time intervals is required prior to proceeding with DSP operations. The time interval between each number in a sequence may be microseconds, milliseconds, days, weeks, or even years. "Sampling" is the process of converting a continuous time/continuous amplitude signal into a discrete time/discrete amplitude numeric representation of the signal. The analog signal is converted to a series of numbers through the use of an Analog-to-Digital converter (ADC). A sample-hold circuit (placed in front of the ADC) "freezes" the input waveform while the conversion takes place. At time  $T_{later}$ , (the reciprocal of the sampling rate) the process repeats itself yielding a series of numbers which represent the analog signal. The sampling rate  $F_s$ , must be at least twice the highest frequency component of interest in order to accurately represent signals within the bandwidth of interest. In addition, a lowpass analog filter with a corner frequency of  $0.5F_s$  must be placed in front of the sample-hold circuit in order to defeat the phenomenon of "aliasing." Aliasing allows high frequencies ( $>0.5F_s$ ) to take on the identity of low frequencies ( $<0.5F_s$ ) and pass through the ADC, thus disrupting the DSP system. The resolution of the A/D converter depends on the number of bits utilized. For example, an 8-bit A/D converter capable of handling a  $\pm 2.5\text{-V}$  signal (full scale) would provide  $5\text{ V}/256$  levels or 20-mV-per-bit resolution. Once the proper lowpass filter, sample-hold, ADC, and DAC circuits have been designed, DSP operations on analog signals may take place.

listing 2-continued

relation is a powerful means of detecting periodic signals in noise. If a received noisy signal having a signal/noise ratio of -3 dB (which means that the noise is twice that of the signal) were 256-point autocorrelated with a replica of itself, the resulting output would have a signal-to-noise ratio of 8.2 dB, a gain of 11.2 dB!

Cross-correlation is similar to autocorrelation except that the frame of samples is correlated with a different frame of samples instead of with itself. It is used (instead of autocorrelation) when the nature of the signal (shape and frequency) is known beforehand. For example, let's say we are about to receive a periodic signal

buried in noise and know beforehand that the signal will consist of  $1\frac{1}{2}$  cycles of a 1000-Hz sine wave. It is this fact that allows us to use cross-correlation rather than autocorrelation. This a priori knowledge is used to build a frame of samples which matches the characteristics of the signal itself, namely,  $1\frac{1}{2}$  cycles of a sine wave at 1000 Hz. The received signal is cross-correlated with the known signal characteristics (see Figure 7). The improvement in signal-to-noise ratio between the original signal and the cross-correlated signal can be significant. If a received noisy signal having a signal-to-noise ratio of -3 dB were 256-point cross-correlated, the result-



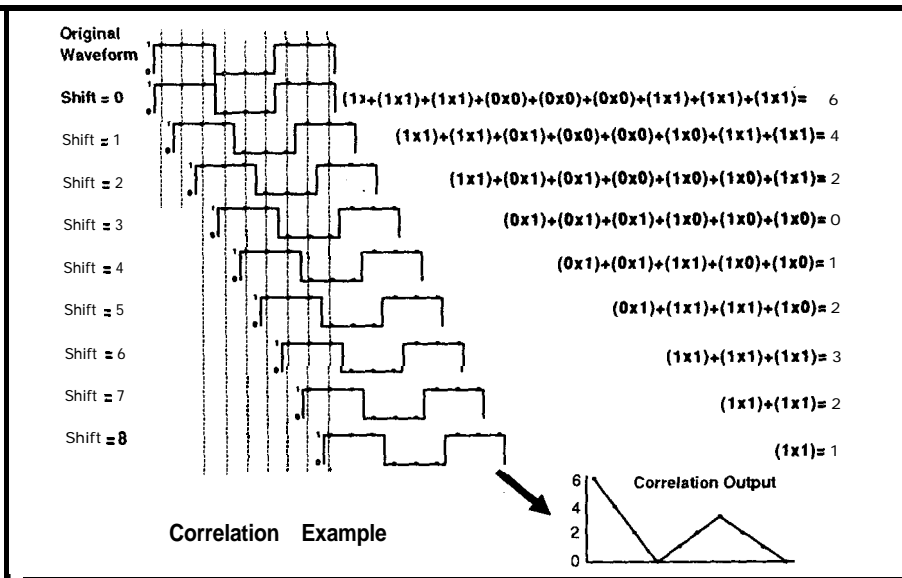


Figure 5—Autocorrelation is a computationally intensive process of multiplying a frame of samples by a shifted replica of the same frame. The duplicated frame is shifted one point and multiplied by the corresponding points in the unshifted frame.

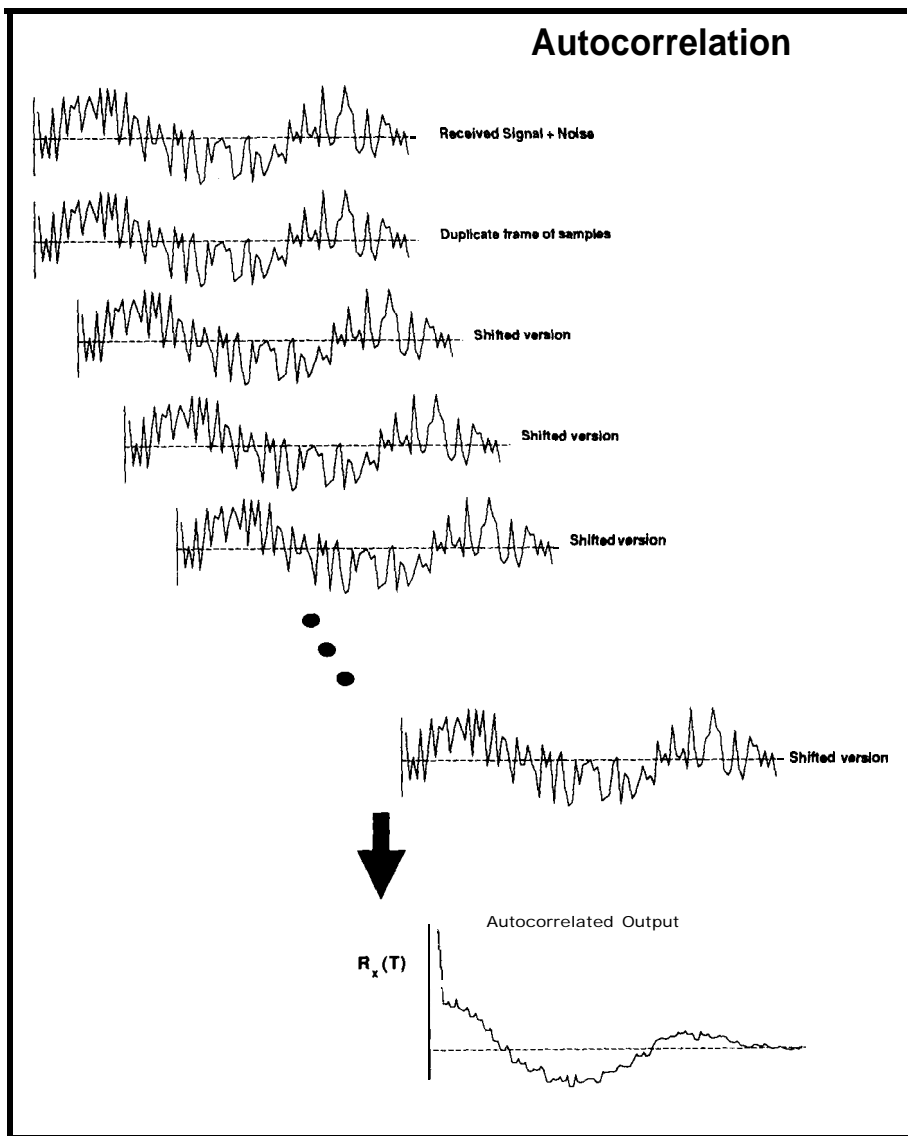


Figure 6—The result of autocorrelation. The signal-to-noise ratio shows dramatic improvement between the original signal and the autocorrelated output signal.

## 8031 $\mu$ Controller Modules

NEW!!!

### Control-R II

- / Industry Standard 8-bit 8031 CPU
- / 128 bytes RAM / 8 K of EPROM
- ✓ Socket for 8 Kbytes of Static RAM
- ✓ 11.0592 MHz Operation
- ✓ 14/16 bits of parallel I/O plus access to address, data and control signals on standard headers.
- ✓ MAX232 Serial I/O (optional)
- ✓ +5 volt single supply operation
- ✓ Compact 3.50" x 4.5" size
- ✓ Assembled & Tested, not a kit

\$64.95 each

### Control-R I

- ✓ Industry Standard 8-bit 8031 CPU
- ✓ 128 bytes RAM / 8K EPROM
- ✓ 11.0592 MHz Operation
- ✓ 14/16 bits parallel I/O
- ✓ MAX232 Serial I/O (optional)
- ✓ +5 volt single supply operation
- ✓ Compact 2.75" x 4.00" size
- ✓ Assembled & Tested, not a kit

\$39.95 each

#### Options:

- MAX232 I.C. \$6.95 each
- 6264 8K SRAM \$10.00 each
- 8052BASIC CPU \$25.95 each

#### Development Software:

- PseudoSam 51 Software (\$50.00) Level II MSDOS cross-assembler. Assemble 8031 code with a PC.
- PseudoMax 51 Software (\$100.00) MSDOS cross-simulator. Test and debug 8031 code on your PC!

#### Ordering Information:

Check or Money Orders accepted. All orders add \$3.00 S&H in Continental U or \$6.00 for Alaska, Hawaii and Canada. Illinois residents must add 6.25% tax.

Cottage Resources Corporation  
Suite 3-672, 1405 Stevenson Drive  
Springfield, Illinois 62703  
(217) 529-7679

## Crosscorrelation

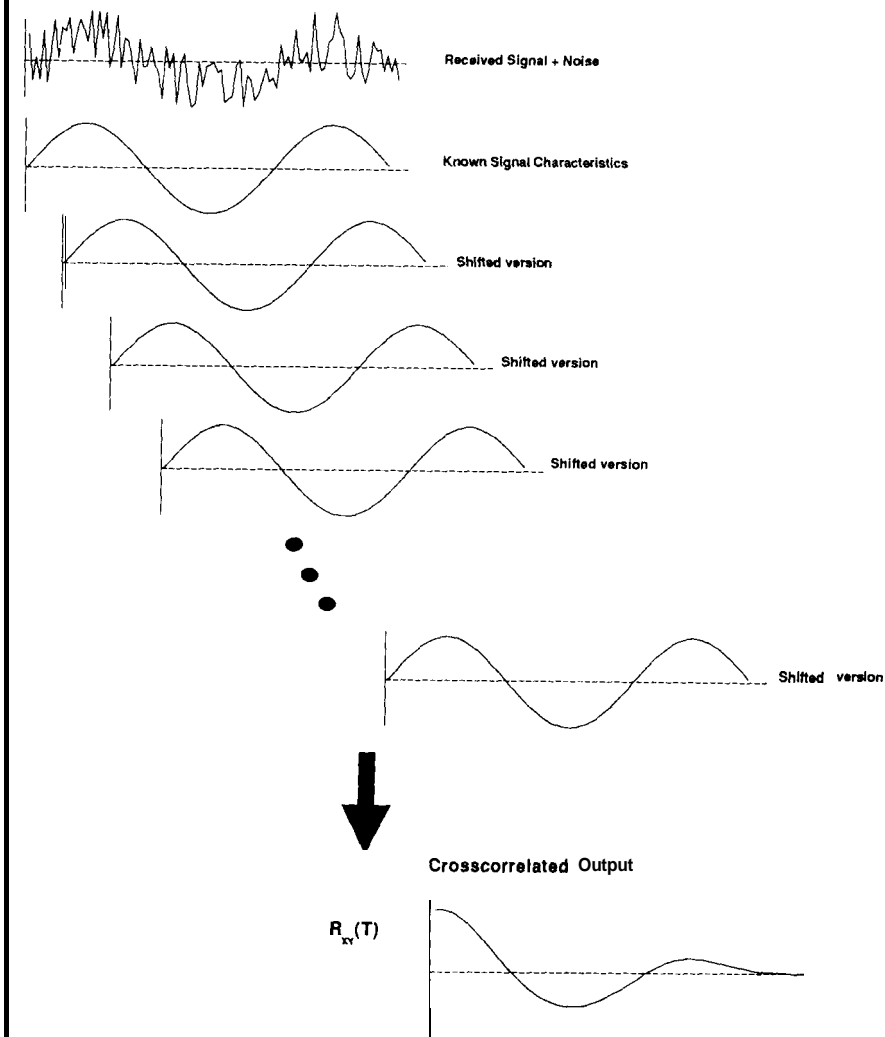


Figure 7 -Cross-correlation offers more dramatic signal-to-noise ratio improvement than does autocorrelation, but requires prior knowledge of the signal's characteristics.

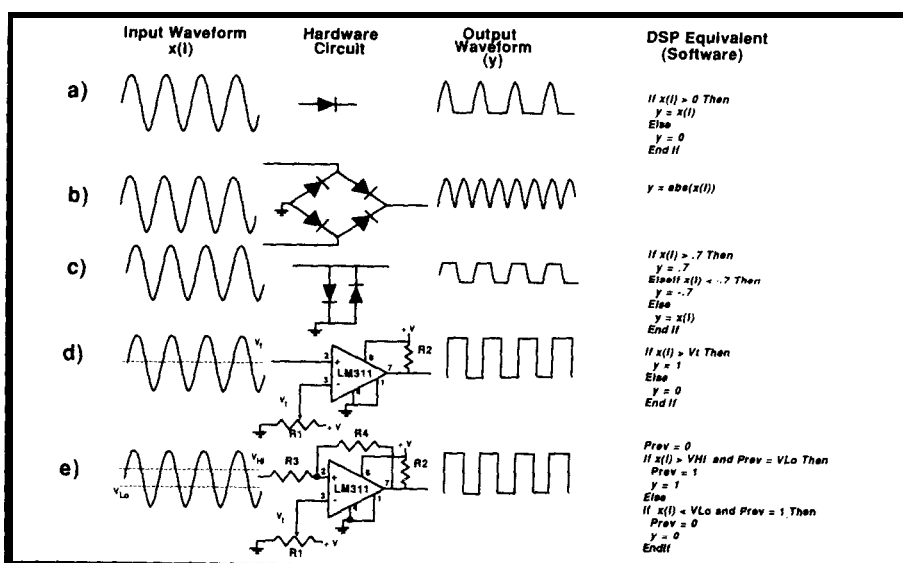


Figure 8—Many common analog electronic circuits can be successfully replaced by a DSP chip with appropriate software. Using the DSP approach offers the advantage of easy on-the-fly changes when compared with physically changing components.

ing output would have a signal-to-noise ratio of 14.5 dB, a gain of 17.5 dB! This represents an improvement of 6.3 dB over autocorrelation.

## REPLACING ANALOG CIRCUITS

Although the foundation of DSP has been built on digital filtering and spectral analysis, simpler, more mundane functions such as signal rectification (digital counterpart of a diode), threshold excursion detection (analog comparator circuit counterpart), peak clipping (digital equivalent of a pair of diodes connected in a clipping configuration), and signal level measurement (digital counterpart of an RMS voltmeter) qualify as legitimate Digital Signal Processing applications, too.

Figure 8 shows a few commonly used electronic circuits and their DSP software equivalents. Pseudocode is used to describe the DSP software design. A single diode which performs a half-wave rectification function (see Figure 8a) is easily performed in DSP software as are full-wave rectifiers and peak clippers.

As a more in-depth example of nonclassical DSP, we will look at a comparator function performed in DSP software as shown in Figure 8d. Many of you have probably built comparator circuits around ICs like the LM311. The input signal is applied to the "+" input while the threshold voltage is applied to the "-" pin. The output goes to +Vcc as long as the input voltage exceeds the threshold voltage. Otherwise, the output goes to ground or V- (depending on how you connect pin 1). A variable resistor allows for changes in the setting of the threshold voltage. The threshold voltage level in a DSP-based comparator is a variable,  $V_t$ , instead of a variable resistor. To change the threshold level, you change the software statement which initializes the threshold variable. To add hysteresis to the comparator for noise immunity, (as shown in Figure 8e), the incoming digitized waveform is first compared to an upper threshold. After detecting an excursion above the upper threshold,  $V_{Hi}$ , the comparator output remains in the high state until an excursion be-

low a lower limit,  $V_{\text{L}}$ , occurs. The output goes low, and the process repeats. The range between the upper and lower threshold levels is the hysteresis voltage. This hysteresis precludes noise excursions within the window affecting the output transitions. In a DSP approach, the amount of hysteresis is set by program variables coupled with program code, allowing easy changes. The program HYSTER.BAS shown in Listing 3 is a Turbo BASIC program which illustrates DSP comparator functions (with and without hysteresis).

Why perform functions in the digital domain (via DSP) when a diode, comparator, or analog filter (constructed of resistors, capacitors, and an op-amp) will suffice? The answer is flexibility. DSP functions are typically performed in software allowing "circuit" characteristics/transfer functions to be shaped through program statements instead of analog components like resistors, capacitors, diodes, and op-amps whose characteristics vary with tem-

perature and age. This is analogous to the early use of general-purpose microprocessors to replace digital logic in electronic designs.

### DSP IMPLEMENTATION

The previously mentioned BASIC programs demonstrate DSP fundamental concepts, but you may be wondering how a real application could be performed. If you equip your PC with an A/D converter board, you can capture analog signals to disk, then perform filtering in nonreal time. However, if real-time filtering is desired, your PC won't perform very well (at least for frequencies  $>100$  Hz). In a real-time PC-based filtering setup, the analog signal would be connected to the A/D converter. The A/D converter would be read through an I/O port. Filtering would consist of summing a list of products of coefficients multiplied by past stored input samples. The sum-of-products operation would occur with each new sample's arrival. The PC could either

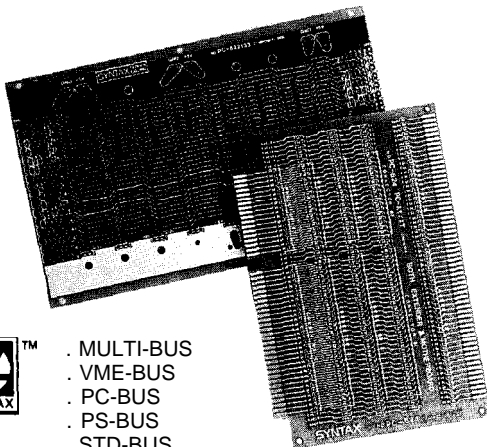
be interrupt driven or could poll the "end of conversion" pin on the A/D converter. For a sampling rate of 8000 Hz, (which is required for toll-quality speech), the sum-of-products operation must occur every 125  $\mu$ s. For a 15-tap filter, the following steps must occur for each sample:

1. shift each of the past 15 sample values down one place in the storage buffer (drop the oldest sample)
2. read the current sample from the A/D converter and place in the first location of the sample buffer
3. multiply each current and past sample by the 15 filter coefficients (accumulate each product)
4. output the sum to the D/A converter
5. wait for the next A/D sample.

On my 8-MHz IBM PC clone (equipped with a 286 accelerator board), it takes approximately 14 ms to perform the steps outlined above.

## PROTOTYPING SOLUTION

We offer the best quality  
SYNTAX FR-4 prototyping PCB at the best price!  
All from the stock



- . MULTI-BUS
- . VME-BUS
- . PC-BUS
- . PS-BUS
- . STD-BUS
- S100-BUS
- . TRANS BOARD
- UNI BOARD

. Please Call Today for Catalog  
'Custom Design are Available  
'Dealer program offered

Manufacturing & Marketing

**MING**<sup>TM</sup>  
Engineering & Products, Inc.

Tel: (818) 570-0058 • Fax: (818) 576-8748

## LEARN MULTITASKING

**Smx**

### COLLEGE KIT \$95

The **smx** College Kit allows you to learn and to experiment with multitasking. It includes demo source code, a tutorial **User's Guide**, a **Reference Manual**, and help files – everything you need to try out multitasking! The CK is a reduced version of our **simple multitasking executive**. It runs at full speed and is ROM'able. Use it with MS-DOS or stand-alone. Works with the 66x86 family and **Microsoft** or Turbo C and assembler. The full version of **smx** is available for \$1495 with no royalties.

**MICRO pd DIGITAL**

1 - 800 - 366 - 2491

FAX 714-691-2363

CYPRESS, CA 90630-5630

```
'DEMONSTRATES A COMPARATOR FUNCTION WITH AND
'WITHOUT HYSTERESIS
DIM X(1000)
PI = j.14159265
DEFINT I
SCREEN 9,0
INPUT"Enter percent noise to be added to signal ",N
INPUT"Enter points/cycle desired ",CY
FOR I = 0 TO 501
  sn = ((rnd(1)*2)-1)*n/100
  X(I) = SIN(2*PI*I/CY)+sn
NEXT
CLS
LOCATE 6,2:PRINT"A";
LOCATE 15,2:PRINT"B";
LOCATE 20,2:PRINT"C";
LOCATE 5,72:PRINT"VHI";
LOCATE 1,72:PRINT"VLO";
LOCATE 6,72:PRINT"0 volts";
LOCATE 14,72:PRINT"1";
LOCATE 16,72:PRINT"0";
LOCATE 19,72:PRINT"1";
LOCATE 21,72:PRINT"0";
LINE(0,0)-(639,349),3,B
VHI = .45
VLO = -.45
' ASSUME OUTPUT INITIALLY OFF
VOUT = -1
PREV = VLO
' SHOW UPPER THRESHOLD
LINE(50,75-50*VHI)-(499+50,75-50*VHI),4,,&HOF0F
' SHOW LOWER THRESHOLD
LINE(50,75-50*VLO)-(499+50,75-50*VLO),4,,&HOF0F
' SHOW ZERO VOLTAGE LINE
LINE(50,75)-(499+50,75),3,,&H8888
' PLOT NOISY WAVEFORM
FOR I = 0 TO 500
```

```
LINE(I+50,75-50*X(I))-((I+1)+50,75-50*X(I+1)),1,4
NEXT
' DO COMPARATOR FUNCTION WITHOUT HYSTERESIS
LINE(50,200)-(50,200)
FOR I = 0 TO 500
  IF X(I)>0 THEN Y=1 ELSE Y=-1
  OLDVOUT = VOUT
  LINE-(I+50,200-20*Y),14
NEXT
' DO COMPARATOR FUNCTION WITH HYSTERESIS
LINE(50,295)-(50,295),14
FOR I = 0 TO 500
  IF {X(I) > VHI} THEN
    IF (PREV = VLO) THEN
      PREV = VHI
      VOUT = 1
    ELSE
      IF PREV = 0 THEN
        VOUT = 1
        PREV = VHI
      END IF
    END IF
  ELSEIF (X(I) < VLO) THEN
    IF (PREV = VHI) THEN
      PREV = VLO
      VOUT = -1
    ELSE
      IF PREV = 0 THEN
        VOUT = -1
        PREV = VLO
      END IF
    END IF
  END IF
  LINE-(I+50,275-20*VOUT),14
NEXT
100 A$=INKEY$:IF A$="" THEN 100
STOP
```

Listing 3—The comparator functions of DSP, both with and without hysteresis, are illustrated in HYSTER. BAS. This is another illustration of a DSP's ability to perform functions normally relegated to the analog electronic world.

