

CIRCUIT CELLAR **I N K**[®]

THE COMPUTER APPLICATIONS JOURNAL

DSP
Architecture
Comparison

DSP Basics

Working with
Analog Design

Digital Signal Processing

SPECIAL SECTION:
Compilers for the 8051

February/March 1991 — Issue 19

\$3.95 U.S.
\$4.95 Canada



What a Difference

EDITOR'S INK

Curtis Franklin, Jr.

In **CIRCUIT CELLAR INK #18**, my editorial was about the changes and trends I saw at the Embedded Systems Programming Conference. Since then, I've seen a lot of new technology, including what seemed like several Sargans of new computers at COMDEX in Las Vegas. On the whole, I'm impressed by a lot of the engineering I've seen.

There were a few major trends that ran through most of the engineering and marketing: GUIs, laptop computers, networks, and (relatively) inexpensive personal workstations. All the trends point in one direction: Personal computers are becoming more personal every day.

Graphical User Interfaces (GUIs) are supposed to make computers much easier to use. Instead of having to remember the different commands and software switches, you just have to remember what each of those blasted little hieroglyphics means. Now, we use Macintosh computers to put **CIRCUIT CELLAR INK** together, and I'm trying very hard to get Windows to run on the 80286 machine at home, but I'm still not convinced that every icon is worth a thousand words. Part of the problem is maturity (the system's, not mine). MS-DOS, CP/M, Unix, and most of the other "classic" small-computer operating systems have had many years to develop utility programs, secret tips, and masterful techniques. The Mac has much of the *arcana* required to make a comfortable operating system, but it just doesn't feel like a computer. Whether I like it or not, windows and other graphical elements are going to show up in more applications, regardless of the underlying system. I just wish that someone would teach the interface designers a bit about color theory...

Whatever the operating system, it appears likely that many folks will be running it on a portable "laptop" computer. Even people who never plan to take the computer off their desks are buying the little machines because they don't look like computers. Personally, I rather like the look of a desk covered with several computers, but my interior design aesthetic is not shared by all. Having said that, I do think that the laptop trend will continue to grow. I feel that way because I simply love my laptop computer. I have a Toshiba T1000 that is an important part of my work routine. Quite apart from being a useful tool, it's the epitome of a nonthreatening computer. I react to it almost more as a pet than as a machine, and that's the sort of reaction that sells lots of laptops to computer-phobic man-

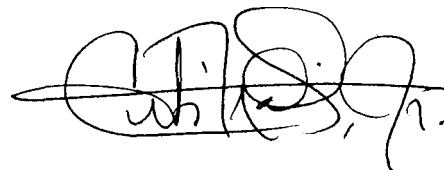
agers. Whether you program, engineer, or write, I recommend a laptop computer-as your second machine. If you're a technical professional, you won't want to miss the sort of power that you'll soon be able to put on your desktop.

COMDEX featured the (by now) standard array of 80386- and 80486-based computers. Enhanced video, big fast disks (BFDs), and network connections with enough bandwidth to download a Peterbilt were de rigueur, but the real new has nothing to do with these. The big news was SPARC. Sun designed a RISC processor and gave away the design, hoping that it would become a standard. If the early machines shown at COMDEX are any indication, they may get their wish. Just as important, many people are pushing SPARC as an embedded control engine. We plan to have more on that in a future issue.

We've claimed, in this magazine, that the microcomputer revolution is over, won by the power and convenience of embedded controllers. It remains for more people to consciously use computers. IBM, Apple, and Tandy are all making specific plans for this to happen, but in a real way, all of the trends are moving events in the direction of greater use and acceptance. It's good to know that **CIRCUIT CELLAR INK** readers will be leading the way with applications that bring microprocessing power to more people.

...FOR SOMETHING COMPLETELY DIFFERENT

There's no better way to understand how and why a computer works than to put it together yourself. If you know someone who would like to roll up their sleeves and get into a computer, you might try giving them a copy of a "How to Build a PC/AT Clone" video. That's right, the same medium that shows folks how to bathe their cat and firm their tushie can now teach them how to build a clone AT. Give JVF Distributors a call at (415) 4884179 and ask for more details. The existence of a video like this says a lot about how far we've come towards accepting high-tech help with our daily chores.



FOUNDER/
EDITORIAL DIRECTOR
Steve Ciarcia

PUBLISHER
Daniel Rodrigues

EDITOR-IN-CHIEF
Curtis Franklin, Jr.

MANAGING
EDITOR
Ken Davidson

PUBLISHING
CONSULTANT
John Hayes

ENGINEERING STAFF
Jeff Bachiochi
Edward Nisley

CONTRIBUTING
EDITORS
Thomas Cantrell
Christopher Ciarcia

NEW PRODUCTS
EDITOR
Harv Weiner

CONSULTING
EDITORS
Mark Dahmke
Larry Loeb

CIRCULATION
COORDINATOR
Rose Manse/la

CIRCULATION
CONSULTANT
Gregory Spitzfaden

ART & PRODUCTION
DIRECTOR
Tricia Fabish

PRODUCTION
ARTIST/ILLUSTRATOR
Lisa Ferry

BUSINESS
MANAGER
Jeannette Walters

ADVERTISING
COORDINATOR
Dan Gorsky

STAFF RESEARCHERS

Northeast
Eric Albert
William Curlew
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...