## 68000 Designers ARE YOU FED UP?



- Simulation too slow?
- ICE too costly?
- Target hardware not ready?

**GO FOR HIGH SPEED DEVELOPMENT... Put the MISTER-8™ in your PC!**

The MISTER-8™ is a complete 68000 development system that can help you get your software out quickly.

- Write your 68000 program on any PC/AT or Clone.
- Cross assemble with the included support software.
- Download at high speed through the I/O channel.
- Execute your program...inside the PC.

### FEATURES

- Low cost. ACIA allows two-terminal debugging of complex code. Use two or more boards for multiprocessing (software support available). Uses 8 MHz 68008, 64K SRAM; three 28-pin sockets for up to 128K of your EPROM code.

MISTER-8™ with cross assembler ..... \$495  
MR8-DB Debug EPROM ..... \$75

**MICRO RESOURCES**  
Making Technology Work

50 SOUTH EIGHTH STREET, LEWISBURG, PA 17837

717-524-7390 or  
717-523-0777

## BCC52 BASIC-52 COMPUTER/CONTROLLER

The BCC52 Computer/Controller is Micromint's hottest selling stand-alone single-board micro-computer. 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 new 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, 3 parallel ports, a serial terminal port with auto baud rate selection, a serial printer port, and it is bus compatible with the full line of BCC-bus expansion boards. The BCC52 bridges the gap between expensive programmable controllers and hard-to-justify price-sensitive control applications.

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.

Since the BASIC-52 is bus oriented, it supports the following Micromint expansion boards in any of Micromint's card cages with optional power supplies:

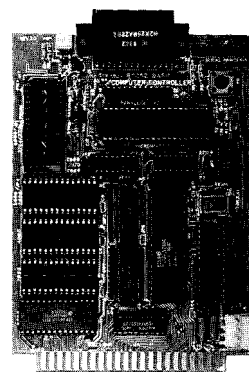
BCC22 Smart terminal board	BCC13 8-Channel 8-bit A/D converter
BCC25 LCD display board	BCC30 16-Channel 12-bit A/D converter
BCC33 3-port I/O expansion board	BCC18 Dual channel serial I/O board
BCC400 8-Channel optoisolated I/O expansion board	BCC55 Prototyping bad
BCC40R 8-Channel relay output board	BCC45 Stepper Motor bad
BCC53 Memory ad 6-port I/O exp. board	

BCC52 BASIC-52 Controller board \$199.00  
BCC-SYST.5 '52 PAK™ Starter System \$449.00

Includes: BCC52, ROM A&B UTIL, CC01, MB08, UPS10  
BCC52 OEM 100 Quantity Price \$149.00

BCC52C Lower power all-CMOS version \$199.00

Note: The BCC52 series is available in Industrial Temperature Range, fully tested, OEM 100 quantity price—\$189.00. Call for other OEM pricing.



**To Order Call**  
1-800-635-3355  
Tel: (203) 871-6170  
FAX: (203) 872-2204  
TELEX: 643331

Micromint, Inc. - 4 Park Street, Vernon, CT 06066

Reader Service #135

Reader Service #139

This translates to a maximum sampling rate of 74 Hz. With a 74-Hz sampling rate, the highest frequency that can be described (due to the Nyquist theorem) is only 37 Hz! If an 80287 numeric coprocessor chip is installed, the processing time is reduced to 1.6 ms, yielding a maximum sampling frequency of 625 Hz. Even with a coprocessor, the highest frequency that can be handled is only 312.5 Hz; not very good for real-time digital filtering of audio signals. We need a high-speed processor which possesses special instructions optimized for the sum-of-products operations required in digital filtering. General-purpose microprocessors like the 80186, 80286, 68000, and so on, lack this special feature. Custom DSP hardware designs which utilize multiplier-accumulator chips and bit-slice processors are difficult to design and support even though they provide the greatest performance. Most high-end, high-speed DSP is performed through the use of such circuitry.

An approach which bridges the

gap between general-purpose microprocessors and exotic bit-slice designs is the general-purpose Digital Signal Processor, first unveiled in 1983 by Texas Instruments (the TMS32010). Since 1983, many other manufacturers have introduced general-purpose DSP chips such as the Motorola 56001 (which is used by NeXT), Analog Devices' ADSP-2100, and AT&T's DSP16 and DSP32, to name a few.

DSP chip prices have undergone a dramatic deflation since 1983. For example, the TMS32010 (first-generation DSP chip) originally cost about \$500 in 1983. Now, the same chip costs around \$10! Future generations of DSP chips are expected to offer more performance at lower cost, ensuring future DSP usage to be widespread. One such chip which currently offers high performance at moderate cost is the second-generation TMS320C25, a 40-MHz CMOS digital signal processor. By utilizing this chip with its 100-ns instruction cycle, the 15-tap digital filter example can be performed in only 7  $\mu$ s!

In the next article, we will discuss the TMS320C25 along with examples of digital filters and other interesting applications.

The flexibility of programmable DSP coupled with the emergence of low-cost "microprocessor-like" digital signal processor chips ensures a "DSP Revolution" in the electronics world in the 90s just as the microprocessor revolutionized electronics in the 80s. ❖

**Dean** McConnell is a Senior *Software Engineer* for **Rockwell International, Richardson, Texas, where he is involved with development of real-time computer-controlled communication systems for the military.**

To receive advance information about DXP-25, call or write:

**IntelHome**  
571 Responsive Way  
McKinney, TX  
(214) 548-8503  
Fax: (214) 548-1521

IRS

2 10 Very Useful  
2 11 Moderately Useful  
2 12 Not Useful

## ID160/161



### Powerful PC-based Logic Analyzer for only \$595

\* 100MHz model for \$745

- 50 MHz Sampling Speed
- Multi-level triggering
- 8K trace buffer
- 32-channel capability
- Event Timer/Counter
- Performance Histograms
- Hardcopy output
- Disassembles 8-bit micros
- and much more!
- Satisfaction Guaranteed



**INNOTECH DESIGN, INC.**  
P.O. Box 3304  
Cerritos, CA 90703-3304  
Tel: 714-527-8540 FAX: 714-527-1812

Reader Service #126

## We got you covered...

... with complete embedded system support for popular PC compilers on Intel 80x 86 and NEC V-Series microprocessors. Now you can develop an embedded application with your choice of compiler—with full support for source level debuggers and in-circuit emulators.

Paradigm LOCATE supports popular C, Pascal, BASIC & Modula-2 compilers from Microsoft, Borland and others.

- comprehensive user's manual
- compiler startup code & examples
- free technical support
- math coprocessor emulation support
- extensible MS-DOS emulator
- full Intel OMF output

Lack of an emulator got you down? Not a problem with our Turbo Debugger interface option. Now you can debug your target hardware from the comfort of your PC.

**Paradigm LOCATE \$295.00**  
**Turbo Debugger Interface \$195.00**

Satisfaction Guaranteed  
Orders: (800) 537-5043  
Technical Support: (508) 478-0499  
BIX: join paradigm  
Visa/Mastercard/C.O.D. Accepted



**Paradigm Systems**  
P.O. Box 152 Milford, Massachusetts 01757  
Turbo Debugger is a registered trademark of Borland International

Reader Service #143

# The BCCH 16

## Porting a Multitasking BASIC to the H 76

Part 2  
Jack Ganssle

**T**om Cantrell introduced the H16 microprocessor in *CIRCUIT CELLAR INK* #11. His treatment covered the hardware details of the CPU and the specifics of this implementation of the BCCH16 single-board computer. Hardware is not complete without companion software; this article addresses the flipside of the coin: the software.

Those of you familiar with the 8-bit architectures will be pleasantly surprised by the flexible instruction set offered by the H16. Like the 68000 family, registers are essentially modeless—you can use any register in any instruction. A wide variety of addressing modes makes getting to data easy. Built-in instructions take care of tedious functions like multiplication and division. In short, programming the H16 is much like working on a large mainframe computer, with many of the same low-level resources (with the exception of hardware floating point).

The press is full of stories of the legal firefight between Motorola and Hitachi, but this battle revolves around the H8 processor and not the H16. It appears that the H16 is safe from litigation (or, at least as safe as any chip is in the Silicon Valley tort wars).

### REGISTERS

The H16 includes 1K of very fast RAM that comprises the processor's register set. As Tom's article pointed out, the RAM can be divided into memory and a set of 16 32-bit registers, or as 16 banks of registers, each bank including 16 registers. I have no intention of duplicating his particu-

larly lucid description of the various modes of this memory. I do think that for most purposes the RAM will be used completely for registers. The processor is entirely too powerful for much help from a little fast on-board RAM; most applications will use huge external RAM arrays.

At any time you have access to registers R0 to R15, but with one instruction you can change between 16 different sets of R0 to R15. Years ago, as a novice assembly programmer my credo was "More registers!" Now, after fixing hundreds of thousands of lines of old code, I think too many registers is more a curse than a blessing, since it is usually impossible to remember what is supposed to be in each. Two hundred fifty-six 32-bit registers is a lot! However, Hitachi's banked approach, where only one set is available at a time, is well suited for multitasking applications. Although no sane programmer would attempt to use all of them in in-line code, if you allocate one bank to each independent task (assuming no more than 16 tasks), then a context switch can occur in about a microsecond. More on this later.

So, for all practical purposes the H16 has 16 general purposes registers. Its orthogonal instruction set lets you use any register in any instruction; where in the HD64180, only the HL register pair can address memory, in the H16 any register can. The artificial restrictions imposed by the limited number of HD64180 opcode bits are nonexistent in the H16.

As Tom mentioned, the H16 supports 24-bit memory addresses. While 32-bit registers make handling large

integers much easier, they are essential for providing convenient access to this large memory space. Any register can contain all or part of an argument address; you never have to worry about not being able to fit an address into a register.

### ADDRESSING MODES

A large part of the power of the H16 derives from its wide variety of addressing modes. Any computer lets you load a register from a specific address; the more sophisticated permit indirect loads (e.g., load from the address in register 10) or even double indirect (e.g., load from the address in the one pointed to by register 10). The H16 supports almost every conceivable addressing scheme. Table 1 summarizes each of the legal addressing modes.

Some of the modes are identical to those you'd find in any computer. For example, the Register Direct mode lets you load from or to any register. Immediate is just like immediate mode on the HD64180; it lets you load a register (or memory address) with data that is included with the instruction (like `LD HL, 12 3 4`). The register indirect mode is used to access a memory location whose address is in a register, somewhat like the HD64180's `LD A, (HL)`, except that any register can hold the memory address.

Other addressing modes parallel those found in the 68000 family. Autoincrement and autodecrement are two of the most useful. The instruction `MOV @R1+, R2` loads R2 with the data in the memory address pointed to by R1. R1 is then incre-

mented. The converse, `MOV @ -R1 , R2` decrements `R1` before the move. In handling arrays, lists, queues, or virtually any data structure, these formats are invaluable. As we'll see later, the H16 supports 8-, 16-, and 32-bit transfers; these autoincrement and autodecrement addressing modes will adjust the register contents by 1, 2, or 4 as needed, so regardless of data type after the instruction, the register will point to the correct next entry in your table. Automatically.

The Register Indirect with Index and Register Indirect with Scale access memory using a number of parameters. As an example, `MOV @ (table, R1*4, R2) , R14` loads `R14` from an entry in an array called "table." The address in `R1` is multiplied by the scale factor 4 (letting you easily get to variably sized data), and is then added to the beginning address of the array ("table") and the contents of register `R2`. Wow!

As with most modem computers, several Program Counter relative addressing modes are included. These are formatted much like the other addressing modes. By making the PC one part of the address, it is possible to write reentrant code--code whose location can be changed without reassembling the source. This is crucial in complex applications where programs or overlays are dynamically loaded in different sections of memory. The same program might be loaded at several different locations, so must be address independent. While the HD64180 supports relative addresses with short jumps, in the H16 it is possible to code all program transfers and data moves as relative.

One of the more interesting modes is Register Double Indirect. While Register Indirect accesses a location pointed to by a register, Double Indirect uses the contents of a register as a pointer to a memory-resident pointer

to the final address. You can thus easily work with tables of pointers without cumbersome intermediate loads. Even better, several displacements can be combined so the addressing register can be independent of the table's base address. A construct of the form `MOV @ (table1, @ (table2, R2) ) , R3` is perfectly legal and rather mind boggling. `R2` is added to `table2` to form a pointer to memory. This memory pointer is loaded to an internal (nonvisible) register and is then added to `table1`, finally yielding the address of the argument. I guess you might use this if you are referencing a table of pointers indirectly through another table of pointers.

Unlike 8- and many 16-bit microcomputers, you are free to use any addressing mode with any instruction. Even better, instructions that require two arguments let you use any mode for any argument. Memory-to-memory moves, without intermediate register loads, are quick and easy. Complex constructs like `MOV @ (table1, @ (table2, R2) ) , @r12+` are perfectly legal.

This flexibility does not come free. More complex addressing modes require longer opcodes. Obviously, a memory-to-memory move with the two addresses specified as 32-bit absolute addresses requires an B-byte opcode just to store the addresses of the two arguments; additional byte(s) are needed to specify the instruction. Further, these lengthy instructions tend to be slow, since the processor has many bytes to read before beginning the operation.

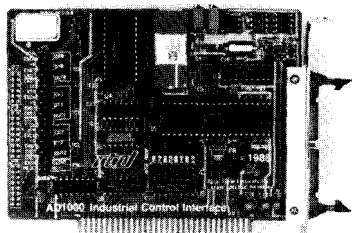
Smart programming can drastically speed these operations and reduce code sizes. In fact, specifying a 32-bit address is not something that has to be done frequently. An index register can be used to indicate the start of a data area. Or, program-counter-relative addressing might be used, especially if the data is relatively close to the code (say, within 64K). Certainly all array and table access should be made indirectly through a register, so only the initial load of the register will require a long instruction.

Addressing Mode	Mnemonic	Description
Register Direct	Rn	register contents
Register Indirect	@Rn @(disp,Rn)	register points to argument in memory displacement+register points to argument
Register Indirect with autoincr.	@Rn+	Get argument from memory pointed to by register; then increment Rn
Register Indirect with autodecr.	@-Rn	Decrement Rn; then get argument from memory pointed to by Rn
Immediate	#number	Argument is the number
Absolute address	@number	Argument is at address <number>
Register Indirect with scale	@Rn*Sf @(disp,Rn*Sf)	Register times Sf points to memory Register*Sf+displacement points to mem Note that Sf can be 1, 2, 4, or 8
Register Indirect with Index	@(disp,Xm*Sf,Rn)	Register Xm*Sf + Rn + disp points to memory
Program Counter Relative with index	@(disp,Xm*Sf,PC)	Register Xm*Sf + disp + PC points to memory
Program Counter Relative	@(disp,PC)	Program counter + disp points to memory
Register double indirect	@@Rn @(disp2,@(disp1,Rn))	Register Rn points to address of argument in memory Rn + disp 1 points to memory value; this value is then added to disp2 and the result points to the final argument

Table 1-11 The H16 supports a large number of addressing modes. Any mode may be used with any instruction, and for instructions that use two arguments, any addressing mode may be used for any argument.

# PC Bus Data Acquisition and Control Cards

STOP paying a small fortune for  
PC data acquisition and control  
interfaces!



## AD1000 — \$295

Real Time Devices, Inc. designs and manufactures the lowest cost industrial/scientific interface cards for the PC/XT/AT bus. Our commitment is to offer only high quality U.S. designed and manufactured interfaces—not cheap imports. All our cards are backed by a one year warranty, 30 day NO RISK return policy, and free technical support!

**AD1000** — 8 channel 12-bit 20 uS A/D; sample & hold; three 5-MHz timer/counter; 24 TTL digital I/O lines. \$295

**AD200** — 4 channel 12-bit 125 uS A/D; three 5-MHz timer/counter; resistor configurable gains; 24 digital I/O lines. \$259

**AD500** — 8 channel 12-bit integrating A/D; programmable gain of 1, 10, & 100. Extremely stable & accurate \$239

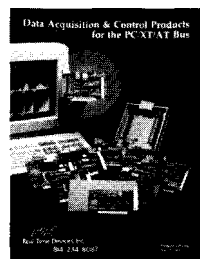
**AD100A** — Single channel, differential input, 12-bit integrating A/D; 8-bit D/A output; gains of 1, 10, & 100. Plus 10 digital I/O lines. \$215

**ADA300** — 8 channel 8-bit 25 uS A/D; single 8-bit D/A; 24 TTL digital I/O lines. \$239

**DA600** — Fast settling dual/quad 12-bit D/A; internal double buffering. \$207/\$298

**DG24/96** — 24/48/72/96 line TTL compatible digital I/O cards; 10 MHz. 8255-based. Optional buffers and pull-ups. \$95/1274

**TC24** — Five 5 MHz timer/counters; uses powerful AM9513 chip. Also 24 digital lines. \$218



## FREE CATALOG

CALL or WRITE today for your copy of the latest catalog and price list for our complete line of PC bus data acquisition, control, prototyping, extender cards, technical books and accessories!

Custom/OEM designs on request!

Real Time Devices, Inc.

**rtid** 820 N. University  
P.O. Box 906  
State College, PA 16664

814/234-8087

Reader Service #151

The H16 is thus a harbinger of one programming philosophy of the 90s: while complex CPUs facilitate easier coding, they demand more expertise and more careful analysis of the problem. Thoughtless use of HD64180 programming conventions will cripple the H16's performance; only by taking advantage of the inherent structure implicit in any programming problems will its potential be fully realized.

## INSTRUCTION SET

Early microprocessors and mini-computers relied on the "single accumulator" instruction format, where most memory transactions and all arithmetic occurred through a single register (A in the HD64180). In the H16, the instruction set is orthogonal; every register can address memory, and each can perform math and logic operations. Where in the HD64180 careful attention is paid to how registers are loaded, since math can only take place on A, in the H16 you can even do math directly to memory.

All of the processor's registers are 32 bits long, suggesting that 32-bit math and logic operations are the norm. This would be quite awkward when trying to manipulate a character string! Nearly every H16 instruction takes several forms, so you can specify the length of the operation. Want to move a byte of memory? Issue `MOV . B R2, R1`. A word (16 bits)? `MOV . W R2, R1`. Of course, a long (32-bit) operation is possible, using the form `MOV . L R2, R1`. (Note that the mnemonics use an argument format reversed from the HD64180. The first argument is the source, the second is the destination.)

This word length option is crucial to any real application, but I feel it does make the code a bit hard to read. The 68000 suffers from this problem as well.

Hitachi publishes several well-written and comprehensive H16 data books, so I'll make no attempt to detail all of the instructions here. Rather, I'll just describe some of those more commonly used.

The move (MOV) instruction has already been informally introduced. In its various forms it transfers data between registers, memory, and I/O with few restrictions.

## ARITHMETIC AND LOGICAL INSTRUCTIONS

As mentioned, any instruction can operate on registers or memory. Therefore, ADD can work between registers or memory, as shown in Listing 1.

The H16 includes all normal math instructions, including SUB, AND, ADD with carry, decimal add, negate, NOT, logical inclusive and exclusive OR, and compare. Instructions are provided to perform sign extensions and similar functions. All of these are pretty traditional and hold few surprises.

Like other high-performance processors, the H16 has built-in integer multiply and divide routines. Both signed and unsigned versions of these instructions are provided.

One of the weaknesses of the HD64180 is its limited multiply instruction. The H16 includes both an 8-by-8-bit multiply (with a 16-bit result), and a 16-by-16-bit version yielding 32 bits. Normal integer math is therefore trivial to implement.

The divides are similar. Two 16-bit numbers can be divided, giving an 8-bit quotient and an 8-bit remainder. You can also divide a 32-bit number by one of 16 bits, generating a 16-bit quotient and a 16-bit remainder.

## BIT INSTRUCTIONS

A number of intriguing bit manipulation instructions are included.

```
ADD    R1,R2      ; R2=R1+R2
ADD    @R10,data  ; address "data" = data + what is
                          ; pointed to by R10
ADD.B  mem1, mem2 ; 8-bit add of two memory locations
```

listing 1 -Any H16 instruction can operate on either registers or memory.



A bit field extract (BFEXT) operation lets you remove any number of bits, starting at any bit position, in a long word. The resulting bits are then right-justified in the result and zero-filled to the left. Tedious rotates and ANDs can be replaced with a single instruction. Its complement is bit field insert (BFINS). In addition, bit test and set instructions allow you to emulate the similar HD64180-like functions.

Six types of shift operations are provided. The rotate instructions let you revolve bits around a register to the left or right, either through the carry or not. Arithmetic shifts shift a register left or right, without changing the sign of the value (i.e., a negative value will stay negative since the sign bit will be propagated during the shift). Logical shifts are similar, but are bitwise, and do not preserve the argument's sign. With all of these instructions you provide an argument that specifies the number of bit positions to shift; the days of multiple single-bit rotate instructions are happily long gone.

## PROGRAM TRANSFER INSTRUCTIONS

The H16 offers a number of program transfer instructions. Both unconditional branches and jumps are included; the branch instruction is always PC relative. JMP can take any addressing mode, so you can even do a double indirect through memory pointers.

Similarly, the branch to subroutine (BSR) instruction is program counter relative while jump to subroutine (JSR) can use any addressing mode. The subroutine invocation instructions all save the return address on the stack, inviting the use of the RTS (return from subroutine) instruction.

Like many computers, the H16 maintains a condition code register that tracks the results of the previous instruction. While most nonarithmetic instructions don't affect the condition codes, the move instructions do, signaling an important departure from the HD64180 family. Beware when converting code!

Conditional branches are crucial to loop control and decision making. The H16 has a good selection of these, covering conditions such as carry set or clear, zero or nonzero, greater than, less than, minus or plus, or overflow/nonoverflow. All branch instructions are program counter relative, promoting the use of reentrant code. Unlike most 8-bit CPUs, whose relative conditional jumps are usually limited to fairly short jumps (typically  $\pm 127$  bytes), the H16 supports both short and word (16-bit displacement) jumps.


## MISCELLANEOUS INSTRUCTIONS

One powerful instruction pair is LDM (load multiple registers) and STM (store multiple registers). These take a 16-bit bitfield indicating which registers to save or restore, and a memory array destination address. They could be used for convenient context switching (although I prefer switching register banks if no more than 16 processes are competing). Other applications

**New, Improved!**

# SIBEC-II


*The ideal solution for embedded control applications and stand-alone development.*



- Intel 8052AH BASIC CPU
- Serial printer output and 5, 8 bit I/O ports
- 5 in.<sup>2</sup> prototyping area
- Memory: 8K RAM, expandable to 128K
- Power requirements: 5V.DC @ 300 ma. only
- PROM programmer; ZIF socket for 2764 or 27128 EPROM
- Interrupt handling capability
- Built to exacting standards and warranted
- Still only \$228.00 including documentation (quantity 1)

Inquire about our PDK51 8051-8052 product development kit for the IBM-PC/XT/AT: \$595.  
Our BXC51 8051/8052 BASIC compiler: \$295.

**Call now! 603-469-3232**

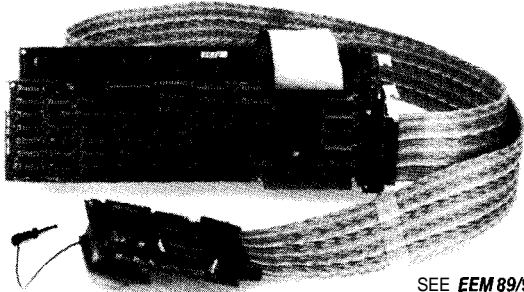


Binary Technology, Inc.  
Main Street • PO Box 67 • Meriden, NH 03770

Reader Service #108

# 68HC11

## PC-based emulator for 68HC11



SEE EEM 89/90  
Pages D 1324-1326

- PC plug-in or RS-232 box.
- Pull-down menus with full window support, combined with command driven User Interface.
- Up to 3.3 MHz (E clock) real time emulation.
- No intrusions to the 68HC11's resources.
- 48 bit wide 16K deep trace. All functions usable without disturbing emulation. Time stamping. Two level trigger.
- Symbolic and C Source Level Debugging, including in-line assembler and disassembler.

Prices: 64K Emulator and pod \$2590: 4K Trace \$1995\*

**CALL OR WRITE FOR FREE DEMO D/SK!**

# NOHAU

CORPORATION

51 E. Campbell Avenue  
Campbell, CA 95008  
FAX (408) 376-7869  
(408) 866-1820

\*US only

Reader Service #141

include debuggers, short interrupt service routines, and the like.

"String" instructions are provided to support automatic looping. They permit block moves (like the **LDIR** instruction in the HD64180) and comparisons.

**LINK** and **UNLK** (unlink) automatically create stack frames, as used by many compilers and some assembly programs. **LINK** pushes a register on the stack, saves the SP to the register, and then adds an immediate value to the SP. **UNLK** reverses this process. These instructions provide a convenient way of passing parameters on the stack—a common activity in C. HD64180 C compilers really suffer from the terrible tediousness of working off the stack, but the H16's orthogonal instruction set makes stack manipulations as easy as register operations.

As I mentioned, this is by no means a complete list of instructions, but it does illustrate the nature of some of the most common instructions.

## SUPERVISOR/USER MODE

To support complex multiuser applications, the H16 can operate in two different modes. On reset, it defaults to supervisor mode, giving the programmer all of the resources available to the CPU.

The CPU enters User mode by changing one bit in the system status register. Many instructions will no longer be available; trying to **execute a HALT**, for example, will force an exception to take place. Certainly, in a multiuser **system** you don't want users executing **HALT** instructions! Many of the system control registers (for example, those that define the operation of the internal 1K RAM) are not accessible in user mode. Further, the internal I/O devices cannot be accessed; the chip assumes all access to I/O is made through calls to an operating system executing in supervisor mode.

A **TRAP** instruction lets the user-mode code invoke the supervisor

program. A simple mechanism lets the user program pass an argument to the supervisor.

## SPEED

The question always asked about a new processor is "just how fast is this baby, anyway?" Modem CPUs often defy speed measurements, due to their complex operation.

The H16 includes an 8-stage **prefetcher**; an independent bus interface unit can read and store up to 8 bytes before they are needed by the execution unit. These bytes are fetched linearly; the bus unit assumes the next byte of the instruction stream will follow the previous one. An interrupt or program branch invalidates this algorithm, and the prefetcher must be flushed and then restarted.

Thus, the prefetcher makes it all but impossible to determine the time is needed to get an instruction from memory. If it is already prefetched, then the time is zero; if it must be read,

# EXPRESS CIRCUITS

MANUFACTURERS OF PROTOTYPE PRINTED CIRCUITS FROM YOUR **CAD** DESIGNS  
TURN AROUND TIMES AVAILABLE FROM 24 HRS — 2 WEEKS

### Special Support For:

- TANGO.PCB
- TANGO SERIES II
- TANGO PLUS
- PROTEL AUTOTRAX
- PROTEL EASYTRAX
- smARTWORK
- HiWIRE-Plus
- EE DESIGNER I
- EE DESIGNER III
- PADS - PCB
- OTHER PACKAGES ARE NOW BEING ADDED
- FULL TIME MODEM
- GERBER PHOTO PLOTTING

*Express*  
Circuits

314 Cothren St., PO. Box 58  
Wilkesboro, NC 28697

Quotes:  
1-800426-5396  
Phone: (919) 667-2100  
Fax: (919) 667-0487

then the time is a function of the clock rate, DMA activity, and wait states.

Further complicating matters is the wide number of **addressing** modes. Any instruction can take literally hundreds of forms, depending on the source and destination arguments.

The Hitachi data book includes 13 pages describing timing. Some instructions (like a NOP) can take only 100 ns at 10 MHz, while others can take 50–60  $\mu$ s (a complicated divide, for example). It's all but impossible to construct a simple timing chart. Even worse, comparing the execution speed of the H16 to another processor is very difficult without running extensive benchmarks on both. Fact: the HD64180 executes most instructions faster than the H16. Fact: the H16's instructions do considerably more than the HD64180's. Fact: the H16 can handle 32 bits in one instruction, which might take quite a few instructions to simulate on an HD64180.

## CODE COMPARISONS

It is often instructive to compare code written for two different processors to see how each CPU's architecture can be best taken advantage of. Since the BCCH16 is the successor to the BCC180 computer, it makes sense to compare the H16 to the familiar HD64180.

Suppose you'd like to add two 32-bit numbers, as is done in Listing 2. If we were to store the result in num2, the H16 code would be:

```
ADD.L @num1,@num2
```

This is almost an unfair comparison! It really shows the power of having 32-bit registers. Obviously, multibyte math operations are much simpler with the H16.

Perhaps you have a table of pointers to data, and wish to compare a value in a register (say BC on the HD64180 or R2 on the H16) to that to which the pointers point. The code might look like Listing 3 if a 16-bit compare were done (ignoring the case where the search fails). The double indirect mode really pays off here. If a 32-bit compare were done, then the

```
HD64180:
LD      HL, (num1)      ; get low 16 bits of numbers
LD      DE, (num2)
ADD     HL,DE
LD      (result),HL    ; save low part of answer
LD      HL, (num1+2)   ; get high 16 bits
LD      DE, (num2+2)
JR      NC, SKP        ; skip if no carry from low add
INC     HL              ; propagate carry
SKP:    ADD     HL,DE
LD      (result+2),hl  ; save high part of result

H16:
MOV.L   @num1,R1
ADD.L   @num2,R1
MOV.L   R1,@result
```

listing 2—Adding two 32-bit numbers is easy with the H16 because of its internal 32-bit registers.

```
HD64180:
LP:     LD      HL, table ; BC has comparison value
LD      E, (HL)
INC     HL
LD      D, (HL)          ; DE points to comparison data
INC     HL
LD      A, (DE)          ; get low part of data
CP      A, C              ; low parts match?
JR      NZ, LP
INC     DE
LD      A, (DE)
CP      A, B              ; high parts match?
JR      NZ, LP

H16:
MOV     #table-2,R1      ; point to table of pointers
LP:     ADD     #2,R1      ; point to next entry
CMP.W   @R1,R2           ; value the same?
BNE     LP                ; loop till found
```

listing 3—Pointers are also easy to work with on the H16 thanks to its double indirect addressing mode.

H16 code would be no longer; the CMP instruction would be CMP .L rather than CMP .W. On the HD64180, quite a few extra instructions would be needed.

If you're processing 8-bit data in a limited address space (like 64K), then the HD64180 processor may be a bit less convenient to use than the H16, but is still a pretty good choice. The H16 really shines at handling large data items in big address ranges.

## SUPPORT PRODUCTS

No CPU is an island, entire to itself. Without decent support hardware and software, even a 500-GFLOPS processor is worthless.

The H16 is a relatively recent introduction, so its support base is currently somewhat limited. The BCCH16 single-board computer is the most obvious support product for the device. The BCCH16, as Tom Cantrell

described, is a complete H16 computer on a single card. Just add a terminal and power supply for operation. It comes complete with a BASIC in ROM, so code can be generated seconds after applying power. The BCCH16 is ideal for inclusion in controllers and other products, since its price is quite reasonable. It is also a great platform for evaluating the chip for designers considering including the CPU in their proprietary embedded systems.

Hitachi does sell a cross-assembler that runs on an IBM PC. It generates Motorola S records, which can be burned into ROM or downloaded into an emulator. The package includes a quite sophisticated macro assembler that was originally ported from a VAX. No, it's not the fastest compiler in the world, but on my 386SX, assembling 2000+ line modules takes less than a minute. The assembler has an extensive selection of pseudo ops. The

. SECTION pseudo op lets you partition your program into apparently hundreds of individual sections, classed into up to five types (code, data, common, stack, and dummy).

The bundled linker seems to handle even very large programs without much trouble. Like the assembler the linker is no speed demon but is very robust.

For some reason, the odd nature of H16 assembly language (with typically thousands of periods in a module) crashes both WordStar and Microsoft Works, my two primary editors. Recently, I started using Qedit

from SemWare, which is immune from these problems. (At \$55, Qedit is a fantastic editor; it is WordStar compatible, very reliable, and tiny—a must for use on laptops).

While developing the code for H16 BASIC for the H16, I used the 64700 In-Circuit Emulator from HP. To my knowledge, there are no really good debuggers yet available for the H16 (hint, hint—this might be a market opportunity for someone), so the emulator was essential. Unfortunately, the HP emulator cannot use the detailed debugging information produced by the Hitachi Linker. As a result, even

symbolic debugging was unavailable. Bill Auerbach, a local C whiz, wrote a PC-based emulator driver for me that annotates the disassembled code with symbols extracted from the Hitachi link file.

Software Environments sells an assembler, linker, and C compiler for the H16, but I have no experience with these products. They can be reached at (214) 991-0084.

Finally, Hitachi produced a software guide for the H16 that includes several hundred pages of sample code. A number of routines are included, such as sorts, ASCII conversions, and math routines. This is manual number ADE-502-008, and is essential as an introduction to writing code for the H16. The H16 data book (book number ADE-602-003) is the bible, but lacks examples. The software guide fills this gap.

#### INTRODUCING H16 BASIC

The BCCH16, without an embedded language, would appeal only to die-hard H16 fans. It seemed natural to port the BASIC-180 language used on the HD64180-based BCC180 to the H16.

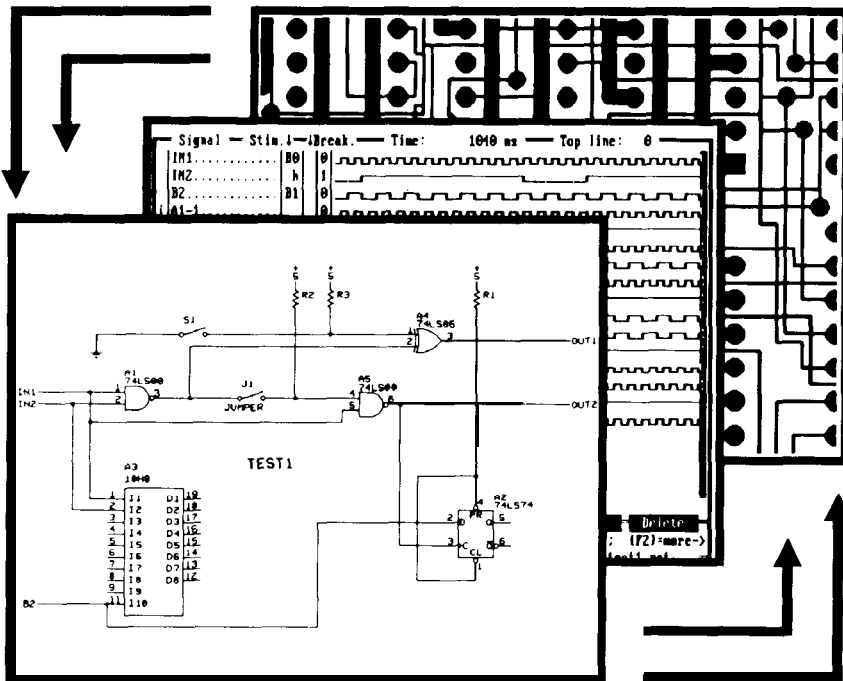
H16 BASIC resides in two ROMs on the board. As an embedded language, it is invoked by the reset signal; the BCCH16 by default, with the H16 BASIC ROM set, powers up in the BASIC environment.

H16 BASIC is an interactive language, so you can enter, run, and modify programs just as if you were working on an interpreter like Microsoft's MBASIC.

The BASIC is a true compiler. When you type RUN, it converts the tokenized BASIC code to native H16 machine language. You get the benefits of an interpreter (interactive development) with the speed of a true compiler.

Many of you are no doubt familiar with BASIC-180 from the BCC180 and Steve Ciarcia's articles in the February and March 1988 issues of BYTE, so I won't dwell on the language's features here. Of more interest is how H16 BASIC was configured to take advantage of the additional

## PROMOTE YOUR PC TO WORKSTATION STATUS™



A FULLY INTEGRATED SOLUTION FOR DESIGN OF PC BOARDS .  
AT A FRACTION OF THE COST OF A WORKSTATION.

SCHEMA II+ schematic capture	\$495
SCHEMA-PCB layout with ROUTE & PLOT	\$1495
SCHEMA-SUSIE logic simulator	\$995

FREE 1-800 customer support

FREE DEMO DISK

F\*111130 day money back guarantee

**1-800-553-9119**



**OMATION**

resources of the H16 processor, and of the BCCH16 computer board.

By now, it must be pretty obvious that the H16 excels at 32-bit integer operations. BASIC-180 on the HD64180 was designed to work with 16-bit integer values, since that length is better suited to 8-bit computers. On the H16, we elected to change this to 32 bits, giving considerably more range to integers while costing nothing in execution time—after all, the H16 can do a 32-bit add or subtract in a single instruction. There is a penalty to pay in memory, since all integers and other variables are saved in RAM.

Fortunately, the H16's huge linear address space makes it easy to sacrifice some RAM to extended variable storage. On the BCC180, we used an elaborate dynamic mapping scheme to take advantage of more than 64K of memory; every task resided in a different physical address but identical logical addresses. Due to the HD64180's memory management unit, no more than 32K of RAM could be made available to a program without using complicated remapping techniques.

The BCCH16 includes up to 1 MB of RAM, in addition to the ROMed

compiler in low memory. This is a lot of space! By completely recoding the compiler, all of the memory is seamlessly available to the user. While the HD64180 version of the language accessed memory dynamically through HL, limiting the logical space to 64K, on the H16 we use the 32-bit registers as memory pointers. Artificial 64K limits simply disappear; arrays can exceed 64K, programs can grow to huge sizes, and no complex remapping ever takes place.

H16BASIC's operation is split into three phases:

- Interactive editing, where the user enters the program
- Compilation, where the program is converted to machine code
- Runtime, where the compiled program is executed

In all phases, certain variables essential to the compiler are stored at the beginning of the RAM area. About 10K of RAM is used for these house-keeping variables.

The tokenized code is stored immediately after the RAM data area. Tokenized code is a compressed, easier-to-compile version of the ASCII BASIC statements entered by the user. Tokens can take virtually an unlimited amount of space, depending on the size of the program.

When compilation starts (i.e., the "RUN" statement is entered), H16 BASIC converts the tokens to H16 machine code in a single pass. A symbol table, used for the duration of the compilation, is built from the end of memory down. The compiled code is stored immediately after the tokens. It, too, can take an almost unlimited amount of space. Of course, for a simple program only a few hundred bytes might be generated. Addresses are assigned to the variables declared by the programmer, working from the top of memory on down, on top of the symbol table. The symbol table exists only during compilation, and variables only during runtime, so they can share the same space.

Finally, at runtime the compiled code is executed where it was com-

## GET TO WORK!

### A New Project

Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files.

### Get It To Market--FAST

Don't wait until the hardware is finished to debug your software. Our Simulators can test your program logic before the hardware is built.

### No Source!

A minor glitch has shown up in the firmware, and you can't find the original source program. Our line of disassemblers can help you re-create the original assembly language source.

### Set To Go

Buy our developer package and the next time your boss says "get to work", you'll be ready for anything.

### Quality Solutions

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

### BROAD RANGE OF SUPPORT

- Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	

- All products require an IBM PC or compatible.

**Cross-Assemblers** as low as \$50.00

**Simulators** as low as \$100.00

**Cross-Disassemblers** low as \$100.00

**Developer Packages** low as \$200.00

(a \$50.00 Savings)

**So What Are You Waiting For?**

Call us

**PseudoCorp**

**Professional Development Products Group**

716 Thimble Shoals Blvd, Suite E

Newport News, VA 23606

**(804) 873-1947**

piled, using the variables defined from the end of memory. A number of standard routines (such as support for formatted printing) will be called by the compiled code; these were copied from the compiler's ROM to the RAM area just ahead of the compiled code during compilation.

The linear address space does make memory management refreshingly simple.

## MULTITASKING

One of the primary reasons for H16 BASIC is its support of multitasking. Almost any real control application requires the handling of multiple asynchronous events. While polled I/O is one (computationally expensive) way of responding to external I/O, multitasking is much more effective.

H16 BASIC supports both timer interrupts and device interrupts. Device interrupts are those produced by serial channels, DMA completion events, and the like. These are the most common sort of interrupts in the process-control environment. In effect, the **program** never really **needs** to wait for I/O. When the event occurs, a hardware interrupt signals the software so it can take the appropriate action.

Suppose a serial channel receives a character. It asserts an interrupt signal to the CPU, which immediately suspends whatever activity it was working on. An interrupt service routine is automatically invoked by the CPU to read that character and store it in a buffer.

H16 BASIC lets you couple a BASIC routine to any of these device interrupts. You could tell the BASIC to start line 200 if your external optical encoder passes the **rezero** mark. The BASIC might then reset a counter variable to zero and return.

Another use of interrupts is to sequence activities between multiple processes. Especially in the embedded world, it is usually easy and desirable to segment a program into a number of modules that execute concurrently and independently. For

example, a keyboard scan routine might run all the time, passing keystrokes to the "main" routine (task) through variables. Another routine might refresh the displays. Each can run independently.

H16 BASIC uses the CPU's timer to generate 60 interrupts per second. Whenever one of these interrupts occurs, the compiler switches execution to another BASIC task. Although a task priority scheme is implemented (so you can make one task run more often than another), generally most run at equal priority. Each task thus gets an equal share of computer time, and an equal chance to run.

Obviously, if H16 BASIC takes a long time to switch between tasks, then an awful lot of the computer's resources will be wasted just with the multitasking housekeeping. Fast "context switching" is thus crucial to fast operation.

BASIC-180 needs about 800  $\mu$ s to switch tasks. A big part of the problem is to completely save the state of the CPU, all of the registers, flags, and other CPU parameters must be preserved, and then those for the new task just being invoked restored. On the H16 this can be done in one instruction!

H16 BASIC uses the H16's internal RAM as 16 banks of registers. Each task is then assigned one of these register sets. To save and restore all 16 registers, the compiler simply issues a "change register bank" instruction, completing this tedious operation in about a microsecond.

Of course, switching tasks is more complicated than that. H16 BASIC must decide which task is the next eligible one to run; this is a function of which one is the highest priority, and which has not run for the longest time. As a result, task switching takes somewhere around 200 microseconds—still a significant improvement over the BCC180.

The BCC180 version of the BASIC supports 32 tasks, while the BCCH16 handles only 16. This, the only trade-off made in the conversion, was deemed wise because of the resulting efficiency of context switching.

## THE CONVERSION PROCESS

What does it take to convert 20,000 lines of HD64180 assembly language to a totally new processor? Initially, we hoped to write a translation utility to automatically convert most of the code. This proved impractical, since the code was completely structured around the limited addressing modes of the HD64180. While it is certainly possible to directly convert HD64180 code to H16 code, the resulting software would be slow, horrible to maintain, and a general embarrassment. As a result, we recoded, by hand, every bit of the language.

Fewer lines of code resulted, since the H16's instructions are so much more powerful. The number of bytes of code increased; the H16's instructions are powerful, and therefore long. The code runs considerably faster and the resulting shorter routines are easier to read and work on.

## SUMMARY

The H16 processor is a significant step ahead in embedded processors. Compared to the HD64180, it offers more performance, a better instruction set, and nice wide registers. The linear address space makes working on large programs much easier, and the code runs faster as well.

The penalty? A 16-bit CPU is always more expensive than one of 8 bits, and so will never supplant the smaller bus widths for many applications. Even if the CPU were free, two ROMs and two RAMs are used even in the simplest application. If you need performance, the H16 is a good choice. ♦

---

The BCCH16 board is available from MICROMINT Inc., 4 Park St., Vernon, CT 06066, (203) 671-6170, Fax: (203) 872-2204. Call for price and delivery.

---

*Jack Ganssle is president of Softaid, a vendor of microprocessor development tools. When not busy pushing electrons around, he sails up and down the East Coast on his 35-foot sloop.*

## IRS

213 Very Useful  
214 Moderately Useful  
215 Not Useful

# An Exercise for the Student

## *Building Software from the Ground Up*

### FIRMWARE FURNACE

Ed Nisley

**H**ave you ever noticed that you learn the Really Interesting Stuff by experience? You may spend weeks to discover something you can summarize in one sentence. But, had you seen that sentence when you started the project, you wouldn't have known what it meant; true knowledge does not come without effort.

Several BBS discussions lately touched on the practical aspects of compiling **programs**, rather than about the program code itself. I will start by reviewing the essentials of **multimodule** programs, then describe how the **MAKE** utility can simplify program creation. Finally, I will look at compilation speed and how you can improve it... under both **DOS** and **OS/2**.

#### THE BASICS

Your first PC program undoubtedly fit into one disk file. Compiling and linking were a snap and you couldn't imagine any other way of doing it. But then you took on a few-thousand-line project, only to discover that one file makes for painfully slow changes. There must be a better way!

Obviously, you must break your source code into separate files so that a change affects only a single file. You recompile that file, run the linker, and get back on the air. The linker is generally faster than the compiler (unless you have lots of files!), so the edit-compile-link-test cycle becomes fairly rapid.

Each of the source files should contain functions that are related to each other, which reduces the number of references between files. I have several standard files that always

contain the same type of routines: **STARTUP**, **SCREEN**, **PRINTER**, **SERIAL**, and so forth. In some **cases**, you can reuse code from **one** project on the next, although microcontroller firmware is less reusable.

But working with many source files means you must coordinate the code. Most programs have a collection of constants to determine sizes, addresses, and so forth; obviously, every source file must use the same constants! Similarly, code in one file will refer to routines or variables in another, so you must make some symbols **PUBLIC** in one file and **EXTERN** elsewhere.

For assembler projects, I collect all of the constants into a single file called **DEFINES.INC**. Every source file starts with an **INCLUDE** statement to copy text from **DEFINES**. This ensures that there is one definition of each constant and simplifies the inevitable changes. Listing 1 shows how this works.

**DEFINES** also holds structure definitions, since often the structure is declared in one file and used in several others. For example, **STORAGE.ASM** declares a variable called **Packet** and defines it **PUBLIC**, while **SETUP.ASM** defines **Packet** as an **EXTERN** symbol and uses its address. The linker connects the reference in **SETUP** to the declaration in **STORAGE**.

Another **INCLUDE** file named **MACROS.INC** holds assembler macros. I've collected a bunch of **AVMAC51** macros to simulate "missing" 8051 instructions: **CJE**, **DEC DPTR**, **LJZ**, and so on. Another **INCLUDE** drags these definitions into every source file. **MACROS** also holds special-purpose macros for each project.