FEATURES

12 DSP Architectures for Signal Processing Applications *Matching the Tools to the Job*

by Bill Schweber

Differences in DSP architectures have a major impact on application suitability. Improve your DSP IQ with a tutorial on the architectural differences.

22 A PC Stopwatch *Improved Timing for Acquisition and Control*

by David P. Schulze

Are PC timing limitations standing between you and a successful application? Here's a way to beat the PC timing blues.

24 Digital Image Processing *Software-based Digital Signal Processing*

by Chris Ciarcia

You don't need hardware for DSP. This hands-on software article shows software techniques for digital image enhancement.

38 Mini-DSP *A Digital Signal Processor Experimentation Unit*

by Steven E. Reyer

There's no better way to learn DSP techniques than by diving into a project. This Circuit Cellar INK Design Contest winner is a perfect first step.

46 Analog Circuit Design *Stripping Away the Mystery for Digital Designers*

by Mark E. Nurczyk

It's an analog world-and this article shows you the top techniques for designing circuits for this very real world.

SPECIAL SECTION: Compilers for the 8051

58 Oh Say, Can You C? *Circuit Cellar INK Evaluates Three C Compilers for the 8051*

by M. Scoff Martin, Tim McDonough, & Curtis Franklin, Jr.

Circuit Cellar INK looks at three PC-based cross compilers for the Intel 8051 family.

69 High-Level Languages for Microcontrollers *Don't Believe the Hype*

by Ed Nisley

72 Using High-Level Languages on Embedded Controllers

by Ken Davidson

Are high-level languages for microcontrollers truly the greatest invention since sliced bread? Circuit Cellar INK looks at the issue from two different directions.

DEPARTMENTS

1 Editor's INK
What a Difference
by **Curtis Franklin, Jr.**

1
5 Reader's INK-Letters to the Editor

8 NEW Product News

74 Firmware Furnace
It's Just You and the CPU: Intel 80x86 Instruction Timings
by Ed **Nisley**

82 From the Bench
Multidrop A/D and D/A Network
Using your PC's Printer Port and Four-Conductor Phone Cable
by **Jeff Bachiochi**

93 Silicon Update
Hot Chips In The Summertime
A Report From "Hot Chips II"
by Tam **Cantrell**

100 Practical Algorithms
Making Hash
A Table Built for Speed
by **Scott Robed Ladd**

103 Domestic Automation
CEBus Gets Physical
The Standard Takes Two More Steps to Maturity
by **Ken Davidson**

105 **ConnectTime**—Excerpts from the Circuit Cellar BBS
Conducted by Ken Davidson

112 Steve's Own INK
The Sophomore Slump
by Steve **Ciarcia**

97 Advertiser's Index

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
Omaton Inc. All programs
and schematics in **Circuit**
Cellar INK have been cam-
fully reviewed to ensure that
their performance is in ac-
cordance with the **specifi-**
cations described, and **pro-**
grams are posted on the **Cir-**
cuit 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, be-
cause of the possible **vari-**
ation in the quality and con-
dition of materials and work-
manship of reader-as-
sembled project. Circuit
Cellar INK disclaims any re-
sponsibility for the safe and
proper function of **reader-**
assembled project based
upon or from plans, descrip-
tions, or **information** pub-
lished in Circuit Cellar INK.

CIRCUIT CELLAR INK (ISSN
08968985) is published bi-
monthly by Circuit Cellar In-
corporated, 4 Park **Street,**
Suite 20, Vernon, CT **06066**
(203) 875-2751. Second-
class postage paid at Ver-
non, CT and additional of-
fices. One-year (6 issues)
subscription rate U.S.A. and
possessions \$17.95, Canada/
Mexico \$21.95, all other
countries \$32.95. All subscrip-
tion orders payable in U.S.
funds only, via international
postal money order or check
drawn on U.S. bank. Direct
subscription orders to Circuit
Cellar INK, Subscriptions, P.O.,
Box **3050-C,** Southeastern,
PA 19398 or call (215) 630-
1914.

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

Entire contents **copyright**
© 1991 by Circuit Cellar In-
corporated. All rights re-
served. **Reproduction** of this
publication in whole or in
part without written consent
from Circuit Cellar Inc. is
prohibited.

Letters to the Editor

READER TO READER

This letter is a follow-up to a recent letter asking about receiving faxes by shortwave and displaying the information on an Amiga. There is now a commercial program called Amiga Video Terminal by Advanced Electronic Applications that can handle just about any form of visual radio communication short of fast-scan television. It includes image processing features for correcting individual lines or entire pictures and an AREXX port. The price is \$349.95 including hardware interface.