**C** projects, on the other hand, have a separate header file for each source file. The headers contain constants relating to the code in the corresponding source file, as well as function prototypes for any **PUBLIC** routines. Any file that needs the routines or constants simply **#includes** the appropriate header file.

The C language specifies that variables declared outside of a function and all functions are public (the C term is "global") by default, so there is no need for a special keyword. If you want a variable accessible only within a file, add the "static" keyword to the declaration (obvious, right?). To reference the variable in another file, you must use the "extern" keyword on the declaration.

It is worth mentioning that C has a bewildering variety of options for making variables public, external, internal, global, local, static, or automatic. Spend some time reading your manual, because an innocent-looking declaration can produce obscure and baffling errors.

When a header file is included in several source files, each variable is declared extern in every file. One (and only one!) file must have an additional declaration without the **extern** to tell the linker where to reserve storage. Listing 2 shows how this works; you must ensure that the header file and C source file agree on each variable.

You can use the C preprocessor to change the header file on the fly to define or declare the variable depending on which source file is **#including** the header. Listing 3 shows how this works; the advantage is that the

initialization information is in the header rather than the source code file. There is, of course, a corresponding disadvantage which I'll describe later.

## MAKE-ING A DIFFERENCE

Once you have distributed your code among several source files, you are faced with the fact that the simple batch file that compiled a single file is no longer adequate. A batch file will recompile all of the source modules every time you run it, so there is no advantage to source in multiple files!

A MAKE utility provides the intelligence to recompile only changed source files. Using MAKE, you never have to recompile "everything, just to make sure" that you didn't forget a file. Even better, when you change a header file, MAKE will recompile all of the files that include the new file. I defy you to get that sort of relation right by hand more than half the time.

Vanilla MAKE programs come bundled with nearly every compiler, third-party vendors sell fancy MAKES, and you can find public-domain or shareware MAKES on your favorite BBS. I have been using Polytron's PolyMake for several years, so that's what I'll use in the examples. PolyMake has several features that don't show up in other MAKES, but the basics are the same regardless of which one you use.

All MAKE utilities depend on file extensions to identify the file content. You tell MAKE that .ASM files produce .OBJ files and .OBJ files turn into .HEX files, along with the exact assembler and linker command lines needed for the conversions. If some .ASM files depend on .INC files, you must also give MAKE a list of these dependencies.

Once MAKE knows all this, it can determine which source files are required to rebuild each .HEX file. Unlike a batch file, however, MAKE examines the file timestamps for each pair of "input" and "output" files and recompiles only those source files with a later timestamp. For example, if SERIAL.ASM was changed at 12:30 and SERIAL.OBJ was created 12:31,

### DEFINES.INC

```

MAXLEN EQU 100           ; maximum packet data length
BITRATE EQU 192          ; serial data rate / 100
BITCLOCK EQU 1152        ; serial bit clock / 100
COMPORT EQU 1            ; serial port: 1 or 2

PACKDEF STRUC            ; data packet structure definition
PLEN DB 0                ; packet data length
PTYPE DB 0               ; packet type
PDATA DB MAXLEN DUP(0); packet data bytes
PACKDEF ENDS
END

```

### STORAGE.ASM

```

INCLUDE DEFINES.INC

Packet PACKDEF <12,0,"Sample Packet"> ; data packet storage
PUBLIC Packet

BitDiv DW BITCLOCK/BITRATE ; serial port divisor
PUBLIC BitDiv

IF COMPORT EQ 1 ; serial port address
PortBase DW 3F8h
ELSE
PortBase DW 2F8h
ENDIF
END

```

### SETUP.ASM

```

MODEL LARGE
INCLUDE DEFINES.INC
DATA
EXTRN BitDiv:WORD
EXTRN Packet:BYTE
EXTRN PortBase:WORD

CODE
MOV DX,PortBase ; pick up port address
MOV AX,BitDiv ; and clock divisor

MOV SI,OFFSET Packet ; point to packet
MOV CL,[SI].PLEN ; pick up length
END

```

Listing 1—The DEFINES.INC file contains constants used throughout the other assembler files. This ensures that all files are using the same constants.

### NET.H

```

extern int OurNode; // our network node address

PACKET *NTGetHead(PQUEUE *pQueue) // get head packet from queue

```

### NET.C

```

#include "net.h"

int OurNode=31; // set default value

printf("OurNode initialized to %u\n",OurNode);

PACKET *NTGetHead(PQUEUE *pQueue) {
/* function body here! */
}

```

### SERIAL.C

```

#include "net.h"

printf("OurNode is %u\n",OurNode);

if (NULL == (pPacket = NTGetHead(&InQueue))){
/* and so forth */
}

```

Listing 2—C language programs generally have a header file for each source file. The header contains variable definitions and function prototypes. At least one source file must declare each variable without the 'extern' keyword.



```

NET.H
#undef EXT

#ifdef NET                // if NET is defined
#define EXT              // then vars are not external
#define INITIALIZE      // and they are initialized
#else                    // otherwise
#define EXT extern      // they are external
#endif INITIALIZE        // and not initialized
#endif

#undef NET

EXT int OurNode          // our network node address
#ifdef INITIALIZE        // initialize if called for
= 31
#endif
;
PACKET *NTGetHead(PQUEUE *pQueue); // get head packet from queue

```

```

NET.C
#define NET              // initialize variables
#include "net.h"

printf("OurNode initialized to %u\n",OurNode);

PACKET *NTGetHead(PQUEUE *pQueue) {
/* function body here! */
}

```

```

SERIAL-C
#include "net.h"        // variables are extern here

printf("OurNode is %u\n",OurNode);

if (NULL == (pPacket = NTGetHead(&InQueue))) {
/* and so forth */
}

```

**Listing 3**—The C preprocessor allows you to combine variable declaration and definition in one statement. Only one source file has a #define NET statement, so the variable is initialized in that file and external to all others.

MAKE concludes that SERIAL.OBJ is up to date.

MAKE uses both the time and date in these comparisons, so you must set your PC's clock-calendar correctly. While all PCs since the AT have a battery-powered clock, you should make sure that it is working correctly. If you reset the clock, MAKE may become confused about which files are current. Take care if you have a 24-hour workday around the Daylight Saving Time switchover!

Depending on how many different processors you use, the file extensions may pose a problem. An .ASM file may contain Microsoft MASM statements for 8086 processors, Avocet AVMAC51 statements for 8051 processors, or nearly any assembler for any processor. I use .ASM for the former and .A51 for the latter, but refer to .ASM files in this column where the distinction isn't important.

Listing 4 shows a simplified version of the makefile I used with the RTCMON debugger. There are four main sections: source dependencies, macro definitions, an implicit rule to

## MACINTOSH CROSS ASSEMBLERS

**µASM™** *NEW!!*  
*Version 3.0!*

\*TEXT EDITOR, CROSS ASSEMBLER, AND COMMUNICATIONS FACILITY IN A COMPLETE INTEGRATED DEVELOPMENT ENVIRONMENT

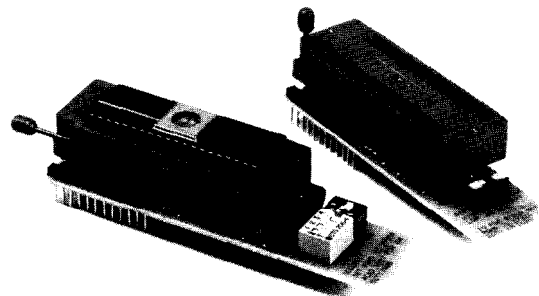
- MACROS
- CONDITIONAL ASSY
- LOCAL/AUTO LABELS
- SYMBOL TABLE CROSS REF
- S OR HEX FILE OUTPUT DOWNLOADS TO MOST EPROM PROGRAMMERS

**US \$149.95**  
**EACH**  
**PLUS S/H\***

AVAILABLE FOR MOST 8-BIT MICROPROCESSORS. CALL OR WRITE FOR TECHNICAL BULLETIN. 30 DAY MONEY BACK GUARANTEE. MC/V/AE.

- PER SHIPMENT:
- \$4 CONTIGUOUS USA
- \$8.50 CANADA AK, HI
- \$15 INTERNATIONAL

**Micro Dialects, Inc.**  
DEPT. C, PO BOX 30014  
CINCINNATI, OH 45230  
**(513) 271-9100**



### 87C51 PROGRAMMER \$125.

Logical Systems brings you support for the Intel 87C51. The UPA87C51 programs this popular microcontroller on general purpose programmers that support the 2732A. With the UPA87C51 you can program the 8751 and 87C51 security bits and the 87C51 encryption array. Logical Systems, helping you get the most out of your programming equipment with our growing line of adapters. OEM inquiries welcome.

ADAPTER	PROGRAMS	PRICE
UPA8751	C8751, 8751H, AMD8753H, 8744	\$95.00
UPA87C51	C8751, 8751H, AMD8753H, 6744 87C51, 87C51FA	125.00
UPA63701V	Hitachi HD63701V0	65.00
UPA451 N	Signetics SC87C451 (64 pin DIP)	125.00
UPA63701X	Hitachi HD63701X0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
UPA63701Y	Hitachi HD63701Y0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
UPA63705V	Hitachi HD63705V0	65.00

**CALL (315) 478-0722 or FAX (315) 475-8460**

**LOGICAL SYSTEMS CORPORATION**  
P.O. Box 6184, Syracuse NY 13217-6184 USA, TLX 6715617 LOGS

```

#-source file dependencies

#UPDATE
rtcmon.obj:    defines.inc macros.inc
cmds1.obj:    defines.inc macros.inc
cmds2.obj:    defines.inc macros.inc
cmds3.obj:    defines.inc macros.inc
cmddec.obj:   defines.inc macros.inc
console.obj:  defines.inc macros.inc
setup.obj:   defines.inc macros.inc
storage.obj: defines.inc
utils.obj:   defines.inc macros.inc
#ENDUPDATE

#- put temporary and OBJ files on RAM disk
WK = H:
.PATH.obj = H:

#- generic Avocet 8031 assembly
#   this depends on having .A51 extensions on the assembler files

.a51.obj :
AVMAC51 ${r,$*}.A51 OJ=${.PATH.obj}${r,$*} PR=${WK}${r,$*} \
MF=${WK}${r,$*}.MXP XR=${WK}${r,$*} PG PL=64 PW=130 NORE
DEL ${WK}${r,$*}.MXP

#- how to build RTCMON.HEX

rtcmon.hex : rtcmon.obj cmddec.obj cmds1.obj cmds2.obj cmds3.obj \
console.obj setup.obj storage utils
avlink51 <@<
rtcmon.hex=${.PATH.obj}rtcmon.obj
${.PATH.obj}cmddec.obj,
${.PATH.obj}cmds1.obj,
${.PATH.obj}cmds2.obj,
${.PATH.obj}cmds3.obj,
${.PATH.obj}console.obj,
${.PATH.obj}setup.obj,
${.PATH.obj}storage.obj,
${.PATH.obj}utils.obj
< MA=${.PATH.obj}rtcmon.MAP SY=rtcmon.SYM -SM -SP -SN

```

Listing 4-A MAKE utility can simplify rebuilding your program because it knows which files depend on others.

Disks	C	a c h e	Time	Disk
3		no	no	682
3		yes	no	345
3		yes	yes	267
100		no	no	450
100		yes	no	300
100		yes	yes	264

Table 1--Time required to compile, assemble, and link the MONITOR program under IBM PC-DOS 3.3. The cache was set for 256 K bytes and the RAM disk for 2.5 MB.

assemble source files, and a specific rule to link RTCMON.HEX.

The source dependencies between the #UPDATE and #ENDUPDATE lines tell PolyMake that most of the .OBJ files depend on DEFINES.INC and MACROS.INC. Although you can generate this information by hand, the MAKEDEPS program analyzes all of the source files to find INCLUDE statements and inserts the dependency lines between the two markers. You must run MAKEDEPS only when you change INCLUDE statements.

The two macro definitions put all of the work and object files on a RAM disk to speed up the assembler. Although this could be done directly in the operation lines, I have learned through experience that you never bury disk identifiers deep in files where they'll come back to haunt you. In the actual MAKEFILE, these values are set from environment variables defined when DOS runs AUTOEXEC.BAT or OS/2 processes CONFIG.SYS.

The implicit rule (.A51.OBJ :) is read as though it said "if you need to

make an .OBJ file and have an .A51 file around, run AVMAC51 with the following options and file names." The sourcedependencies mentioned above tell PolyMake what other files are required for a particular .OBJ file.

The explicit rule (RTCMON.HEX :) tells how to create the RTCMON.HEX file from a collection of .OBJ files. The AVLINK51 program runs from a file containing all the stuff between the last two "<" signs, plus a command line with all the switches shown on the last line of Listing 4.

MAKE programs trace their ancestry back to the original UNIX MAKEs, so obscure options and cryptic switches come with the territory. It took me months to become proficient with PolyMake, but it now remembers everything I know about rebuilding all my programs...if I lose the MAKEFILES, I'm sunk!

Because MAKE executes the compiler as a child process, you may find that there isn't enough RAM for both MAKE and the compiler. PolyMake comes in both large and small varieties, but this is obviously a desperate move to squeak under the 640K-byte barrier. There is a solution, but you may not like it right away: OS/2.

## PUMPING PERFORMANCE

Regardless of which compiler you use, which machine you're using it on, or how clever your MAKEFILE, you won't get results fast enough to suit you. There are a few simple steps you can take to improve your machine's performance at a reasonable cost.

To examine compilation performance, I used the source code for the INKnet and MC-Net MONITOR program. There are 41 files containing 373K of source code, so I have a strong interest in reducing the "compile-link" part of the debugging cycle.

The test hardware is a 20-MHz IBM Model 80-111 with 8 MB of RAM, running either DOS 3.3 or OS/2 1.1. The disk is a 115-MB IBM ESDI drive rated at IO-Mbps peak transfer speed and 28-ms average seek time.

The software I use is Microsoft C 5.10, MASM 5.10, LINK 5.03, and PolyMake 3.1. I haven't tried to iso-

late the difference between DOS mode, protected mode, and bound versions of the same program, nor is this a comparative review of forty-three different compilers.

The results are summarized in Table 1. Basically, you can reduce the compile time about 60 percent by trading RAM space for disk accesses!

Compiling MONITOR with DOS set up "straight from the box" without a disk cache, RAM disk, and BUFFERS=3 takes an astounding 682 seconds, over 11 minutes. The disk sounded like it was trying to scrub the oxide off the platters. I haven't heard noises like that since I fixed the washer.

The disk is obviously the limiting factor, so the best way to boost performance is to reduce disk I/O. There are at least three ways to do that: increase the number of DOS buffers, add a disk cache, and put files on a RAM disk. All of these require additional RAM, so if you have only 640 KB or 1 MB on your machine, you must splurge on more RAM. Fortunately for all of us, DRAM prices are dropping like the proverbial rock; you

can't use high prices as an excuse any longer.

DOS includes a simple disk cache controlled by the BUFFERS= statement in CONFIG.SYS. Changing to 100 buffers dropped the compile time to 450 seconds, a third faster and fully four minutes less waiting. Not bad for a paltry 50 KB of buffers and no charge for new software!

Condition	Time
DOS Box	312
Full-screen, foreground	417
Full-screen, background	430

Table 2—Time required to compile, assemble, and link the MONITOR program under IBM OS/2 1.1. All times use a 254 KB disk cache, 1.7 MB RAM disk, and BUFFERS=3.

All PS/2 setup disks have a hidden file: the IBMCACHE disk cache program. Once you figure out how to install it, it works surprisingly well. In fact, for my applications IBMCACHE is actually better than the highly rated disk cache supplied with Central Point Software's PC Tools. Because

IBMCACHE and the DOS buffers are doing much the same thing, they have a similar effect on performance: IBMCACHE is better, but I gave it 200 KB more buffer space to work with.

Each source file gives rise to several other files, most of which are never read again. The disk caches don't know this, of course, so they discard older data that will be needed in a few seconds. A better solution is to put the intermediate files on a RAM disk. Although details on IBMCACHE are skimpy, it seems to know enough not to cache data destined for the IBM RAM disk.

I set up the IBM VDISK program for a 2.5-MB RAM disk in extended memory. Environment variables put all of the compiler temporary files, as well as the .OBJ and .LST outputs, onto the RAM disk. This, in conjunction with the cache, puts the overall time under 270 seconds, which is largely independent of the number of DOS buffers.

In all honesty, all of the tests have the compiler temporary files on the RAM disk; I forgot to flip the environ-

## 68000 K-System Factory Direct Prices

Now there is a bus that makes it easy to use the entire family of 68000 components. Utilizing native 68000 signals, the K-Bus makes it possible to create low cost 68000 systems in a straightforward manner. The simplicity inherent in the K-System concept allows the system designer the ability to concentrate on meeting the demands of the applications. This same simplicity combined with its low cost makes the K-System ideal for applications ranging from personal use through educational and laboratory applications up to industrial control and systems development. All of this is accomplished at no sacrifice in performance or reliability.

The convenient size (4 x 5 1/4 inch) of the K-Bus boards permits the optimal division of system functions thus simplifying system configuration. The motherboard incorporates integral card guides and compatible power connectors which minimizes packaging requirements. Both SKDOS and OS-9/68000 are fully supported allowing efficient system utilization in both single and multi-user application.

Boards currently in production:

AVAILABLE IN KIT FORM

<b>K-BUS</b>	12 Slots, .8" centers, PC type power connectors	\$129.95
<b>K-CPU-68K</b>	10MHz 68000 CPU, 2 ROM sockets (12 or 16MHz)	\$129.95
<b>K-MEM</b>	256K static RAM or 27256 type EPROMs (OK installed)	\$59.96
<b>K-ACI</b>	2 serial ports with full modem controls (68681)	\$99.96
<b>K-FDC</b>	Floppy disk controller (up to four 5 1/4 drives)	\$99.95
<b>K-SCSI</b>	Full SCSI implementation using 5380 chip	\$99.96
<b>K-DMA</b>	2 channel DMA controller using 68440 chip	\$129.95
<b>K-PROTO</b>	General purpose/wirewrap board	\$39.95
<b>K-xxx-BB</b>	Bare board with documentation for above	\$39.95

### Software:

<b>SKDOS</b>	Single user, editor, assembler, utilities, BASIC	\$150.00
<b>OS-9/68000</b>	Multi-user, editor, assembler, SCRED, utilities BASIC, C, PASCAL, FORTRAN are available	\$300.00

Inquire about our UniQuad line of 68xxx Single Board Computers.

Quantity and package discounts available

Terms: Check, Money Order, Visa, MasterCard—Prices include UPS ground shipment in continental U.S.

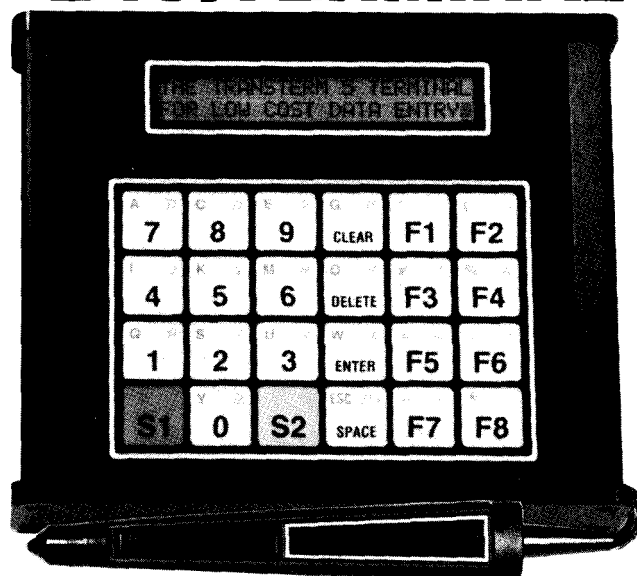
## Hazelwood Computer Systems

Highway 94 at Bluffton  
Rhineland, MO 65069 • (314) 236-4372

UniQuad™ K-Kits™

Reader Service #124

## \$249. TERMINAL



- Featuring • Standard RS-232 Serial Asynchronous ASCII Communications
- 48 Character LCD Display (2 Lines of 24 each)
  - 24 Key Membrane Keyboard with embossed graphics.
  - Ten key numeric array plus 8 programmable function keys.
  - Four-wire multidrop protocol mode.
  - Keyboard selectable SET-UP features—baud rates, parity, etc.
  - Size (5.625" W x 6.9" D x 1.75" H), Weight 1.25 lbs.
  - 5 x 7 Dot Matrix font with underline cursor
  - Displays 96 Character ASCII Set (upper and lower case)
- Options—backlighting for display, RS-422 I/O, 20 Ma current loop I/O.

COMPUTERWISE, INC.

302 N. Winchester • Olathe, KS 66062 • 913-829-0600 • 800-255-3739

Reader Service #113

ment variables and didn't want to blow the rest of the day taking the measurements again. Had the temporary files gone to the hard disk, the no-cache times would be even worse.

With both the cache and RAM disk in effect, the hard disk light blinks when PolyMake starts the compiler on a new file. Because all of the compiler passes don't fit into the cache, it is flushed for every source file. Some informal tests I did indicate that a 512-KB cache doesn't improve things much because the disk is such a small part of the overall compile time. The process is now CPU bound!

What does this mean for you? First of all, if you do not have enough RAM to take advantage of disk caching, buy some. Then, jacking up the BUFFERS= or installing a disk cache will give you the biggest win. Putting temporary files on a RAM disk is a lesser win, but still worthwhile.

If you are using a 286 AT clone, EMS expanded RAM is the only way to go because it is somewhat faster than extended RAM. Clones with a 386 can use the processor's memory mapping hardware to convert extended into expanded RAM, so all you need do is buy the RAM and something like Quarterdeck's QEMM to handle the mapping.

#### MULTI-WHAT?

Of course, you can break the 640-KB barrier and take advantage of that new RAM directly by switching to OS/2. Although that sounds suspiciously like an ad, it turns out to be true. I have been using OS/2 as my "development platform" since last summer, and I'm well pleased with the results. Just for completeness, I ran the performance tests under OS/2 as well.

Table 2 summarizes the OS/2 results on the same hardware. The fact that OS/2 takes longer to compile MONITOR is not surprising, because OS/2 has mode switching overhead during interrupts. The fact that the DOS Box takes only 17% more time than DOS is surprising, particularly in light of all the early doomsayers.

But the DOS Box isn't multitasking, so a better comparison is with a protected mode compile. OS/2 takes about half again as long as DOS to do the compile, nearly seven minutes versus 4:27. Ouch! However, when the compile is put into the background the time increases only a few more seconds, which is comforting.

Does this mean OS/2 is a piece of junk? The answer to that may be the biggest surprise of all: no!

Remember that MAKE compiles only changed files. Rebuilding MONITOR after editing one file takes less than a minute in either environment, so the difference is irrelevant; it takes that long to sit back, stretch, and check the birds at the feeder.

Balanced against the increased compile times are all the advantages of OS/2: real multitasking, relief from the 640K limit, largely crashproof debugging, and an improved operating system interface. Even though most of the programs I develop are meant for real-mode execution, OS/2 is a better development environment.

Indeed, MONITOR would be an ideal protected-mode program. I had

to write a special-purpose multitasking kernel for it; but OS/2 provides far better multitasking built into the operating system. One of these days I'll have to convert it and see how much code "goes away."

If you, like most people, have dismissed OS/2 as something for the future, it is time for another look. If you develop code for a living, the future is now.

#### AND THEY'RE OFF!

This is INK's second anniversary, so, once again, it's time for you to give me some guidance on what you'd like to see here in the upcoming year. Send me a BBS message to tell me where to go...✚

*Ed Nisley is member of the Circuit Cellar INK engineering staff and enjoys making gizmos do strange and wondrous things. He is, by turns, a beekeeper, bicyclist, Registered Professional Engineer, and amateur raconteur.*

#### IRS

- 2 16 Very Useful
- 2 17 Moderately Useful
- 2 18 Not Useful

E

O

W

## COMMUNICATIONS

**508-485-1144**  
FAX 508-481-7222  
BBS 508-460-9203

LOW BUDGET SPECIALS

**19" COLOR SUPER VGA MONITOR**  
**LIKE NEW 6 MONTH WARRANTY** ..... \$800.00  
**USED 3 MONTH WARRANTY** ..... \$600.00  
**USED WITHOUT CASE** ..... \$300 TO \$500.00  
**19" COLOR SONY 1280 X 1024** ..... \$1600.00  
 THESE SONY TRINITRONS HAVE A FULL 6 MONTH WARRANTY PARTS AND LABOR. LIMITED QUANTITY  
**16" Ikegami 64KHZ 1280X1024 NEW** ..... \$899.00  
**16" PANKSONIC 64KHZ USED** ..... \$599.00  
**14" Ikegami TTL VGA CHASIS** ..... \$249.00  
**NEW WITH 1 YEAR WARRANTY**

**19" PHILLIPS 1024X800 48 KHZ GRAY SCALE** ..... \$350.00  
**NEW 1 YEAR WARRANTY - MAY BE ORDERED FOR VGA AT NO EXTRA CHARGE.**  
**WHEN USED IN VGA MODE THE MONITOR WILL RUN 800 X 600 X 256 GRAY SCALE OR 1024X768X16 GRAY SCALE ONLY**

CALL US ABOUT OUR **LARGE VARIETY OF GRAPHIC CARDS !**



194 Main ST. Marlboro, MA 01752

## Building an LED Moving Message Display

How many times have you heard, "I'll just be a minute Honey, we only need a couple of things," as your wife disappears into the supermarket? I could count on one hand the items to purchase: milk, bread, snacks, and some freshly sliced meats from the deli. Sitting in the car, parked on top of those bright yellow diagonally painted lines which seem to constantly shout, "Can't you read, jerk? NO PARKING ZONE!" I kept muttering to myself, "She'll be right out, I'll wait here one more minute." I just couldn't figure out what was so time consuming. Were the customers required to shop on their hands and knees? Was my wife having an affair with the butcher? Or was it "bake your own bread" day in the bakery? I had to know!

The opportunity presented itself the next afternoon, just before leaving work. "Mrs. Jeff on line four," was the page. I punched the flashing button on my desk phone. "We need milk on your way home. Both kinds: regular and low fat." "Already?" I gasped. I knew this was it. "OK, see ya soon," I answered, ending the conversation. The phone cord was still swinging as I exited in a dash, one arm already threaded through a coat sleeve.

You find the strangest people in the grocery store. From people with calculators who know what their bill will be even before they get to the register, to moms who bring their kids along to help and end up with more cookies, candy, and cereal than anything else. Something seemed different today. Every aisle contained people standing around, looking up as if there were fireworks on the Fourth of July. The shopping cart traffic was at a standstill while their drivers stood reading scrolling advertisements hanging from the ceiling like circus trapezes. Three and four displays in every aisle, each with its own continually changing message. "DUZ DUZ IT, to all of your dishes." "10 for 99 cents-get SLICK disposable razors." "DOUBLE COUPON item-Minute Mash, now Microwaveable." It was mesmerizing and worse, frightening. There was no way to avoid them. Displays so big and bright you could read them from 50 feet away.

"Milk, ah yes," I thought. Spinning around after finally getting my bearings, I made my way to the dairy cooler. There were feeding frenzies taking place at various points in the store. Obviously, the advertising blitzes were effective. I grabbed the milk and left without spending more than an additional \$18. Not bad for 43 minutes.

## UNNATURAL RESOURCES

New technologies are constantly gobbled up by the advertising media. One benefit of their prolific use is to **bring down** the price of technology to a point where we can put it to a more productive use.

My last column described the basics of LED displays and included a small project: an 8x10 LED array display. Although it **was** designed to be used with any 8255 PPI, the RTC52 and RTCIO were chosen as the engine because most of the readers are familiar with them. *[Editor's Note: For more information about the RTC52 and RTCIO boards, see "From the Bench" in the April/May 1989 issue of CIRCUIT CELLAR INK.1* Before moving on, let me present the software promised for the 8x10 display presented last issue.

Listings 1, 2, and 3 show sample code which updates the display as a background task running on an RTC52. BASIC can be used to manipulate data in the RAM used as a display buffer. A ten-byte display window maps directly to the ten columns of LEDs. The background routine grabs data one byte at a time and applies it to port A of the 8255. Ports B and C enable each column in sequence, one per interrupt. Even though only one column is active at a time, the persistence of our retinas blurs the separately enabled columns into a coherent image. TIMER0's overflow interrupts BASIC and displays the next column's data. At the end of each interrupt, it's back to BASIC again until the next overflow.

```

10 MTOP=1F00H : REM IN CASE ONLY 8K RAM
20 INPUT "WHAT IS THE 8255 BASE ADDRESS
   (0E800H)?"B
30 DBY(19H)=B-(INT(B/256)*256) : REM 8255 LSB
40 XBY(B+3)=80H : REM CONFIGURE AS ALL OUTPUTS
50 TIMER0=0
60 TMOD=12H : REM TIMER0 = 8-BIT AUTO RELOAD
70 DBY(18H)=INT(B/256) : REM 8255 MSB
80 DBY(1AH)=1FH : REM BUFFER MSB
90 DBY(1BH)=00H : REM BUFFER LSB
100 DBY(1CH)=0 : REM START WITH COLUMN ZERO
110 REM NOW FILL THE DISPLAY BUFFER WITH NULL
    DATA
120 FOR Q=1FF0H TO 1FFFH
130 XBY(Q)=0
140 NEXT Q
150 IE=82H: REM THIS ENABLES THE DISPLAY
160 REM PUT SOME DATA INTO THE BUFFER
170 FOR Q=0 TO 7
180 XBY(1F10H+Q)=XBY(1F10H+Q-1)+2**Q
190 NEXT Q
200 FOR Q=0 TO 7
210 XBY(1F18H+Q)=XBY(1F18H+Q-1)-2**Q
220 NEXT Q
230 PRINT "HIT ANY KEY TO STOP"
240 REM NOW MOVE THE DISPLAY BUFFER LSB
250 FOR Q=0 TO 20H
260 FOR X=0 TO 5 : NEXT X : REM DELAY LOOP
270 DBY(1BH)=Q
280 NEXT Q
290 IF GET<>0 THEN 310
300 GOT0 250
310 DBY(1BH)=0 : REM POINT TO NULL DATA (BLANK
    DISPLAY).
320 IE=40H : REM TURN OFF INTERRUPTS
330 END

```

listing 1 - A sample BASIC program that initializes the hardware, enables interrupts, and updates the display memory in the foreground.

```

*****
;
;      8 X 10 LED ARRAY DISPLAY
;
;      JEFF BACHIOCHI
;      11/11/1989
;
;      REQUIREMENTS:
;
;      DBY(18H) = 8255 BASE ADDRESS MSB
;      DBY(19H) = 8255 BASE ADDRESS LSB
;      DBY(1AH) = DISPLAY BUFFER ADDRESS MSB
;      DBY(1BH) = DISPLAY BUFFER ADDRESS LSB
;      DBY(1CH) = 00H (START AT COLUMN ZERO)
;
;      ALSO USES REGISTERS 1DH, 1EH AS
;      TEMPORARY STORAGE
;
*****
;
;      ORG      400BH
;
VEC:  LJMP     BEG      ;TIMER0 INT VECTOR
;
;      ORG      4200H
;
BEG:  CLR      EA      ;STOP ALL FURTHER INTS
      PUSH    ACC     ;AND SAVE ALL REGS USED
      PUSH    B
      PUSH    DPH
      PUSH    DPL
      MOV     A,1CH   ;GET COLUMN #
      CJNE   A,#0AH,G01 ;COLUMNS = 10?
      CLR    A       ;IF SO RESET COLUMN #
      MOV    1CH,A
G01:  SETB    C       ;READY TO ROTATE
      CLR    A
      MOV    B,1CH   ;HOW MANY TIMES?
      INC   B       ;ADJUST IT
LP1:  RLC     A       ;ROTATE
      JC    LP2     ;BACK TO START?
      DJNZ B,LP1   ;DO UP TO 8 TIMES
      MOV   1DH,A   ;SAVE IT FOR PORT B
      MOV   1EH,#00 ;SAVE ZERO FOR PORT C
      SJMP  IT
LP2:  RLC     A       ;ROTATE
      DJNZ B,LP2   ;DO UP TO 2
      MOV   1DH,#00H ;SAVE ZERO FOR PORT B
      MOV   1EH,A   ;SAVE IT FOR PORT C
IT:   MOV    DPH,1AH ;BUFFER MSB
      MOV    DPL,1BH ;BUFFER LSB
      MOV    A,1CH   ;COLUMN # AS OFFSET
      CLR    C
      ADD   A,DPL   ;ADD IN OFFSET
      MOV   DPL,A   ;THEN REPLACE
      CLR    A
      ADDC  A,DPH   ;ADD IN ANY CARRY
      MOV   DPH,A   ;THEN REPLACE
      MOVX  A,@DPTR ;DATA FROM BUFFER
      MOV   DPH,18H ;8255 BASE MSB
      MOV   DPL,19H ;8255 BASE LSB
      MOVX  @DPTR,A ;DATA TO PORT A
      INC   DPTR   ;MOVE TO PORT B
      MOV   A,1DH  ;TEMP DATA
      MOVX  @DPTR,A ;TO PORT B
      INC   DPTR   ;MOVE TO PORT C
      MOV   A,1EH  ;TEMP DATA
      MOVX  @DPTR,A ;MOVE TO PORT C
      INC   1CH    ;NEXT TIME NEXT COLUMN
      POP   DPL    ;REPLACE ALL REGS USED
      POP   DPH
      POP   B
      POP   ACC
      POP   PSW   ;THIS ONE NEEDS POPPING
      SETB  EA    ;ENABLE INTS
      RETI   ;OUT-A-HERE
;
*****
;
;      END

```

listing 2 - This machine language handles the TIMER0 overflow interrupt jump vector and outputs the contents of the display memory in the background.

```

1  REM APPEND THIS IF USING A 32K RAM
2  REM TO LOAD THE INTERRUPT ROUTINE
3  GOSUB 500
500 D=4200H : C=0
510 READ B
520 IF B=0A5H THEN 570
530 XBY(D)=B
540 D=D+1
550 C=C+B
560 GOTO 510
570 IF Co33627 THEN PRINT "CHECKSUM ERROR":END
580 IF XBY(4200H)<>0C2H THEN PRINT "NO RAM":END
590 XBY(400BH)=02H : REM INST. TIMER0 JUMP VECT.
600 XBY(400CH)=42H
610 XBY(400DH)=00H
620 RETURN
630 DATA 0C2H,0AFH,0C0H,0E0H,0C0H,0F0H,0C0H,083H
640 DATA 0C0H,082H,0E5H,01CH,0B4H,00AH,003H,0E4H
650 DATA 0F5H,01CH,0D3H,0E4H,085H,01CH,0F0H,005H
660 DATA 0F0H,033H,040H,00AH,0D5H,0F0H,0FAH,0F5H
670 DATA 01DH,075H,01EH,000H,080H,009H,033H,0D5H
680 DATA 0F0H,0FCH,075H,01DH,000H,0F5H,01EH,085H
690 DATA 01AH,083H,085H,01BH,082H,0E5H,01CH,0C3H
700 DATA 025H,082H,0F5H,082H,0E4H,035H,083H,0F5H
710 DATA 083H,0E0H,085H,018H,083H,085H,019H,082H
720 DATA 0F0H,0A3H,0E5H,01DH,0F0H,0A3H,0E5H,01EH
730 DATA 0F0H,005H,01CH,0D0H,082H,0D0H,083H,0D0H
740 DATA 0F0H,0D0H,0E0H,0D0H,0D0H,0D2H,0AFH,032H
750 DATA 0A5H

```

listing 3—This BASIC program can be appended to Listing 1 to poke Listing 2's object code into RAM at 4000H.

## E-X-P-A-N-D-I-N-G ON THE IDEA

The maximum average current for an LED in these arrays is about 20 mA, with peak currents of about 100 mA allowed. Peak currents come into play when we multiplex rows or columns of LEDs. When multiplexed, each row or column is on for only  $1/n$  of the total time (where  $n$  is the number of rows or columns and duty cycle is  $1/n$  times 100%). So, for a row or column LED to appear to be the same intensity as a nonmultiplexed LED which is always on, the row or column LED must be  $n$  times brighter. Fortunately, the current necessary to do this is somewhat less than  $n$  times because the efficiency of an LED goes up as the current increases. As the number of columns increases, the duty cycle becomes smaller, until finally even boosting the LED current to the maximum allowable will not produce enough intensity for the display to be seen clearly.

Let's take the relatively simple project presented last issue and expand it into a scrolling sign similar to the ones you see in stores, airports, and sports bars. The intent of the design is to give a highly visible display to applications where dangerous conditions exist. Single LEDs, LCDs, and CRTs can all be used for status conditions, but in many applications the operators may not be within range to easily read such information. Using a large LED display can communicate important information to anyone immediately without a fifty-yard dash. To eliminate the problem of increasingly smaller duty cycles, we treat each 5x8 module as a separate entity. Each module would have a maximum duty cycle of  $1/5$  (20%), eliminating the contrast and maximum peak current problem. Designing for a maximum LED peak current of 100 mA (which, at a 20%

duty cycle, is equal to 20 mA average current) sounds easy, but finding a standard device to sink 100 mA for each of eight column LEDs is not. After perusing many data books, I came across a Sprague part which contains an 8-bit latch capable of sinking 100 mA for each of the eight bits. With all LEDs on, that's 800 mA; not too shabby since most TTL devices have a tough time with 20 mA/bit.

Now that we can sink the 800 mA coming through the column LEDs, we must be able to source and control it. A plain old PN2222 had the oomph I was looking for, capable of handling up to one amp of collector current. Refer to Figure 1 for the schematic of the expandable array display. Note that circuitry for only one array module is shown. This is duplicated eight times on the board, once for each display module.

## NEW CIRCUITRY-OLD RULES

To ease the design and implementation of this new display, we will keep the same restraints as before: the display's control will come from the three ports of an 8255, but this time with a bit more functionality assigned to each port. Port A will pass column data to the arrays. The first five bits of port B will be column enables and the last three will be module enables. Port C will consist of eight board enables. Each board will control eight 5x8 LED array modules. Up to eight boards can be used, totaling 64 5x8 LED arrays (that's six feet worth of characters!). When a board is enabled by port C, the top three bits on port B are decoded into a strobe for one of the eight modules on the board. This strobe latches the column data from port A and the column enable from the lower five bits on port B. Each module has its own data latch and a column latch which holds the module's column LEDs lit until the next access to that module.

With the above system, any one of the five columns on any one of the eight modules on any one of the eight boards can be latched with the data on port A. A single-board system can display up to eight characters while an eight-board system can display up to 64 characters at a time. The boards can be stacked vertically for a 64-x40-pixel display, or horizontally for an 8-x320-pixel display. The information displayed is limited only by your imagination and programming ability.

## THE NECESSARY EVIL-SOFTWARE

The code developed for this project allows any one of 32 character canned messages to be scrolled through the display using from one to eight display boards. Alternatively, a 64-character message can be entered live through the RS-232 or RS-485 port of the RTC52 used to control the display. Scroll speed can be adjusted from the keyboard, or the display can be set into a nonscrolling mode to display a short, constant message. A portion of the system RAM is set aside for use as a display buffer. The beginning of the display buffer is first padded with null data, which allows the display to start blank. Next, each character of

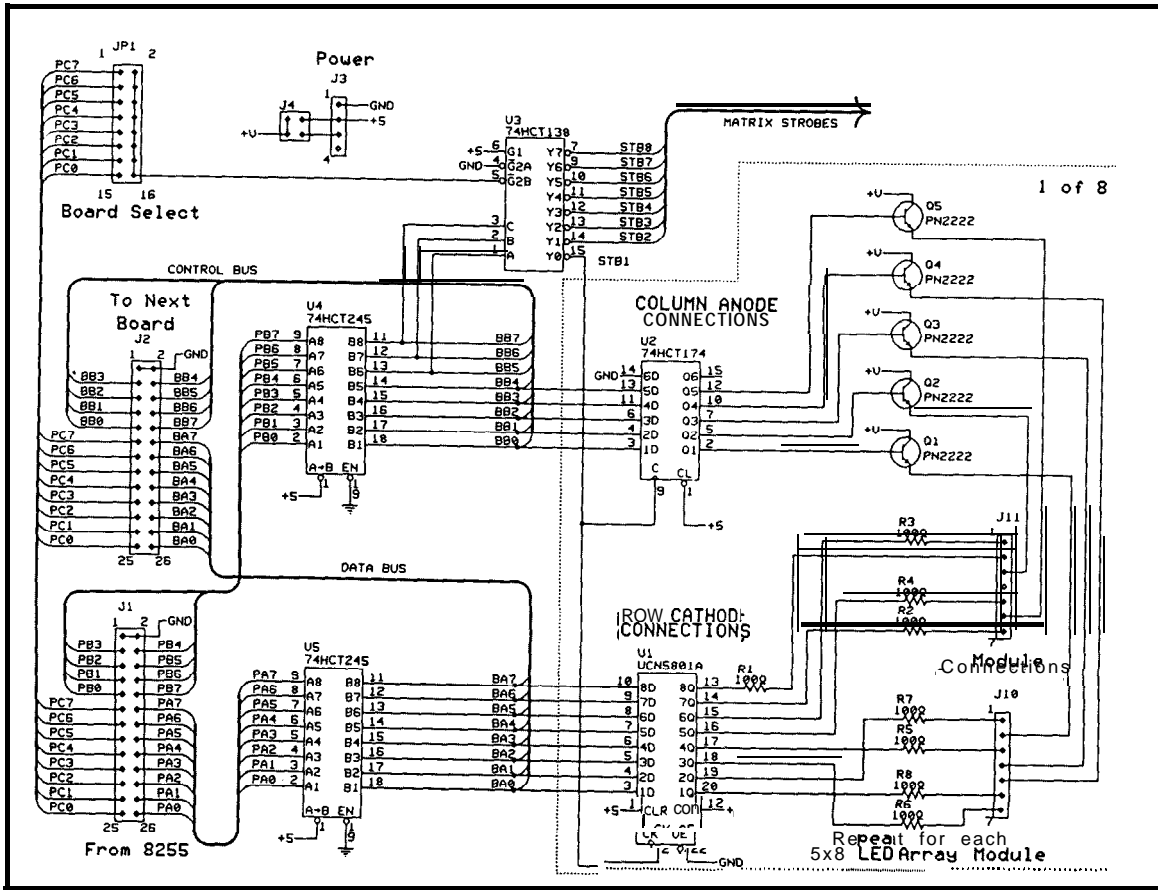


Figure 1—The expandable array display uses 30-ohm series resistors to restrict LED current to 60mA, and Sprague UCN5801A latches capable of sinking 100 mA on each bit. The UCN5801A comes in an unusual 22-pin 0.4inch-wide DIP package.

## MS-DOS EPROM PROGRAMMING SYSTEM NEEDS NO INTERNAL CARD

**EPROMS**  
 2708 (3 supply)  
 2758, 2716  
 27C16, 2516  
 2532\*, 2564\*  
 68764\*, 68766\*  
 2732, 2732A  
 27C32, 2764  
 2764A, 27C64  
 27128, 27128A  
 27C128, 27256  
 27C256, 27512  
 27C512, 27C010\*  
 27010\*, 27C1001\*

**EEPROMS**  
 2804, 2816A  
 2864A, 28256\*

**MicroControllers**  
 8741A\*, 8742  
 8748\*, 8748H\*  
 8749\*, 8749H\*  
 8751\*, 87C51\*  
 8752\*, 8744H\*

\*Socket Adapter Required (Diagrams included)

CONNECTS TO YOUR SYSTEM'S  
**PARALLEL PRINTER INTERFACE**

• A FULL FEATURED, EASY-TO-USE SYSTEM WORKS WITH ANY DESKTOP OR LAPTOP MACHINE  
 • ADAPTIVE, HIGH-SPEED ALGORITHM MINIMIZES PROGRAMMING TIME AND INSURES VALID DATA  
 • SYSTEM PROGRAMS ALL STANDARD DEVICES OR EQUIVALENTS FROM ANY MANUFACTURER  
 ALL SYSTEM COMPONENTS FIT NEATLY INTO CASE FOR TRAVEL OR STORAGE

### SYSTEM SOFTWARE COMMANDS

• PROGRAM EPROM(S) FROM DISK FILE	• SAVE EPROM(S) OR BUFFER TO DISK	• COPY EPROM(S) : VERIFY EPROM
• READ DISK FILE INTO BUFFER	• PROGRAM EPROM(S) FROM BUFFER	• ERASED : BUFFER EDITOR
• READ EPROM(S) INTO BUFFER	• COMPARE EPROM(S) WITH BUFFER	• SELECT DEVICE TYPE : DEVICE CHECKSUM

**BUFFER EDIT/TOV HAS 18 BYTE LEVEL COMMANDS FOR DETAILED OPERATIONS**

SYSTEM INCLUDES: PROGRAMMING UNIT, POWER PACK, CONNECTING CABLE, OPERATION MANUAL & SOFTWARE **\$239**

SOFTWARE AVAILABLE ON 3 1/2" OR 5 1/4" DISK  
TO ORDER SEND CHECK, MONEY ORDER. WRITE OR CALL:

VISA

**ANDRATECH**  
 P.O. BOX 222  
 MILFORD, OHIO 45150  
 (513) 831-9708

MASTER CARD

CALL OR WRITE FOR MORE INFO. -- ADD \$5.00 FOR SHIPPING - \$4.00 COD

# A-BUS™

## Data Acquisition & Control

Low Cost A/D, Motion Control, Relays, D/A, Digital I/O, Counters...

**Sample Applications:**  
 Read sensors: voltages, light levels, temperatures, keypads, touch-tones:  
 Switch electrical devices:  
 Automate experiments:  
 Test equipment.

**Bus adapters for:**  
 PC/XT/AT & compatibles:  
 Apple II: Tandy and others. Or use RS-232 serial adapters for remote applications

### New Products

FA-154: 12-bit 8-channel A/D, 10µs	\$17
MC-1 08: Metal cover for Motherboard	\$4
AC-1 09: Acrylic cover for Motherboard	\$4

**Call for new catalog: (203) 6564806**  
 Mon-Fri 9-5 Eastern time (or Fax 203 656-0756)

**ALPHA Products**  
*"Innovation through Application"*  
 242-C West Avenue, Darien, CT 06820

Reader Service #102

Reader Service #101



the message string is converted into display data using a look-up table.