As long as I'm writing about Amiga products, there is now X-10-compatible software for the Amiga. Ami-X10 by Digital Dynamics works with the X-10 Powerhouse computer interface and provides for unlimited event schedules. It multitasks quite well under the Amiga's operating system and costs \$49.95.

The companies may be contacted at:
Advanced Electronic Applications
P.O. Box C2160
Lynnwood, WA 98036
(206) 775-7373

Digital Dynamics
739 Navy Street
Santa Monica, CA 90405
(213) 396-9771

Roy G. Clay III
New Orleans, LA

FUTURE TOPICS

This letter is in response to your request for input on future themes. I cast my vote for an issue on astronomy. In particular, the advent of robotics has made available **motors** of the type useful in building computerized optical telescope equatorial drives. Chargecoupled devices **are** available for capturing images in minutes as opposed to hours for photographic techniques.

A computerized tracking mechanism for keeping an optical telescope aligned for extended periods of time

when photographing deep-space objects would be a project I would be interested in tackling. Then there is the area of radio-telescopes with a wide variety of related topics which would make interesting projects.

I became interested in Steve's "Ciarcia's Circuit Cellar" when I ran across his three-part article for performing Mandelbrot calculations that appeared in BYTE. Except for the disappointing resolution available on VGA displays, I probably would have built one. I enjoy going through each issue of **CIRCUIT CELLAR INK** because of the ideas I see implemented using small single-board computers of modest capacities.

I am interested in computing and have recently begun to reacquaint myself with astronomy. I'm itching to "build something" that I will get a lot of use and pleasure from.

I hope my input has been of some help. I look forward to seeing some astronomy topics in future issues.

Thomas Duprex
Madison, NJ

*Astronomy is high on the interest list around **CIRCUIT CELLAR INK**, too. Unfortunately, the engineering staff has been too busy with other projects to get around to designing good astronomy applications. If any readers have put together good astronomy applications, we'd like to see them and, possibly, run them in the magazine. Send us your best!*

INDEX PLEA

I recently received my file of back issues of **CIRCUIT CELLAR INK**: Wow! Wonderful assortment.. .

How about an index for **CIRCUIT CELLAR INK**? Either on-disk or off-, as long as there's an index. One of my great disappointments in computer magazines all these years has been that they don't have good indexing. They deal with machines that are particularly good for this purpose and for coping with huge amounts of data.

Anyhow, I hope you'll give serious consideration to publishing an index to your magazine. Given that you have all the text files in magnetic media and that it's not very hard to search for keywords amongst these files...if

you were to store each page as a file coded by volume and page number, it would be trivial to index.

I saw the safety note in your magazine saying that a 100-mA current was dangerous. You're off by an order of magnitude: 7 mA can lock up your breathing and the rest follows. This can happen at relatively low voltages (around 40 V) when you're a bit sweaty. Ground fault interrupters cut out at 5 mA unbalanced, assuming that you may be about to stop breathing.

I was disappointed that, in the discussion on leaky pipes in the attic, no one suggested the KISS solution of putting a larger diameter pipe around the leaking pipe, then draining the larger pipe to a watched bucket. This is the method required for dealing with leaking underground storage tank situations. Of course, with the larger overpipe you need a sensor to see if the overpipe is leaking... and so on, ad infinitum. Oh well.

Mr. Premena
Boulder, CO

We have been trying to put a good index together since the end of CIRCUIT CELLAR INK's first year. The trouble is that an index (at least, a good index) is a rather more complex procedure than most people realize. We have all manuscripts and pages in electronic form, but they're not stored in any fashion that eases indexing. Nonetheless, we still hope to have a good index to CIRCUIT CELLAR INK available in the future.

CORRECTIONS AND ADDITIONS

Issue #18, "Using the Motorola MC68HC11"

Page 37, column 3, paragraph 2-The paragraph implies that there are 68HC11 instructions that can only use direct addressing mode. This isn't the case.

Also, the last sentence implies that the EEPROM may be relocated on all versions of the 68HC11. This is only true for the A2, E2, and F1 versions.

Issue #18, 'ONDI-The ON-Line Device Interface, Part 2'

The following items are available from

Cottage Resources Corp.
1405 Stevenson Dr., Ste. 3-672
Springfield, IL 62703
(217) 529-7679

1. ONDI printed circuit board \$42.00
2. DS2250 with socket \$45.95
3. Power transformer \$10.95



Complete

Paradigm has the complete solution for embedded system software development.

Compile

Turbo C++ or Microsoft C... a tough decision. Make your choice and rest easy because only Paradigm LOCATE has the ability to work with popular in-circuit emulators.

library and floating point initialization. LOCATE frees you to concentrate on the details of the application.

Locate

Paradigm LOCATE provides a full spectrum of options for controlling the locate process. Bind physical addresses to code and data, automatically handle initialized data or generate optional EPROM and documentation files. Intel 80186/188 users will appreciate how Paradigm LOCATE uniquely eliminates the hassles of memory chip select initialization. And Paradigm LOCATE is much faster than your current tools, locating large applications with full debug information in just seconds.