Finally, the display buffer is filled to the end with null data. Once the hardware is initialized and the interrupts enabled, the background code continually transfers the data held within the display buffer into the latches for each of the 64 possible LED array modules. The routine is fed with buffer start and end addresses and a scroll or no scroll flag.

At each interrupt, the service routine must transfer data for one column of each module on each board, and must be short enough to update up to 64 modules five times, once for each column, at a rate which fools the eye into thinking all the LEDs are on at once. This routine, the results of which are shown in Figure 2, takes a maximum time of 200  $\mu$ s every 4 ms. That's 20 ms for five scans, one for each of the five columns, or 50 Hz, which is about the minimum time for our eyes to see one image.

If in scrolling mode, the interrupt routine must also determine at which point to move the scroll window. The scrolling window shown in Figure 3 is a view of the display buffer. It is the scroll window that moves through the display buffer and not movement of the data in the buffer that gives the illusion of scrolling. A scroll counter is decremented once each scan. When the scroll counter reaches zero, the scroll window is advanced. The counter's

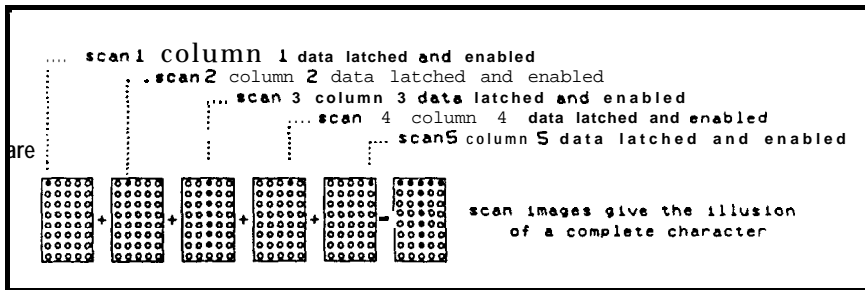
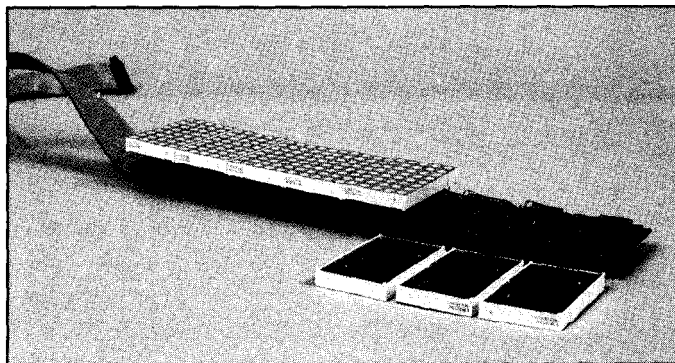


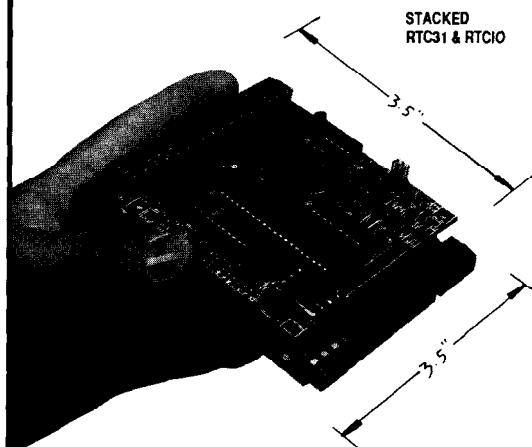
Figure 2—Each column scan displays one strip of the total character. Our eyes merge these into one.



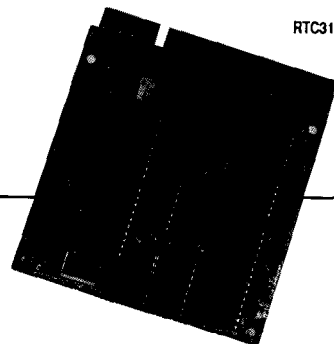
## MICROMINT Introduces "Micro" Controlling!

After years of experience in manufacturing OEM controller boards and talking to customers, we think we have hit upon just the right combination of format and function to satisfy even the toughest case of "relay mentality." Realizing that not every computer/controller application warrants a Cray X/MP, Micromint offers a tiny 8031/8052-based controller board for those dedicated and cost-sensitive installations.

New **MC-NET™** software links your desktop up to 31 RTC controllers.



RTC31



### RTC31 and RTC52 Technical Specifications

- 8031 processor (RTC31) or Micromint 80C52-BASIC processor (RTC52)
- 11.05-MHz system clock
- Uses 8K or 32K memory chips
- Up to 64K bytes of RAM or EPROM
- 5-volt-only operation
- 11 0-1 9200 bps AS-232 and/or RS-485 serial port
- Use stand-alone or networked
- 12 bits of parallel I/O
- Vertical-stacking expansion bus
- Screw terminal connections
- Small 3.5"x3.5" format
- 60 mA typical operating current (RTC52)

RTC31-1	8031 Controller	\$119.00
RTC31-1	OEM 100-Quantity Price	\$79.00
RTC52-1	80C52 Controller	\$139.00
RTC52-1	OEM 100-Quantity Price	\$99.00

### RTCIO Technical Specifications

- Three bidirectional parallel ports (24 bits)
- 8-channel, 8-bit A/D (0-5V); 9,000 samples/sec
- 4-channel, 8-bit D/A (0-5V); 2- $\mu$ s response time
- Battery-backed clock/calendar and presettable time-interrupted capability
- DC to DC conversion—5-volt-only operation
- Screw terminal connections
- Small 3.5"x3.5" format

RTCIO	RTCIO board with parallel I/O and A/D converter	\$129.00
RTCIO	OEM 100-Quantity Price	\$89.00



Micromint, Inc.

4 Park Street, Vernon, Connecticut 06066  
Tel: (203) 871-6170 • Fax: (203) 872-2204

### Available Soon!

- RTC-OPTO 8-channel Optoisolated I/O Expansion Board (expected availability December, 1989)
- RTC-SIR Serial, Timer, and Infrared I/O Expansion Board (expected availability January, 1990)
- RTC-LCD LCD, Keyboard, and X-10 I/O Expansion Board (expected availability February, 1990)

**ADVERTISER'S INDEX**

Reader Service Number	Advertiser	Page Number
101	Alpha Products	63
102	Andratech	63
103	Ariel Corp.	33
104	Avocet	C4
105	Baradine	75
106	Berry Computers	4
107	Big Bang Software	67
108	Binary Technologies	45
109	Cabbage Cases	67
110	Catenary Systems	27
.	Ciarcia Design Works	78
111	Circuit Cellar	57
112	Circuit Cellar	57
113	Computerwise	58
114	Cottage Resources	37
115	Covox Inc.	11
116	DATAx	79
117	Dycor Industrial	29
118	Emerald Microwave	55
119	Engineers Collaborative	75
120	Express Circuits	46
121	F&W Communications	59
162	GTEK	C3
122	Gott Electronics	13
123	Grammar Engine	11
124	Hazelwood	58
125	Information Modes	79
126	Innotec	41
127	Introl Corp	26
128	Kolod Research, Inc.	49
129	Laboratory Microsystems	28
130	Lear Corn Company	79
131	Logical Systems	54
132	Meredith Instruments	79
133	Micro Dialects, Inc.	54
134	Micro Digital	39
135	Micro Resources	40
136	Micromint	24
137	Micromint	64
138	Micromint	20
139	Micromint	40
140	Ming Engineering	39
141	NOHAU Corp	45
142	Omotion	48
143	Paradigm Systems	41
144	Parallax, Inc.	16
146	PI Computer	6
147	PseudoCorp	50
148	Quinn Curtis	11
149	R & D Electronics	49
150	Real Time Devices	44
145	Ryle Design	79
151	Sierra Systems	C2
152	Silicon Alley	35
153	Sunnyside Solar	71
154	Systronics	79
155	Tardis Systems	70
156	Timeline	7
.	Tinney	49
157	Traxel Laboratories Inc.	71
158	Unkel Software	19
159	Louis E. Wheeler	79
160	Witts Associates	79
161	Wytec Company	28

**HAJAR ASSOCIATES  
NATIONAL ADVERTISING SALES  
REPRESENTATIVES**

**NORTHEAST**

Lisa D'Ambrosia  
Tel: (617) 769-8950  
Fax: (617) 769-8982

**MID-ATLANTIC**

Barbara Best  
Tel: (201) 741-7744  
Fax: (201) 741-6823

**SOUTHEAST**

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

**MIDWEST**

Nanette Traetow  
Tel: (708) 789-3080  
Fax: (708) 789-3082

**WEST COAST**

Barbara Jones & Shelley Rainey  
Tel: (714) 540-3554  
Fax: (714) 540-7103

**SUBSCRIPTION  
PROBLEMS?**

If you have problems with your subscription (delayed or missing issues, change of address, or questions on renewals), call the Circuit Cellar INK Subscriber Service Line at (914) 628-0885 or write:

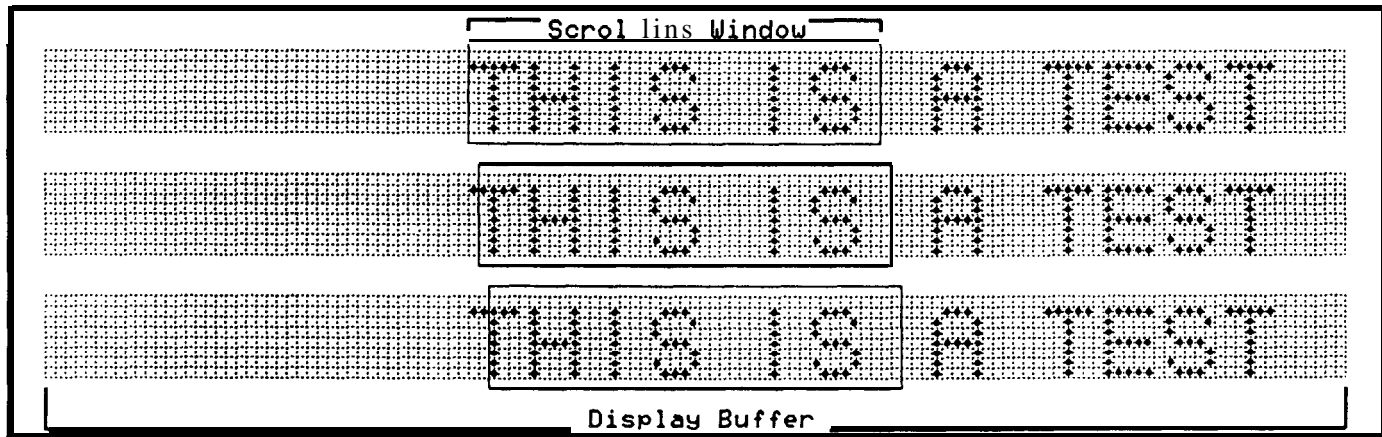
**Circuit Cellar INK  
Subscriber Service Dept.  
P.O. Box 2099  
Mahopac, NY 10541**

**IRS****INK Rating Service**

How useful is this article?

At the end of each article and some features there are 3-digit numbers by which you can rate the article or feature.

Please take the time to let us, at Circuit Cellar INK, know how you feel our material rates with you. Just circle the numbers on the attached card.



**Figure 3**—The scrolling window is a limited view into the display buffer. The buffer holds all the data used to display a message while the window selects which section of the buffer will be displayed.

reload value controls the scroll speed. The user can adjust the reload value by using the "1" key to speed it up (decrease the reload value) and the "2" key to slow it down (add to the reload value).

### CUSTOM CHARACTER CONVERSION

The character conversion is done for a 5x7 character with a leading blank column. The high bit is off in each column, which makes the 5x7 character lower-right justified in a 6x8 block. With minor modifications to the con-

version routine (specifically the number of columns transferred for each character), any width character can be displayed. Define your own size character set or use one from a font editor. The upper 128 characters of the ASCII table could be defined as graphics characters similar to those in the IBM character set.

### THE SMART DISPLAY

The physical dimensions for the display board are 2.4" high by 12" long. Total LED current could be as much as

# It's That Time Again!

## The Second CIRCUIT CELLAR INK Design CONTEST

CALL FOR ENTRIES

Yes, sports fans, it's time to sharpen your pencils, heat your soldering iron, and sort through your chip collection, because the Second CIRCUIT CELLAR INK Design Contest is here! This year, the emphasis is on creativity, with solid design, a unique approach, and an eye to elegant utility being keys to a successful entry.

Each entry must be accompanied by an Official Entry Form. To receive an Official Entry Form and complete set of rules, send a self-addressed, stamped #10 envelope to:

CIRCUIT CELLAR INK  
Design CONTEST  
4 Park Street, Suite 90  
Vernon, CT 06066

All entries must be received by **May 4, 1990**. An individual may enter more than once, but each entry must be received separately, and must be accompanied by an Official Entry Form. Winners will be announced in the August/September 1990 issue of CIRCUIT CELLAR INK.

### PRIZES!

Yes, of course there are prizes! First prize is worth \$500, Second Prize is worth \$200, and Third Prize is worth \$100. In addition, the judges may award Honorable Mention Prizes that consist of \$50 and a one-year subscription (or extension) to CIRCUIT CELLAR INK.

### TWO CONTESTS IN ONE!

Once again, there are two categories: Open and Cost Effective. The Cost Effective category seeks to reward those who work with simple, low-cost controllers and processors in cost-effective designs. Identical prizes will be awarded in both categories. Placement of an entry into a category is at the sole discretion of the judges.

SEND FOR **YOUR OFFICIAL ENTRY FORM AND RULES** TODAY!!

6.4 amps with all LEDs on, depending on the value selected for the series resistors. In reality, average current consumption is about two amps per board unless you choose to display black on white (or red). A 26-conductor ribbon cable carries the control signals to each board attached in a daisy chain configuration. A dedicated LED power supply is attached to each board separately, reducing the wire size necessary to supply maximum current to all boards.

Because of the small physical size of the RTC52 microcontroller system, the microcontroller can be easily contained within the same enclosure as the display. This produces a compact unit capable of stand-alone operation as well as a smart display using RS-232 or RS-485 as a communication interface.+

*Jeff Bachiochi (pronounced "BAH-key-AH-key") is a member of the Circuit Cellar INK engineering staff. His background includes work in both the electronic engineering and manufacturing fields. In his spare time, Jeff enjoys his family, windsurfing, and pizza.*

## IRS

2 19 Very Useful  
220 Moderately Useful  
221 Not Useful

STATEMENT REQUIRED BY THE ACT OF AUGUST 12, 1970: SECTION 3685, TITLE 39, UNITED STATES CODE SHOWING THE OWNERSHIP MANAGEMENT AND CIRCULATION OF CIRCUIT CELLAR INK, published bi-monthly at 4 Park Street, Vernon, CT 06066. Annual subscription price is \$14.95. The names and addresses of the Publisher, Editorial Director, and Editor-in-Chief are: Publisher, Daniel Rodrigues, 4 Park Street, Vernon, CT 06066. Editorial Director, Steven Garcia, 4 Park Street, Vernon, CT 06066. Editor-in-Chief, Curtis Franklin, 4 Park Street, Vernon, CT 06066. The owners: Circuit Cellar Inc., Vernon, CT 06066. The names and addresses of stockholders holding one percent or more of the total amount of stock are: Steven Garcia, 4 Park Street, Vernon, CT 06066. The average number of copies of each issue during the preceding 12 months are: A) Total number of copies printed (net press run) 29,927 B) Paid Circulation (1) Sales through dealers and carriers, street vendors and counter sales: 3,806 (2) Mail subscriptions: 20,296 C) Total paid circulation: 24,102 D) Free distribution by mail, carrier, or other means: samples, complimentary, and other free copies: 877 E) Total Distributed: 24,979 F) Copies not distributed: (1) Office use leftover, unaccounted, spoiled after printing: 1,650 (2) Returns from News Agents: 3,296 G) Total: 29,927 actual number of copies of the single issue published nearest to filing date are: (#11 Oct/Nov) A) Total number of copies printed (net press run) 32,000 B) Paid Circulation: (1) Sales through dealers and carriers, street vendors, and counter sales: 4,750 (2) Mail subscriptions: 24,213 C) Total paid circulation: 28,963 D) Free distribution by mail, carrier or other means: samples, complimentary, and other free copies: 932 E) Total Distributed: 29,895 F) Copies not distributed: (1) Office use, leftover, unaccounted, spoiled after printing: 1,000 (2) Returns from News Agents: 1,105 G) Total: 32,000 I certify that the statements made by me above are correct and complete. Daniel J. Rodrigues - Publisher

The following items are available from

Circuit Cellar Kits  
4 Park St., Suite 12  
Vernon, CT 06066  
(203) 875-2751

1. Blank PC board, manual, and demo software on 5.25" 360K PC-format disk. SD-1 ..... \$49
2. Eight Sprague UCN5801A driver chips. SD-2 ..... \$20
3. Eight LTP2058AE red 5x8 LED array modules. SD-3 .... \$40

Please add \$3 shipping and handling in U.S.; \$8 elsewhere.

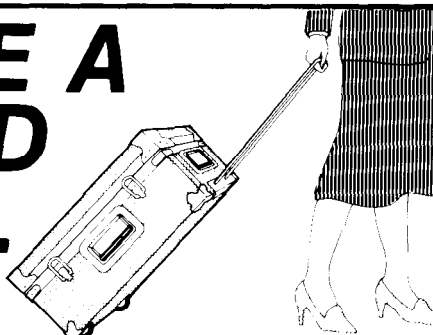
Innovations like these help to make today's technology more cost-effective, reliable, and easier to use. Please share your favorite ideas, chips, and circuits with others.

We will pay \$25 for any From the Bench accepted for publication. All submissions should be typed, double-spaced, and include neatly drawn schematics or Schema configuration, library, and page files.

Include a stamped, self-addressed envelope large enough to hold everything if you wish materials that have not been accepted to be returned.

Submit to: From the Bench  
c/o Circuit Cellar INK  
P.O. Box 772  
Vernon, CT 06066

# TAKE A LOAD OFF...



A rugged CABBAGE CASE? lined with plenty of foam for your equipment can **TAKE A LOAD OFF YOUR MIND** when you've got to travel.

**TAKE A LOAD OFF YOUR BACK** with our exclusive tilt-wheels and extension handle option.



**UNLOAD ON US!**

Call or write to tell us about your shipping or carrying problems  
**WE HAVE SOLUTIONS!**



CABBAGE CASES, INC.  
1166-C STEELWOOD ROAD  
COLUMBUS, OHIO 43212-1356  
(614) 466-2495 (800) 888-2495

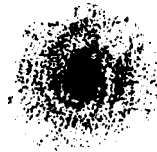
## MC68000/10/20 SIMULATORS

**DEBUG 68000/10/20 PROGRAMS  
ON YOUR PC**

SIM68K and SIM20 are software simulations of the MC68000/10 and MC68020 CPUs, designed to run on the IBM PC and compatibles with DOS 2.0 or higher. No additional hardware required. They accept files generated by a 68000/10/20 assembler. They are complete debugging tools, providing access to registers, flags, and memory. All 68000/10/20 instructions, addressing modes and condition codes are supported.

- Load/Dump Facilities • Disassembler
- Single-Step & Fast Execution • Breakpoints
- Interrupts • Terminal I/O Support
- Symbolic Debugging • Execute Batch Files

SIM68K (MC68000/10).....\$285  
SIM20 (MC68020)..... \$345



**BIG BANG SOFTWARE, INC.**  
7151 W. Hwy. 98, Suite 286  
Panama City Beach, FL USA 32407  
Phone 904-784-7114 Fax 904-235-3475

### A Report from the Second Microprocessor Forum

**A**s I write this, the dust is still settling from the “almost big one” that hit Northern California October 18. If you’re ever in an earthquake, just remember it’s like surfing on solid ground. If you can’t surf, move to Kansas.

Even a 7.1 won’t stop the wizards of Silicon Valley for long (now an 8.1, that’s another story!). They’re hard at work on the chips that will shape the next era of the microelectronics revolution.

I recently attended the Second Annual Microprocessor Forum—a two-day conference, hosted by Mike Slater, editor of the Microprocessor Report newsletter. The Forum brings together the key academic and industrial microprocessor gurus to predict (and promote their version of) the future. Judging from this **year’s gathering**, the chips of the ‘90s will be shaking things up too.

#### ‘ISC WARS

There are three aspects of CPU design which determine performance: architecture, implementation, and technology. A good analogy is a building, whose “performance” is arguably the combination of design (architecture), construction (implementation), and materials (technology). Often these factors overlap, making it hard to separate their respective influences.

The RISC/CISC war continues to dominate the industry mindset. For those of you who don’t already know my opinion (“RISC vs. Reality,” **CIRCUIT CELLAR INK #1**), I’ll state it again..

\*The ‘ISC warriors often confuse architecture, implementation, and technology. Caching and pipelining are implementation techniques which have little to do with (Instruction Set Architecture). Certainly, manufacturing technology has nothing to do with instruction set.

\*CPU performance, whether RISC or CISC, is only one component of system performance. ‘ISC warriors seem to live in a world without I/O.

\*RISC vs. CISC is largely a marketing, not technology, battle. This doesn’t diminish the importance of the battle, but does indicate much of the rhetoric is misguided.

These days, the contenders are more strident than ever (too many chips and not enough sockets?). The RISC and CISC advocates alternately bash one another’s camps. The fight card looks like mud wrestling: i860 vs. MC68040 vs. MIPS vs. SPARC vs. i486 vs. MC88000. Place your bets! The most arcane issues—Is the ‘860 a CPU or coprocessor? What is a 64-bit CPU? More registers or fewer? Vector, superscalar, or VLIW (Very Long Instruction Word)?—lead to “fight’n words.”

In this era of “Complex RISCs” and “Streamlined CISCs” some argue that the terms RISC and CISC have

become technically meaningless. CISC proponents point out that many of the claimed innovations attributed to RISC—cache, multiregister sets, delayed branches, etc.—are not really new. In the absence of technical difference between RISC and CISC, the marketing definition applies: the Intel 80x86 and Motorola 68k family are CISC, anything else is a RISC.

The latest CISCs, such as the 80486 (Figure 1) and 68040 (Figure 2), strike

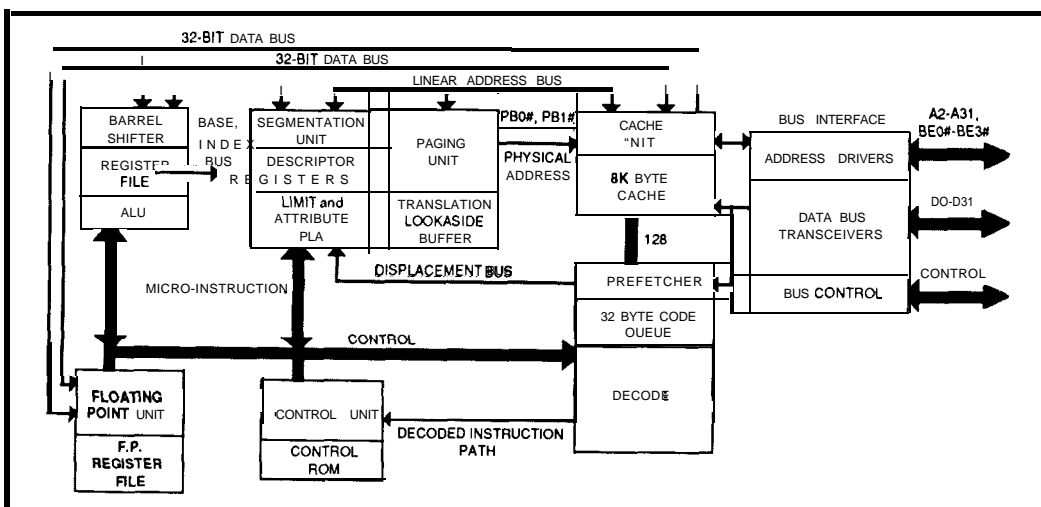


Figure 1—The 80486’s features include a built-in floating-point unit and virtual memory support.

fear in the hearts of the RISC advocates. Notably, the RISC Holy Grail of fewer clocks per instruction is challenged by these clever CISC designs, as shown in Figure 3.

The RISC advocates wish Intel and Motorola would quit making "good" CISCs, but they won't.

Most are starting to recognize that, technologically, the 'ISC wars may not matter much. Integer performance (where instruction set complexity presumably matters) for all the new chips is about the same and the main area for improvement is floating point. In the marketing sense of pitting the aging gunfighters against the new dudes, the 'ISC wars are still a big deal.

## TECHNOLOGY HEROES

The excitement over the architectural merits of each CPU is curious since the real heroes are the process, fabrication, and packaging technologists who come up with the recipe for manufacturing these ever faster, denser chips. While architects and designers battle over 2x improvements, those who actually manufacture the chips may deliver 10x.

The big news for the '90s is that the increases in chip density and speed will continue unchecked! The chip makers are prepared to challenge the formerly conventional wisdom that the limits of VLSI are right around the corner. Now, projections are that the latest technologies (such as x-ray lithography) will carry us into the next century before limits are reached.

In the '90s you can expect to see goodies like:

- 150-MHz 64-bit CPU "modules" containing multimillions of transistors.

- \*Viable, albeit "brute force" (not AI-based), voice recognition technology.

- 16-Mbit and (near the end of the decade) 64-Mbit DRAM technology.

The '90s—the third decade of the microprocessor—will be marked by a power shift that will impact the entire computer business. In the past, micro designers largely followed in the footsteps of earlier mini (VAX), mainframe (IBM), and supercomputer (CDC, Cray) pioneers. Today, having integrated all the pieces (CPU, cache, FPU, MMU) onto single chips, the next steps are relatively uncharted. From now on, it's up to the VLSI chip designers to lead the way to tomorrow's computer designs.

## BENCHMARKS LIE

It was generally agreed by all that benchmark hype has reached the point of ludicrousness. Each claimant proclaims honorable intentions, but if the other guy is going to lie, the temptation is to assume nice guys finish last.

An excellent presentation by Reinhold Weicker of Siemens (author of the Dhrystone benchmark) examined the laundry list of popular benchmarks: MIPS (Native, Peak, VAX), Whetstone, Linpack, Dhrystone, Lawrence Livermore Loops, Stanford Small Programs Benchmark Set, Quicksort, EDN, Spice, Sieve, Rhealstone, and SPEC.

Though widely used to "prove" the superiority of one chip or the other, these benchmarks, with the exception of SPEC, generally suffer from one or both of the following problems.

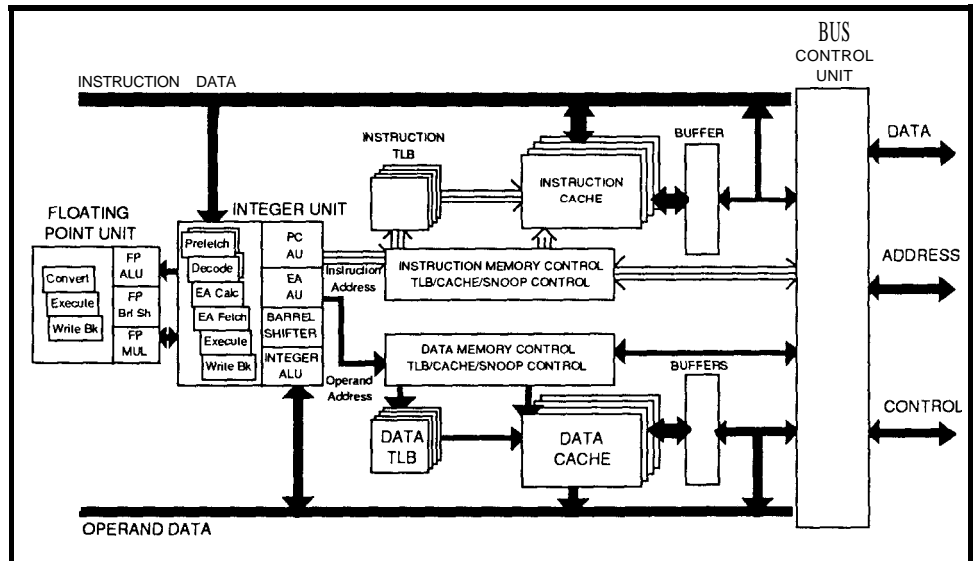


Figure 2 — The 68040 is the next step beyond the 68030.

**Cacheability**—Unfortunately, many of the most popular benchmarks (e.g., Whetstone and Dhrystone) are composed of small routines and data sets which entirely fit in the code/data caches of modern chips.

**Floating-Point Fixation**—Certain benchmarks (e.g., Whetstone, Linpack, LLLoops) effectively measure only one thing: floating point. This is fine for rocket scientists, but of little relevance in most commercial applications. System I/O performance (disk, screen, etc.) is never measured; doesn't it matter?

Despite these limitations, the benchmarks could be valuable if it weren't for the chicanery of the suppliers. As it stands, most of the claims are baloney. "MIPS" (Millions of Instructions Per Second) is the worst offender, truly being Meaningless Information Provided for Sales.

The other benchmark problems are related to "Apples vs. Oranges" match-ups of system design (clock rate, external cache size/speed), languages (Pascal vs. C) and compilers (supplier, optimization level, even benchmark-specific code generation). Together, these "fudge" factors (especially compiler quality) far outweigh any true chip performance difference.

SPEC—the System Performance Evaluation Cooperative—is a group including Sun, MIPS, HP, and others founded to come up with some real-world Unix benchmarks (C compiler, SPICE, etc.) that address the cacheability and floating-point fixation issues. Until then, buyer beware!

	<u>80386</u>	<u>MC68020</u>	<u>i80486 &amp; MC68040</u>
LOAD	4	6	1
STORE	2	6	1
REG-REG	2	2	1

Figure 3 — The distinction between RISC and CISC is starting to blur as can be seen by comparing the number of clock cycles used per instruction between classic CISC processors and their new-generation counterparts.

## PCs VS. WORKSTATIONS

Much has been said of the impending collision between PCs and workstations somewhere in the \$5–10k price range.

Hardware-wise, there is little differentiation between a fully loaded personal computer (IBM/clone, Macintosh) and the latest “personal” workstations from Sun, DEC, Tektronix, and Data General. Whether PC or workstation, the typical hot box has the following specs:

- High-Speed 32-bit CPU+FPU+Cache
- 4–16 MB RAM
- \* Various CRT options
- \* SCSI
- \* Ethernet
- \* Mouse

The claims of PC and workstation suppliers pretty much evolve into the RISC/CISC battle; for now, PCs use CISC ('86, 68k) while workstations use RISC (anything else).

The important differences between PCs and workstations—those that really matter in the upcoming desktop market-share war—aren't technical. They are:

\* Workstations run Unix, PCs usually don't.

• PCs are sold via retail stores and mail order, workstations aren't.

For now, despite technical and price/performance similarities, these differences mean PCs and workstations remain fundamentally different products. Who will be first to bridge the gap?

## SUPERCONTROLLERS

The Microprocessor Forum has a high-end focus. Needless to say, there was nary an 8-bit controller to be seen. However, conceding the existence of non-Unix, non-desktop applications, a session was held on the top-of-the-line chips intended for embedded applications. Naturally, these supercharged chips are a far cry from the 4-bit job in a toaster. They feature all the whizzies of their desktop CPU brethren, and more.. .

80960CA—This 32-bit chip probably is the fastest piece of silicon you can lay your hands on. [Editor's Note: For a more detailed look at the 80960, see "Silicon Update" in CIRCUIT

## TARDIS SYSTEMS

*Small-Computer Specialists*

**TARDIS 386/20**

- 20-MHz 80386
- 1 MB RAM (expandable to 16 MB)
- Shadow RAM Caching
- 65MB Hard Disk (28ms)
- 1.2MB 5.25" or 1.44MB 3.5" Floppy Disk
- 200-watt power supply
- 80387 socket
- Hercules-compatible mono graphics w/hi-res amber monitor
- KeyTronic Enhanced 101-key keyboard
- Scores 22.5 on Norton SI; 26 MHz on Landmark

*MS-DOS, OS/2, Xenix, and Novell Compatible*

**Call for pricing and delivery**

**TARDIS 286/12**

- 1-MHz 80286
- 512K RAM (expandable to 4MB)
- 32MB Hard Disk (28ms)
- 1.2MB 5.25" or 1.44MB 3.5" Floppy Disk
- 200-watt power supply
- 80287 socket
- Hercules-compatible mono graphics w/hi-res amber monitor
- KeyTronic Enhanced 101-key keyboard
- Scores 13.7 on Norton SI; 16 MHz on Landmark

*MS-DOS, OS/2, Xenix, and Novell Compatible*

**Call for pricing and delivery**

**TARDIS 386SX**

- 16-MHz 80386SX
- 1MB RAM (expandable to 2MB)
- 65MB Hard Disk (28ms)
- 1.2MB 5.25" or 1.44MB 3.5" Floppy Disk
- 200-watt power supply
- 80387SX socket
- Hercules-compatible mono graphics w/hi-res amber monitor
- KeyTronic Enhanced 101-key keyboard
- Scores 18.0 on Norton SI; 20 MHz on Landmark

*MSDOS, OS/2, Xenix, and Novell Compatible*

**Call for pricing and delivery**

*MS-DOS 4.1 included with all systems. All TARDIS SYSTEMS computers come with our fast service, 6-month parts and labor warranty, and exclusive TARDIS Upgrade Guarantee.*

**Call 505/662-9401 or circle 155 on the Reader Service Card** for complete custom system pricing and availability.

**TARDIS SYSTEMS**  
945 San Ildefonso — Suite 15 • Los Alamos, NM 87544

CELLAR INK #11.] Intel calls it "SuperScalar," which boils down to executing multiple instructions in a single clock.

**AMD 29000**—The desktop heritage of this "retro-marketed" Unix CPU shows, but shouldn't detract from the good points: 3-bus (instruction, data, address) architecture, register banks/stack cache, branch prediction.

**MC68332**—Essentially a single-chip 68k, outside program ROM/EPROM is still required. The ALU is based on the 32-bit '020, but the external data bus is 16 bits. Includes 2K bytes RAM, glue logic (chip selects), serial I/O, fancy X-channel timer, and a clever clock/power control scheme. What took them so long?

**VLSI 86C010**—This CPU owes its heritage to Acorn Computer (UK). Despite being originally designed for a desktop machine, the 32-bit chip has a unique feature desired in control applications: low cost. Not only is this truly reduced RISC tiny (27,000 transistors), but it is also designed to hook with "slow" outside memory chips (i.e., 80-ns DRAMs), reducing system cost.

The good news is these puppies are quite fast (claimed performance in marketspeak of 10 to 100 MIPS). The bad news is that most are expensive. Also, many still carry unnecessary desktop/Unix architectural baggage (i.e., virtual memory, MMU, cache strategy, etc.) that may help few, and actually hinder many, control applications.

For now, the SuperControllers are confined to the more expensive and esoteric end of the application spectrum. Inevitably they, and future variants, will begin to

migrate into broader markets. Remember, today's "high end" is tomorrow's "low end."

## RIDING IT OUT

Like a 7.1, the earthshaking chips of the '90s aren't the technological BIG ONE. Perhaps AI, superconductors, cold fusion, or another "fundamental" breakthrough will render them insignificant. Nevertheless, I'm sure we can all find good uses for a few million extra transistors.

Now, if they could only figure out a way to make the darn things easier to program.. ❖

*Microprocessor Forum and Microprocessor Report*

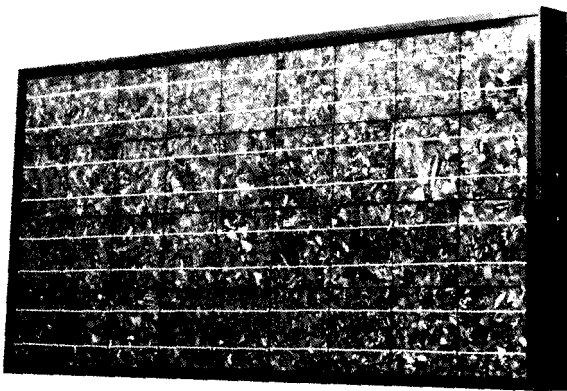
MicroDesign Resources, Inc.  
550 California Ave., Suite 320  
Palo Alto, CA 94306  
(415) 494-3718

*Tom Cantrell holds an M.B.A. from UCLA. He owns Microfuture, Inc., and has been in Silicon Valley for 10 years involved in chip, board, and system design and marketing.*

## IRS

222 Very Useful  
223 Moderately Useful  
224 Not Useful

## ELECTRICITY from SUNLIGHT



Photovoltaics for remote and online power applications. Reliable, renewable electricity from sunlight. Where there is a battery to be charged, there is a place for photovoltaics.

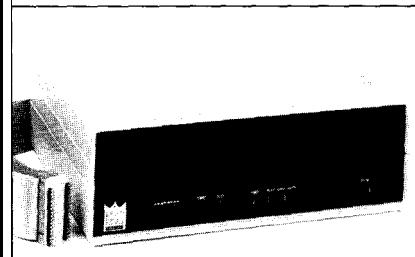
## SUNNYSIDE SOLAR

RD4 Box 808 Coreen River Road  
Brattleboro, Vermont 05301

802-257-1482

or circle 153 on the Reader Service Card

## STOMP OUT EPROM MADNESS



The PROM KING emulates EPROMS, saving both time and money during your development cycle. Programmable in seconds via your PC printer port or any computer RS232 port. It can emulate most 27xxx devices.

- 8K-8M bit devices
- High speed download: -Universal RS232 -PC printer port
- Menu driven software
- Battery backup
- 8-256 bit downloads
- Easily expandable: -4 EPROMS per unit -Up to 8 units
- Also programs like a real EPROM

\$599 for 150nS units with 256K bits  
Ask for pricing of other options

Made in USA by.

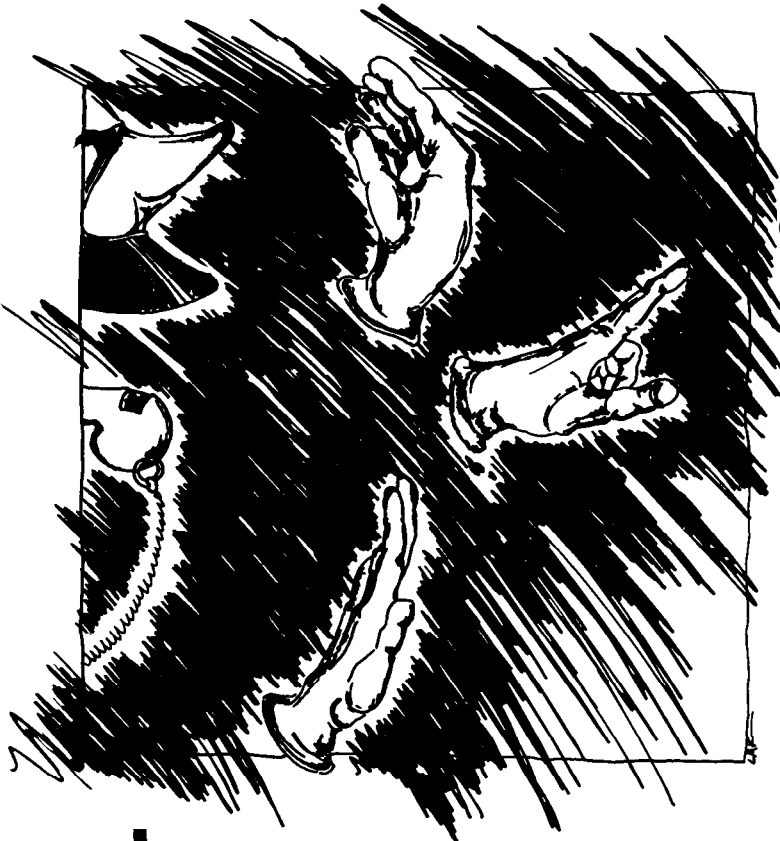
TRAXEL LABS INC.  
BOX 239 • RONKONKOMA, NY • 11779  
516-737-5147



# Memory Management on the HD64180

SOFTWARE  
BY  
DESIGN

Jack Ganssle



It seems that most of CIRCUIT CELLAR INK's readers develop embedded code on 8- and 16-bit microprocessors. However, many of the most common 8-bit architectures are based on rather ancient designs. The venerable Z80 is nearly 15 years old, yet continues to dominate the 8-bit market.

While B-bit technology has remained fairly stable, applications have not. In the '70s, a microprocessor-based product just had to offer bare functionality to be a success; in these more competitive '90s, creeping featurism abounds. Products seem to have to offer virtually every conceivable function, since management and the customer see the increased performance as requiring "only software change."

Developing solutions to these more complex problems requires bigger programs and the corresponding extra address space. In addition, the cost of developing software is forcing most companies to use high-level languages. While most proponents claim C is only about 15-25% less efficient than assembly, extra memory must indeed be available. Finally, the decreasing cost of memory makes more complex problems manageable; as memory costs decrease, you can be sure that someone will find a need for even more!

Traditionally, 16-bit computers have been touted as the solution for memory woes. Certainly the 68000, with a huge linear address space, offers a great way to take advantage of large DRAMs. Those of us working in the controller world frequently can't use these more powerful CPUs; often we want to simply upgrade an old application based on an B-bit CPU.

Hitachi introduced the HD64180 as a high-integration Z80 replacement. One of its most important features is the memory management unit. In my informal surveys, I have found that while the on-board I/O is a requirement of many applications, the MMU is the primary decision factor for many engineers who select the chip.

The chip offers a memory management unit designed to let B-bit applications break through the artificial bounds of a 64K address space. It doesn't automatically handle the MMU; extremely careful programming is required to access extended memory.

This, then, is the problem you face when electing to use an B-bit CPU cum MMU rather than a processor with an intrinsically large address space. It can be tedious and troublesome to take advantage of the MMU. In most cases, it is impossible to come up with a scheme to casually hop all over memory; crucial to the successful use of the MMU is a partitioning of the application to work comfortably within the MMU's restrictions.

## LOGICAL VS. PHYSICAL

It is important to understand the concept of logical versus physical addresses. Physical refers to the processor's actual address range: 1 MB on the HD64180. Physical addresses are applied by the hardware to the memory devices. Logical addresses are issued by your program. Again, on the HD64180 the logical addresses are limited to the same 16 bits that the Z80 used for addressing; logical addresses can never exceed 64K. The Z80 has 64K of logical and physical address space; the two are one and the same. The HD64180 gives you the same 64K of logical space, but 1 MB of physical space. The function of the MMU is to translate the logical addresses used within your program to physical addresses that are sent to memory.

On the HD64180, the MMU is a hardware device built onto the processor's silicon. Every non-DMA memory address is translated by the MMU from 16 to 20 bits.

The HD64180's MMU uses three internal control registers. In keeping with the chip's design philosophy, on reset the MMU gives a straight logical-to-physical mapping, simulating the Z80 and, of course, limiting the address space to 64K.

### MMU REGISTERS

You can divide the HD64180's logical address space into one, two, or three areas. The logical space itself is unaltered; even when divided it is still a contiguous 64K. This process defines how each of the (up to three) areas exists in physical memory.

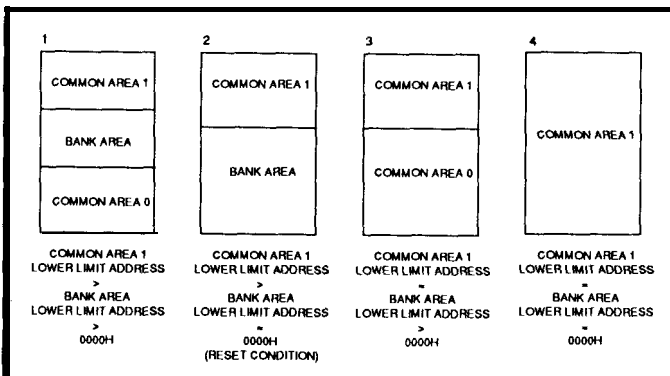


Figure 1—The four possible configurations of the HD64180 MMU allow one, two, or three logical partitions.

Figure 1 shows all of the possible configurations. Note that each is associated with a rule (e.g., Common Area 1 = Bank Area = 0). This rule is a condition programmed into the HD64180's CBAR register, and is what you use to define the logical limits of the address space.

CBAR is an 8-bit I/O port (though most users refer to it as a register) that can be accessed by the processor's OUT0 and IN0 instructions. The lower four bits specify the starting address of the bank area, and the upper four give the start of common 1. These bits determine the upper four bits of the address. If CBAR were A8, then the base area starts at 8000 logical, and common 1 starts at A000.

Common 0, if it exists, always starts at logical 0000 and runs up to the bank area. The bank area then runs to the start of common 1.

Therefore, you can always understand the logical address space by examining the contents of CBAR by itself. No other information is needed. Figure 1 is a quick "cheat sheet" giving the possible configurations.

The logical address is only part of the problem. How does logical space get mapped to physical? Two other registers (actually, I/O ports) provide the rest of the answer.

BBR (the Base Bank Register) specifies the starting physical address of the base area (remember, the logical start was given in CBAR). CBR (Common Bank Register) provides the same information for common 1. Both of these specify the upper 8 bits of the 20-bit physical address.

A simple formula gives the translation from logical to physical address for the bank area:

$$\text{Physical} = \text{Logical} + (\text{BBR} * 4096)$$

The same formula gives Common 1:

$$\text{Physical} = \text{Logical} + (\text{CBR} * 4096)$$



BBR and CBR gives the upper eight address bit only—hence the 4096 multiplier. The lower 12 bits come from the logical address. Thus, the translation only really affects the upper 8 bits; the lower 12 physical bits are always identical to the lower 12 logical.

CBAR	CBR	BBR	COMMON 0	BANK	COMMON 1
F0	00	00		00000-0EFFF	0F000-0FFFF
F0	01	00		00000-0EFFF	10000-10FFF
10	20	00		00000-00FFF	21000-30000
21	10	0F	00000-01FFF	10000-11FFF	12000-1DFFF
C6	10	00	00000-05FFF	06000-0BFFF	1C000-1FFFF

Table 1—Some examples of how the logical address space can be mapped onto the physical address spaces show the flexibility of the HD64180's MMU.

On reset, the HD64180 sets CBAR to F0, and CBR=BBR=0. This maps logical to physical exactly, with no translation; the bank area starts at logical 0000 and common 1 at F000 (since CBAR=F0), the bank area physically starts at 0000 (BBR=0), as does common 1 (CBR=0). If logical address 1000 is generated, the MMU allocates this to the bank area (CBAR=F0; 1000 is less than the start of common 1 at F000), and adds the physical base of bank to it (0000), giving a translated address of 01000. Similarly, logical F800 is in common 1, and is translated by adding CBR (0), yielding 0F800. Table 1 shows some more examples.