Debug

Without complete debugging information, getting a grip on a recalcitrant application can be no easy matter. Paradigm LOCATE is ahead of the pack with complete support for the award-winning Turbo Debugger. Those with significant investments

Turbo Debugger is a trademark of Borland International.

Relax

Your application is done in record time because you've developed the correct choice of software development tools. If you're still struggling, now you can experience the power, flexibility and completeness of Paradigm LOCATE.

Call or write us today for more information on state-of-the-art embedded system solutions from Paradigm for Intel 80x86 and NEC V-Series microprocessors.

Specific questions about what Paradigm LOCATE can do for you? Call toll-free info-line 1-800-582-0864

Our Turbo C++ and Microsoft C	Complete EPROM support including splits and checksums	In-circuit emulator support types, local symbols, scopes and line numbers
Complete system documentation segment map	Full Compiler startup code, run-time library support and floating point	

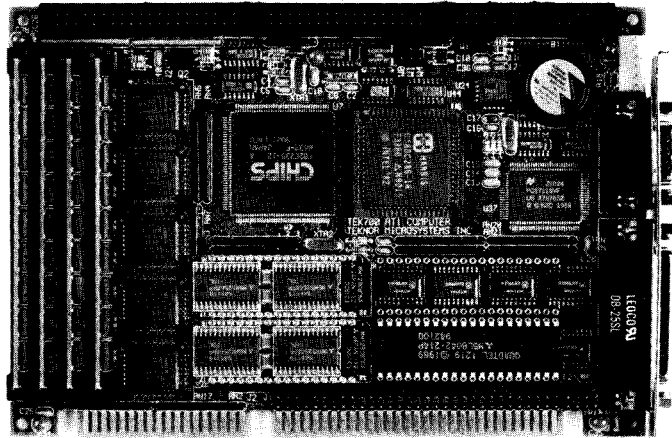


PARADIGM

Paradigm Systems
3301 Country Club Road, Suite 2214 • Endwell, NY 13760
To order: (600) 537.5043 • (607) 748-5968 (FAX)

Paradigm LOCATE is a trademark of Paradigm Systems

VERSATILE PC/AT SINGLE-BOARD COMPUTER



A feature-packed 80C286 16-MHz PC/AT single-board computer has been announced by Teknor Microsystems Inc. The **TEK-AT1** is a half-size PC-form-factor board that can be used in a PC/AT passive backplane or as a stand-alone unit in embedded applications. It employs CMOS technology for very low power requirements and for operation in an extended temperature range of -40°C to $+85^{\circ}\text{C}$.

The TEK-AT1 can support up to 512K of battery-backed static RAM, up to 1 MB of EPROM and FLASH EPROM, as well as up to 4 MB of system RAM. It supports Shadow RAM BIOS for fast execution, and has the ability to boot MSDOS, DR-DOS, and user applications from ROM. A real-time clock with battery backup, IIT2C87 or 80287 math coprocessor support, AT keyboard and speaker ports, and parallel and serial ports are included.

Other features include a watchdog timer to provide reset in the event of a software bug or processor failure; a power failure detector circuit that monitors the DC line and provides power fail warning, low-battery detection, or monitoring of another power supply; and a 62-pin XT expansion header. An on-board floppy disk controller supports up to two 3.5" or 5.25" drives with a capacity up to 1.44 megabytes. An on-board hard disk interface supports the AT embedded standard drives (IDE).

The TEK-AT1 can be ordered with either a 12.5-, 16-, or 20-MHz clock speed and various system memory requirements. The U.S. list price for a single unit is \$875. A **TEK-AT1PG** card, which connects to the header to provide a full VGA display controller with LCD, EL, plasma, and CRT support is also available.

Teknor Microsystems, Inc.
P.O. Box 455
Sainte-Thérèse, Québec, Canada J7E 4J8
(514) 437-5682 • Fax: (514) 437-8053
Reader Service #500

DUAL PORT INTERFACE CARD

A dual-channel, asynchronous serial card providing RS-422 and RS-485 communications has been announced by American Advantech. The PCL-743 is a half-size PC-compatible card capable of reliable communications over distances up to 5000 feet at 56,000 bps in noisy industrial environments. Applications include instrument interfacing, interface to PCs or other computers, multidrop data collection devices, a data terminal link, or other serial interface applications which require

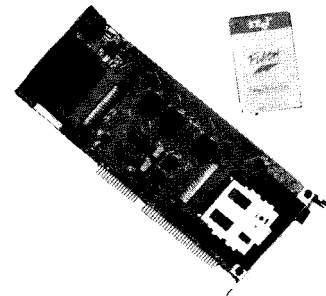
long-distance communication.

The PCL-743 features selectable addressing, including COM1 through COM4. Wait states of 0/2/4/6 are DIP-switch selectable to match the requirements of high-speed 386-based PCs. It also supports TxD, RxD, RTS, and CTS signals. RS-485 driver enable options of always on, RTS enabled, and control bit enabled are available and lumper selectable. Other features include two- or four-wire operation to support simultaneous sending and receiving, and on-board

termination resistors to provide proper impedance matching.