By now it is pretty clear that while the MMU is not all that difficult to use, it does require a lot of thought. Admittedly, with only a few instructions you can set up

any mapping. This does not mean that a casual approach will yield a 1-MB linear address space, since every time the program transfers control over a map boundary the mapping registers must be reset manually. That is, jumps and calls within a large program potentially require remapping.

Few HD64180 users successfully use the MMU to write huge programs without completely restructuring the program's design. The key to the MMU is careful partitioning of the problem.

In most embedded systems, either the program or the data structures are so large that they require the MMU's services. In some pathological cases both might be a problem. Examine the problem first; design a solution based on the program's real needs.

## THE DATA PROBLEM

Accessing large data structures can be an almost intractable task. When writing an assembly program, it is common to search and initialize arrays by pointing HL or the index registers to the base of the variable and incrementally manipulating each location. If the array cannot fit in the processor's logical address space, then the MMU can indeed be used to access it, but whenever the memory pointer is incremented, a test must be made to see if a remap is needed. This isn't hard, but it can be very slow.

If you are working with one or more large arrays, divide the map into two or three segments, and assign one of these to the array structure. Generally, another would be used to access the ROM, and the third to handle program temporaries (miscellaneous variables needed in RAM). The area of the logical map assigned to the array (say 4K) is then a virtual window into the data; at any time 4K of the data is available, and by changing the BBR or CBR value other 4K segments can be used.

It's usually a bad idea to **write** code that directly works on the array. Isolate the entire data structure from the application program by using a driver to access it. The hardware-dependent aspects of MMU management are then buried in one easily maintained subroutine.

Be very careful when handling interrupts with this (or any memory management) scheme. Since the interrupts are by their very nature asynchronous, the interrupt service routine can gain control with any of a number of maps set. If the ISR needs access to the data, some mechanism must ensure that the MMU is set properly. Be especially sure that the stack is restored before the IRET is executed.

## CODE MANAGEMENT

One of the biggest attractions of the HD64180's memory manager is the promise of using large programs. Yes, it is possible to write huge monolithic programs that crudely simulate a more or less linear address space via the MMU. However, this is not advisable unless using a high-level language that can automatically manage the details.

Every experienced programmer divides a complex program into many small functions. This top-down decomposition can be combined with careful grouping of the functions into modules that can share identical logical address space to minimize remapping.

The three-space memory model is the best for handling big programs. Common 0, located at physical address 00000, is never remapped. It contains the interrupt service routines and all other commonly invoked code. The system's "main loop" typically also resides here.

Common 1, which occupies the end of the logical address space, is mapped to the system's RAM, and holds all of the program's variables. It may be remapped, but only at great peril, since the stack will certainly be lost.

The base area, sandwiched between the two common blocks, is where most of the remapping activity takes place. Perhaps the bank area is only 4K long; functions can **be grouped** together until 4K worth has been accumulated, and this group compiled at the logical address assigned to the bank area. Each group is compiled at this same address, and then loaded into different physical segments. (Assembly programmers might use the PHASE and DEPHASE pseudo ops to create these virtual overlays).

A driver routine in common 0 is used to invoke each function; no calls are made between functions without going through the driver. It remaps the MMU as needed every time a function call is made, and then branches to the proper routine.

This approach has a profound benefit. New maps can be selected by reloading just the BBR register. Obviously, this is faster than resetting a complete, complex new map. Even better, the memory management unit is always in a good state; a partial map is never loaded and interrupts can be left on while remapping. A BBR value can be associated with each function, as well as a starting address within the bank area, making function invocation easy.

This is the approach taken in the real-time operating system I described some months ago. [Editor's Note: Jack Ganssle's two-part article "Writing a Real-Time Operating System" appeared in *CIRCUIT CELLAR INK* issues 7 and 8.1 The tasks are each loaded in the bank area. The context switcher, once all normal operating system activities are done, simply reloads BBR before vectoring to the next task.

This approach is by no means intended to be all inclusive; it somewhat precludes the use of both large RAM and ROM spaces. It does help organize big programs in a useful, maintainable fashion.

It seems silly that this tedium can't be taken over by a smart programmer's assistant. Some C compilers do support automatic management of the MMU. In assembly you're on your own. Softools (P.O. Box 2412, Columbia, MD 21045) sells an assembler/linker that lets you divide a program into modules that are all mapped as virtual overlays. The linker generates the code needed to drive the MMU whenever a "long" call is coded. The software also generates source-level debugging information so that an emulator can track the code in each of the different maps.

## SPECIFIC PERILS

The MMU is guaranteed to cause lots of problems if a few rules are not followed. Unfortunately, it seems that most programmers acquire this knowledge through bitter experience.

As I suggested above, it is usually a good idea to keep a small kernel mapped in low memory all of the time. Remap the MMU by calling this driver. Otherwise, your program might accidentally map itself out-the routine doing the remap could get lost when the MMU I/O load instruction is executed! While in certain rare instances this can be useful, it should generally be avoided.

Of course, if you call a routine in the kernel to do a remap, be sure the return address remains mapped in. This certainly applies to the stack as well!

It is crucial to carefully plan each and every load of the MMU registers. Since three registers are needed to define the complete memory configuration, the order in which the mapping registers are loaded is important. In other words, you might map your code out when simply trying to change data access areas. Sometimes CBAR should be loaded first; other times the BBR and CBR registers. In all cases, examine the code to ensure that at each step of the mapping process the program doesn't get mapped out.

A related and extremely common problem is setting the MMU registers with data loaded from RAM. If each value is read from **memory** and then output to the memory

manager, you'll frequently map out the RAM from which the mapping values are being read in one of the intermediate steps.

Finally, interrupt-driven systems must be carefully designed so the stack is always available, and so the interrupt service routines are always visible. Further, when in the process of loading the registers, an interrupt that comes when the MMU is partially set up can be a disaster. Usually it is a good idea to disable interrupts before changing the MMU parameters.

## SOME THINGS CHANGE

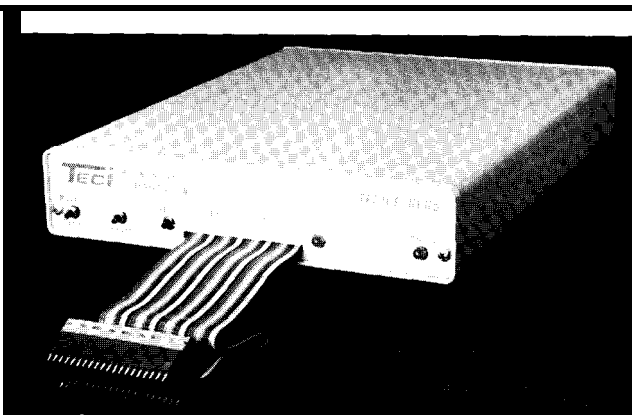
There are as many different ways of handling a memory management unit as there are programmers. All involve careful analysis of the problem, and selection of a memory configuration matched to the application. In many cases, a substantial change to the program's structure may be in order.+



*Jack Ganssle is president of Softaid, a vendor of micro-processor development tools. When not busy pushing electrons around, he sails up and down the East Coast on his 35-foot sloop.*

## IRS

225 Very Useful  
226 Moderately Useful  
227 Not Useful



### 68HC05 IN-CIRCUIT EMULATOR

The **TECICE-HC05** is a low-cost real time emulator for the Motorola 68HC05 family of single chip microcomputers. Any host computer with serial port-I and terminal emulation software can be used with **TECICE-HC05**. Base price is \$1,195.00. Complete development system software is available for MS-DOS computers including the Byte Craft Limited C6805 Code Development System which includes a 6805 C compiler with Integrated Development Environment.

**TECI** THE ENGINEERS  
COLLABORATIVE, INC.

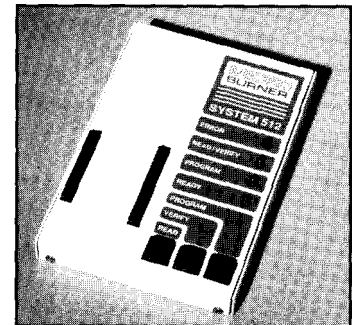
RR#3, BOX 8C . Barton, Vermont 05822  
Phone (802) 525-3458 . Fax (802) 525-3451

## E/EPROM & MICRO PROGRAMMER

Regular \$495

**Special Offer**  
**\$395**

VISA ACCEPTED



### MICRO-BURNER System 512

The **MICRO-BURNER** sets a new price performance standard in programmers adaptable to all single-chip micros with on-chip EPROM. Features include reverse EPROM detection/protection and 12 volts DC operation for portable applications. **30-day** money-back offer.

USA Distributor  
**QUALTEK Corp.**  
Lynwood, WA  
(206) 742-1777

CANADIAN Distributor  
**BARADINE Products Ltd.**  
North Vancouver, B.C.  
(604) 988-9853

## *Excerpts from the Circuit Cellar BBS*

The Circuit Cellar BBS  
300/1200/2400 bps  
24 hours/7 days a week  
(203) 871-1988  
Four incoming Lines  
Vernon, Connecticut

*In this installment of ConnecTime, we'll be covering how to detect leaky pipes, and how to make real short pulses. First, though, with the current popularity of fax machines and especially fax expansion boards for PCs, the issue of how to turn the computer on fast enough to catch the incoming call becomes important, as we find out in the first discussion.*

**Msg#:22732**

From: DAVID WILLIAMS, SR. To: STEVE CIARCIA

I've installed a fax board in my IBM AT clone. Keeping the computer on all the time is a pain. Do you know of a manufacturer who makes an auto-on switch that will boot up the computer when the fax line rings?

**Msg#:22739**

From: JEFF BACHIOCHI To: DAVID WILLIAMS, SR.

Considering the amount of time it takes to power-up a system and run an application program, most fax machines will give up long before you get anywhere near answering the phone. The best you could hope for would be a call-back within x-minutes!

**Msg#:22745**

From: NATHAN ENGLE To: DAVID WILLIAMS, SR.

I have to agree with Jeff about fax machines giving up before your machine gets booted up. It's a race you're practically guaranteed to lose, and I can't really see the fax board manufacturer feeling too happy about it either. You're not leaving very much time for their card to get to the call once you're done booting.

I used to feel very much like you did about leaving the PC turned on all the time; I have been doing it for the past year, though, and I've noticed absolutely no increase in my bills. The power supply in an XT is usually 130 watts; about what a light bulb takes, right?

**Msg#:22789**

From: JEFF BACHIOCHI To: NATHAN ENGLE

The monitor can be switched off with the machine still running. It is the start-up and thermal cycling that deteriorates compo-

nents at a faster rate than leaving them on full time. Remember the power supply rating is a maximum, your system may in fact draw much less (pennies a day).

**Msg#:22837**

From: NATHAN ENGLE To: JEFF BACHIOCHI

That's a good point; I think the way my XT is loaded I come in near the upper end for power usage.

I have been leaving my monitor on since I have one of those ATI VGA cards that has a built-in screen saver function. This is available in shareware for people with other kinds of displays.

**Msg#:24131**

From: DAVID SMITH To: NATHAN ENGLE

On the subject of leaving a monitor on for extended periods of time, I don't think it would be advisable since the screen saver function just shuts off the video but does not turn off the filament in the picture tube. When a filament or heater in a CRT wears out in a TV set they rejuvenate it; but it can only be rejuvenated once or twice. Jameco does sell a fax board that will turn on your computer (Niche Tek, FAX96), but I agree with leaving your computer on (but not the monitor).

**Msg#:24276**

From: JOHN MUCHOW To: DAVID SMITH

Can the monitor be at all damaged by having the constant input from the system while it's off. I've considered turning off my PCS Ultrasync off during overnight jobs, but have been scared to for that reason. I do use PC Mag's VGA Dimmer, though.

**Msg#:24520**

From: NEIL CHERRY To: JOHN MUCHOW

My understanding is that when off you'll do no damage. But it may be possible to "blank" the screen. On Hercules graphics cards you simply switch a bit and the screen goes off. I believe that the EGA and VGA both have the same capabilities. This would be equivalent to turning off the monitor.

*While many of us wouldn't think electronics had any application in the plumbing field, the following thread shows how wrong that thinking is.*

**Msg#:20774**

From: DANIEL L. MILLER To: PAUL HITCHCOCK

Is there a way to detect water leaks from pipes? Say about a 150-foot run of pipe through the attic. I want to automatically shut off the water pump if a leak is detected. Surely someone in the refinery industry has evolved some solutions to this type of problem.

**Msg#:20797**

From: ED NISLEY To: DANIEL L. MILLER

How much of a leak?

If you're looking for a torrent, measuring the pressure drop at both ends of the pipe should give you a pretty good idea. You'd have to calibrate the setup for the normal pressure drop at whatever the usual flow rate is, but that's a simple matter of software.

If you're looking for a trickle, that's a different problem entirely (and the one most likely to occur, too!). Although I don't think it's a standard technique, how about running a length of cotton-insulated two-conductor cable around the pipe (a spiral wrap with a few turns per foot) and measuring the resistance between the conductors? Put a resistor on the far end and you'd have built-in detection for open circuits, too!

The only pipe leaks I've had to contend with were pretty obvious, so the only detector technology I needed was an eyeball. Oh well...

**Msg#:20876**

From: PAUL HITCHCOCK To: DANIEL L. MILLER

Well, I don't know about detecting leaks in long lengths of pipe (especially cold water pipes which sweat), but near joints I have used two coils of wire, separated by a dry cotton pad which I had previously soak in salt water before drying. For a large leak (which is what I was looking for) the resistance change (infinite to finite) was fairly dramatic as I recall. It's been a number of years since I tried this trick, and I don't remember exactly how much wire I used.

**Msg#:20987**

From: DANIEL L. MILLER To: PAUL HITCHCOCK

Thank you. My air conditioning system uses water to cool the in-house heat pump coils. The water source is a source well and the drain is an injection well. It pumps 32 gpm and I tend to get nervous thinking about that circulating through PVC pipe that is too small in diameter and shudders whenever the units switch on and off,

**Msg#:21015**

From: ED NISLEY To: DANIEL L. MILLER

Run, do not walk, to your nearby plumbing supply store and get some straps to secure that pipe! It will eventually crack the fittings (not the glued connections) or the pipe sections that are taking the most strain and all hell will break loose.

If it's shaking that much, you may need a standpipe (closed at the top) to act as a shock absorber-the same sort of thing plumbers delight in putting inside a wall at the end of a long run. After a while the air is absorbed, the standpipe fills up, and you're back to a colossal water hammer banging in your walls.. you have to rip the wall apart to drain the standpipe unless they did a very good job of sloping the pipes.

**Msg#:21053**

From: PAUL HITCHCOCK To: DANIEL L. MILLER

Dan, I read Ed's reply and completely agree: STRAP that pipe! At the flow rate you're talking about, a sustained waterhammer incident can put immense stress on both the pipe itself as well as the joints. (The thought of 32 gallons flowing into the attic every minute makes me shudder!) The stand pipe is a good idea as well and you might even consider encasing the PVC within a length of steel gas pipe.

Without detailed knowledge of your plumbing, my best idea is to install flowmeters at each end of the pipe (both downstream from the standpipe) and hook up a controller that shuts off the pump whenever a certain differential flow rate occurs.

Omega Engineering (One Omega Drive, Box 4047, Stamford, CT 06907-0047) carries an extensive line of flowmeters. I understand they are \*quite\* expensive, however. Maybe somebody can point you in a less costly direction.

**Msg#:21189**

From: DANIEL L. MILLER To: ED NISLEY

Thanks. As soon as I moved in I did what you said and added antiwaterhammer devices. These are small tubes charged with nitrogen with a neoprene bladder that doesn't deflate with time and adds capacitance (i.e., they work similar to the filter capacitor in a power supply and are much smaller than standpipes). I don't have sustained knock but do have turn-on/turn-off transients. I think I'm going to call a plumber and put the main connection pipe outside underground where it belongs and get it out of my attic! Thanks much for help.

**Msg#:21719**

From: FOSTER SCHUCKER To: DANIEL L. MILLER

In the August 1989 issue of Industrial Equipment News, there is an article about the Raychem leak detection system. You wrap this fancy cable around or along your pipe and it tells you when and where it leaks. Contact: Raychem Corp., 300 Constitution Dr., Menlo Park, CA 94025. It's called TraceTek 500. Good luck!

There are numerous methods that can be used to generate individual pulses on the order of several microseconds and longer, but very short pulses can be difficult to come by, as we see here.

**Msg#:20846**

From: TOM CARTER To: ALL USERS

I would like to know a good way to generate a pulse under 100 ns wide using easy-to-get TTL or CMOS or whatever. A pulse of 50 ns or shorter would be best. Thank you for any help.

**Msg#:20866**

From: NATHAN ENGLE To: TOM CARTER

Do you want to get individual pulses that are that wide? Or asking another way, would a 5-MHz waveform do?

If you've got a really short trigger pulse you want to stretch, then you can use an LS221 or any of the "multivibrator" class of chips. These chips should really just be called "pulse stretchers."

If you need single pulses, I think the '221 may do that for you the best. They work kind of like 555 timer chips: you get to play around with an external capacitor and resistor to get the timing you need. Then you trigger your pulse with an edge transition.

Any help?

**Msg#:20890**

From: TOM CARTER To: NATHAN ENGLE

I want single pulses that are triggered by a rising or falling edge (rising only or falling only). I have seen circuits like I need using a 74121 with a 220-pF capacitor and 2.2-kΩ resistor which generate 300-ns pulses. I guess I need to know how small I can make the values and still get a good pulse. Under 50 ns is what I would really like.

**Msg#:20910**

From: ED NISLEY To: TOM CARTER

If you're really interested in good, clean, tidy pulses under 50 ns wide, you need to be careful about which logic family you're using. The rise time and fall time (which are not usually the same) are a significant part of your total pulse width, so you'll have to pick a set of gates that work in your application.

You can use a NAND gate to generate quite nice pulses if you set up a deliberate race condition: feed both inputs from the same source, but one input goes through an odd number of inverters. When the source goes from high to low, you get a negative output pulse as long as it takes the signal to propagate through the inverters. Other logic gates will give you different triggering conditions if you need them.

You can trim down a pulse by inserting pairs of inverters; each pair will shave a few nanoseconds off the leading and trailing edges of the incoming pulse.

The only catch with all this is sensitivity to things like temperature, supply voltage, the phase of the moon, and so forth. You will need a fine oscilloscope to verify what's happening; the more bandwidth the better, because you're interested in the width of a pulse that's almost not there!

---

*The Circuit Cellar BBS runs on a 10-MHz Micromint OEM-286 IBM PC/AT-compatible computer using the multiline version of The Bread Board System (TBBS 2.1M) and currently has four modems connected. We invite you to call and exchange ideas with other Circuit Cellar readers. It is available 24 hours a day and can be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, and either 300, 1200, or 2400 bps.*

**IRS**

228 Very Useful  
229 Moderately Useful  
230 Not Useful

## The Ciarcia Design Works

Steve Ciarcia has assembled a team of engineers, designers, and programmers to produce the products that have made Circuit Cellar famous. Now you can put the Ciarcia team to work for you.

Steve Ciarcia and his staff have designed products ranging from communications and networking components to multiprocessing computers. Current capabilities include every phase of design and production, from initial concept through packaging of the finished product.

Whether you need an on-time solution for a unique problem, complete support for a startup venture, or experienced design consulting for a Fortune 500 company, the Ciarcia Design Works stand ready to work with you.

Remember... a Ciarcia design works!

**Call 203/875-2199 Fax 203/872-2204**

## SOFTWARE and BBS AVAILABLE on DISK

### Software on Disk

Software for the articles in this issue of Circuit Cellar INK may be downloaded free of charge from the Circuit Cellar BBS. For those unable to download files, they are also available on one **360K**, 5.25" IBM PC-format disk for only \$12.

### Circuit Cellar BBS on Disk

Every month, hundreds of information-filled messages are posted on the Circuit Cellar BBS by people from all walks of life. For those who can't log on as often as they'd like, the text of the public message areas is available on disk in two-month installments. Each installment comes on three **360K**, 5.25" IBM PC-format disks and costs just \$15. The installment for this issue of INK (February/March 1990) includes all public messages posted during November and December, 1989.

To order either Software on Disk or Circuit Cellar BBS on Disk, send check or money order to:

**Circuit Cellar INK — Software (or BBS) on Disk**

P.O. Box 772, Vernon, CT **06066**

or use your **MasterCard** or Visa and call (203) 875-2199. Be sure to specify the issue number of each disk you order.

# An Analog State of Mind

**T**he last time I wrote one of these pages, I was talking about fixing a high-tech German car using a two by four. This time around, you're going to read a short lesson on digital overkill. Now don't get me wrong: I'm still in favor of computer solutions to many problems, and the idea of high-powered computing toys still turns me on. I've just spent a lot of time lately hearing people moan about problems they've created for themselves by using too much technology for the problem, and it's starting to get on my nerves.

Let's start with the obvious. I've never made any secret of the fact that I much prefer working with digital circuits to trying to figure out analog. As far as I'm concerned, the world would be a much easier place to work if I didn't have to worry about analog at all. I'm not alone here—most digital electronics engineers have fantasies about a purely digital world. Most of us realize, though, that at some point in the process, you're going to have to bite the analog bullet if you want a working, real-world application. The problem is with the people who say that they're going to design an analog section (under protest, of course) but who try to cut intellectual corners by keeping their digital mindset.

One of the most frequent mistakes I see is caused by folks holding on to a mistaken notion of accuracy. When we're talking about accuracy in a digital sense, I've found that most engineers run out of interest long before they run out of bits. Since all of those oh-so-accurate bits are there, computer designers have gotten used to the idea that you run your answer out to as many digits as you have available. Leaving aside the question of whether anyone really needs to (for example) balance their checkbook down to a millionth of a cent, the result of all this has been a redefinition of how much accuracy is acceptable. Back in the days of the slide rule, when dinosaurs roamed the earth, two digits to the right of the decimal point was acceptable for most purposes, and four places meant that you had done

some serious work. Now that a four-dollar calculator can better that accuracy for simple function, some people forget that there are a lot of instances where you just can't demand accuracy down to a gnat's eyelash.

I've had to learn to accept some "creative looseness" when it comes to dealing with analog signals. When I let myself get into an analog state of mind, I realized that there were places where the quest for accuracy turned around to bite me. If I'm working on a climate-control automation system for my house, I can demand that the input section be accurate down to a tenth of a degree, and I can set action points to that accuracy. The sensors are available, and the microcontrollers can certainly handle the bits. The trouble is, the extra accuracy requires three extra weeks of work, can lead to unpredictable operation of the system, and just isn't necessary. If I were to press for the added accuracy, I'd just be rewarded with aggravation and failure for my efforts. The fact is that single-degree accuracy is finer than most humans can discriminate, so it's certainly accurate enough for the job.

It's all a matter of using the right tool for the job. I don't use a hammer when I want to tighten a bolt, and I don't try to make the analog part of a project work in ways that are contrary to reality. *CIRCUIT CELLAR INK* concentrates on the digital end of the process, but there's no way that any engineer can hope to build successful applications that interface with the real world without knowing the analog. Taking the time to learn makes a hacker into a real engineer, and keeps you from reaching for a pair of pliers when a hammer is what you need.