The PCL-743 has a power consumption of 400 mA typical, 950 mA maximum at +5 V and occupies one short PC bus slot. I/O connectors are dual 9-pin D-type. The PCL-743 sells for \$175.00.

American Advantech Corp.
1340 Tully Rd. 1314
San Jose, CA 95 122
(408) 293-6786
Fax: (408) 293-4697
Reader Service #501



FLASH MEMORY DEVELOPMENT BOARD

Intel has announced the availability of a Flash Memory Developer's Kit. The kit contains the hardware and software required to construct a complete Flash-based solid-state disk system.

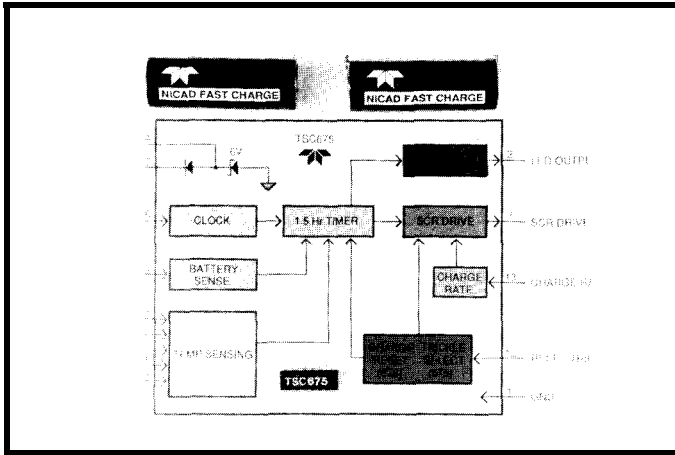
Built on a combination of the Intel Flash Memory Card and Microsoft's Flash File System, the Flash Memory Developer's Kit is designed to let engineers learn the principles of working with Flash memory and its related file requirements.

The Flash Memory Developer's Kit features an add-in board designed to work with either a PC/XT (g-bit) or PC/AT (16-bit) slot. The board contains two memory card sockets conforming to the PCMCIA standard electrical interface. Other features of the board include Vpp generation, board identifier, and switch-selectable page size, page location, and I/O port. The latter two features allow for multiple boards in a system, each using a different I/O and page base port.

The Flash Memory Developer's Kit is available starting at \$495 for a kit with a 1-megabyte Flash card.

Intel Corporation
1900 Prairie City Rd.
Folsom, CA 95630
(916) 351-2746

Reader Service #502



NICAD BATTERY CHARGER IC

A new family of Smart NiCAD battery charger integrated circuits that provides a low cost way to implement fast-charging of NiCAD batteries has been announced by Teledyne Components. The new devices, the TSC675 and TSC676, enable batteries to be charged safely in one hour without risk of overcharge and potential explosion. This feature is vital in the fast-growing equipment populations that include cellular telephones, portable computers, and battery-powered tools, where users need rapid recharge to maximize equipment use.

These integrated circuits have been designed for the most common NiCAD battery charger circuits using SCRs and current-limited transformers. The charge cycle begins when the IC detects the presence of batteries connected for recharge. It ends in two ways: an on-board clock in the IC stops the recharge after 90 minutes; or an external thermistor input accepted on the chip stops the charge cycle if

recharge is completed in less than 90 minutes, because of battery heat generated as full charge is reached.

In addition to automatic battery sense and dual-mode charge termination, the TSC675/676 devices provide an LED output that allows visual checking of the charge-cycle status. Automatic trickle charge is featured on the IC, with a timer override reset pin on the TSC676 and trickle-charge select pin on the TSC675. These features make these devices suitable for microprocessor-controlled charging systems.

Teledyne Components
1300 Terra Bella Ave.
P.O. Box 7267
Mountain View, CA
94039-7267
(415) 968-9241
Fax: (415) 967-1590
Reader Service #503

RS-232-TO-PC KEYBOARD INPUT CONVERTER

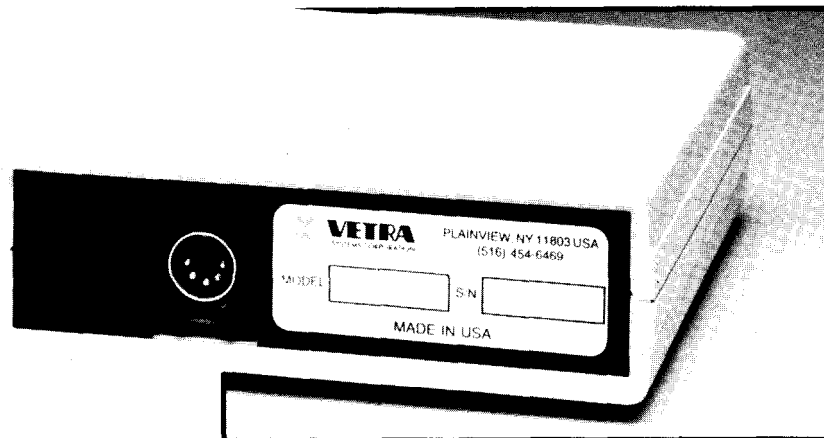
Data input to a PC can be simplified by means of the Smart Pipe RS-232-to-PC Keyboard Input Converter announced by Vetra Systems Corporation. The Smart Pipe interfaces RS-232 protocol devices to the standard keyboard port of an IBM PC or compatible, and makes it unnecessary to create software to handle character input. It converts the asynchronous serial data from RS-232 protocol, coding, and voltage to the standard keyboard protocol, coding, and voltage. It also generates the appropriate power-up responses and provides all the required handshaking so it can remain plugged into the PC at all times.

The Smart Pipe allows a variety of serial devices to be interfaced to a PC, leaving the PC's RS-232 serial ports free for other applications. No special software is required, since the Smart Pipe generates all standard codes. During system and software development, the standard keyboard can be used to simulate the RS-232 inputs. Replacement of the keyboard with the serial device and the Smart Pipe is transparent to the software. Applications for the device include industrial controllers, remote site control, communications network nodes, data acquisition systems, and display and demonstration equipment.

The RS-232 format accepted by the Smart Pipe is one start bit, eight data bits, no parity, and one stop bit. Switches are used to select data rates of 1200, 2400, 4800, or 9600 bps. Since the PC can inhibit input from its keyboard, the RS-232 source must also be inhibited. The Smart Pipe uses the Clear To Send (CTS) signal on its port for this purpose. The RS-232 source is allowed to send data only when CTS is present.

The Smart Pipe is available in two basic models: the VIP-331 and VIP-332. The VIP-331 translates ASCII characters, while the VIP-332 does no code translation; only voltage and protocol conversion. Models are available for the PC/XT (add -X to either model number), PC/AT (-A), or PS/2 (-P). Another version (-U) contains an internal switch to accommodate all three PC types. Pricing was not available at press time.

Vetra Systems Corp.
P.O. Box 714
Melville, NY 11747
(516) 454-6469
Fax: (516) 454-1648
Reader Service #504



NEWPRODUCTNEWSNEWPRODUCTNEWS

DSP FAMILY EXPANDS

Analog Devices has expanded the ADSP-2101 series of digital signal processing chips in two directions, offering devices with features ranging from very low cost to high system integration.

The ADSP-2105 is the low-cost member of the family. Pin- and code-compatible with the ADSP-2101, the ADSP-2105 has 1 k-word program memory and 512 words data memory. The ADSP-2105 also features a single serial port. The major feature of the ADSP-2105 is its price—\$9.90 in quantities from one to 100,000.

At the other end of the family, the ADSP-2111 adds a Host Interface Port to the standard ADSP-2101 function list. The HIP helps target the ADSP-2111 to multiple DSP applications, or applications where the DSP works in tandem with a non-DSP microprocessor. The ADSP-2111 is available with prices starting at \$87, quantity 100.

Both DSPs share major architectural features with

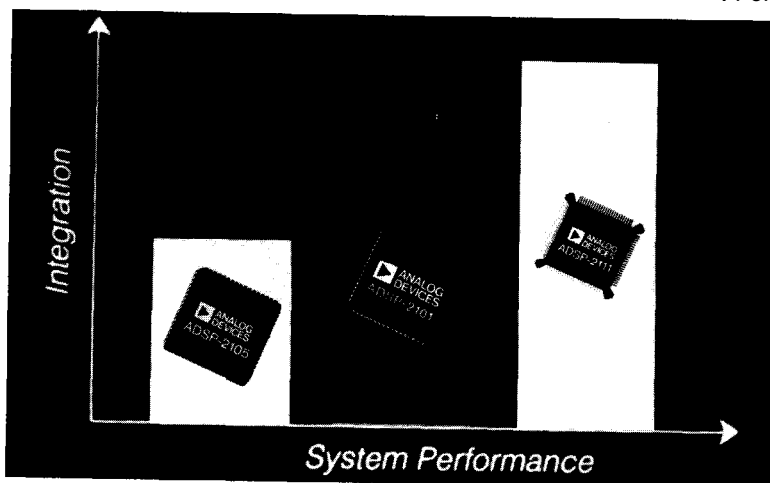
the other members of the ADSP-2100 family. These features include three independent computational units, two independent data-address generators, a program sequencer with zero-overhead looping and conditional arithmetic

instruction execution, and the ability for an internal program to boot into the DSP automatically from external memory or the HIP.

Analog Devices, Inc.
One Technology Way
P. O. Box 9106

Norwood, MA
02062-9106

Reader Service #505



UNIVERSAL PROGRAMMER

- 30 Day Money Back
- 1 Year Warranty
- Option: 4 Gang Adapter

- Software controlled 40-pin universal device programmer
- Interfaces with the IBM PC/XT/AT/386 or compatibles
- High speed parallel interface card to PC
- Programs PLD (PAL, PEF, EPLD, EPID, GAL), E(E)PROM (up to 4Mbits), BIPOLAR PROM, & Microcontroller
- Tests TTL/CMOS Logic & D/S Ram
- Reliable and fast programming with Normal, Intelligent, Interactive, Quick pulse algorithms
- Accepts JEDEC, INTEL Extended HEX, Motorola S, Tektronix HEX, Binary formats
- Manages 8, 16, and 32-bit word split
- Supports most compilers in JEDEC format

\$385

XELTEK
764 San Aleso Ave., Sunnyvale, CA 94086
Tel: (408) 745-7974, Fax: (408) 745-1401

COD, VISA, MC, AMEX

Desktop 9-Track Tape Subsystem

Now, 9-track tape lets your micro exchange data with minis and mainframes

9-TRACK is the first choice for file interchange among data processing professionals. Now, Qualstar's low cost 1/2-inch 9-track Streaming tape systems bring full ANSI data interchange to IBMPCs or Macintosh, giving your micro the freedom to exchange data files with nearly any mainframe or minicomputer in the world.

Available in both 7" and 10-1/2" versions, compact Qualstar tape drives can sit on your desktop, using less space than an ordinary sheet of paper. Systems include DOS or XENIX compatible software, coupler card and cables. High reliability 1600 or 6250 BPI capability may be used for disk backup as well as data interchange. Discover the big advantage 9-track tape has over other micro/mainframe links.

Simply exchange data files on a reel of 9-track tape.

DEC HP ALTOS NCR IBM MAINFRAMES AT&T UNISYS APOLLO SUN

Call us today!

FOR DETAILS AND TO ORDER
FAX (818) 882-4081
PHONE (818) 882-5822

QUALSTAR

9621 Irondale Ave., Chatsworth, CA 91311

#1 Selling 9-Track Systems on the Desktop

© 1989 Qualstar Corp. All product and company names and trademarks are the exclusive property of their respective owners.

68HC11 IN-CIRCUIT EMULATOR

A compact in-circuit emulator, providing complete development support for the Motorola 68HC11 microcomputer family in expanded and single-chip modes, has been announced by Wytec Company. The WICE 68HC11 uses a host PC as a peripheral device for simulating target hardware in real time during software debugging. While running the user's code, the target program can access the host PC's keyboard, CRT display, and speaker in real time. For many applications, it allows software debugging in real time without target hardware.

The WICE 68HC11 comes with window-oriented, user-friendly PC driver software that runs

under MS-DOS on the IBM PC, XT, AT or compatibles and permits real-time and full-speed emulation at clock rates up to 14 MHz. The emulator communicates at 115.2 kbps via an RS232 serial link to the host PC and monitors 30 programmable memory locations, 17 stack locations, and all registers. Besides reading symbol files in 2500AD, Microtek, and Zak formats, and those generated by other assemblers and C compilers, its symbolic debugger includes a utility program which creates a symbol file from the Motorola cross-assembler (a public domain program) for downloading symbols into the WICE 68HC11.

Other features include scope or logic analyzer trigger output, on-line assembler, disassembler, EEPROM erasing and



programming, memory enter, compare, move, fill, search, single stepping, and eight software breakpoints. Its PC interface allows upload and download of files in Motorola S record, Intel hex, and Tektronix hex formats.

The WICE 68HC11 sells for \$795 and includes a 5-V power supply, an emulator cable with a 52-pin PLCC plug, an RS-232-C cable, and PC driver software.

Wytec Company
185C E. lake St. #140
Bloomington, IL 60108
(708) 894-1440
Fax: (708) 307-9809

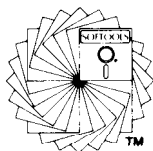
Reader Service #506

Z280-Z80-64180-8085

Advanced Relocating Macro Cross **Assembler**

- Includes linker, librarian, and cross-reference utility
- Generates full source-level debugging information
- Automatically bank switches program greater than 64K using MMU
- Linker allows code placement at both physical & logical addresses
- Outputs binary, Intel Hex and Extended Intel Hex files
- Compatible with M80, SLR, 2500AD, and Avocet Assemblers
- Built-in MAKE facility supports dependency file checks
- FREE demo disk available
- Compatible C-Compilers available soon!

If you're ready for a fast, full-featured, affordable product, give us a call. You'll be pleasantly surprised.



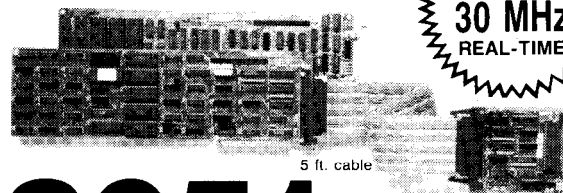
SOFTTOOLS INC.
8770 Manahan Drive
Ellicott City, MD 21043
301-750-3733
301-750-2008-Fax/BBS

ONLY \$249

Reader Service #193



"The Best 8057 Emulator"



8051

SEE EEM 90/91
Pages D 1320-1323

PC based emulators for the 8051 family

8031, 8032, 8051, 8052, 80C152/154/321/451/452/51FA/51GB/515/517/535/537/552/562/652/851, 80532, 83C451/552/652/751/752/851, 8344, 87C451/552/751/752, 8751, 8752, DS5000 + CMOS more.

- PC plug-in boards or RS-232 box.
- Up to 30 MHz real-time emulation.
- Full Source-level Debugger w/complete C-variable support.
- 48 bit wide, 16K deep trace, with "source line trace."
- "Bond-out" pods for 8051, 83C552, 83C451, 83C652, 83C751, 80C515/80C517, 83C752.

Prices: 32K Emulator 8031 \$1790: 4K Trace \$1495' *US only

CALL OR WRITE FOR FREE DEMO DISK!

Ask about our demo VIDEO

NOHAU CORPORATION
51 E. Campbell Avenue
Campbell, CA 95008
(408) 866-1820 FAX (408) 378-7869

Reader Service #182

FEATURE ARTICLES



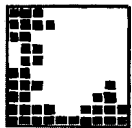
page 12

DSP Architecture for Signal Processing Applications



page 22

A PC Stopwatch



page 24

Digital Image Processing



page 38

Mini-DSP



page 46

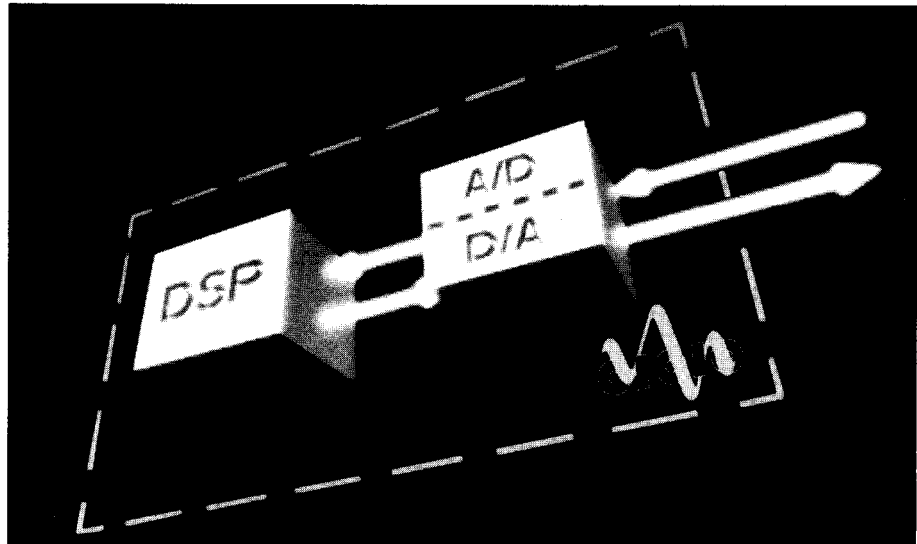
Analog Circuit Design

DSP Architectures for Signal Processing Applications

Matching the Tools to the Job

DSP stands for digital signal processing, but this simple phrase blurs many facets of DSP. DSP can be divided into roughly two application areas: numeric processing, where the function of the DSP circuitry is to accelerate the microprocessor's calculations (off-loading them via an IC optimized for calculation); and signal processing, where the DSP is optimized for not just general acceleration, but is really designed for the execution of signal-processing-type of DSPs which are appropriate for one activity are generally a poor choice for the other.

General numeric processing executes code implementing many disparate activities. These include sorting data, finding averages, comparing and updating bank balances, checking lists, and performing mathematical operations in a wide variety of sequences. In contrast, signal processing code usually implements a variation or embellishment of just a few algorithms. For signal processing, execution time is the real enemy, since many applications require real-time results. It's a very different situation than saying "calculating that bank balance will take 0.13 ms instead of 1.0



imple- processing means menting algorithms that analyze, manipulate, or extract features from data representing the digitized version of analog signals acquired in the real world. Examples include voice recognition, modem signals, and seismic and vibration data, among many o t h e r s .

ms"-in the signal processing world, it's often either fast enough or useless because it doesn't meet the real-time m a n d a t e !

The basis for most signal processing algorithms is the sum-of-products:

$$\sum_{i=1}^n a_i b_i$$

where a_i and b_i are ordered sets of data points or results of previous calculations. Real-world signal DSP applications include digital filters such as the finite impulse response system modeled in Figure 1, or the decimation-in-time (DIT) discrete Fourier transform (or fast Fourier transform, FFT), shown schematically in Figure 2 along with its algorithm flowchart, Figure 3. In both applications, there is a repetitive pattern of data handling, as well as unique symmetries in data structure-factors which do not exist in general numeric processing.

Unlike the selection of a suitable op-amp or A/D converter, picking a DSP also involves things which are difficult to characterize: the performance in the intended application. A suitable architecture will accomplish the complete DSP algorithm in less time, while requiring less memory and other support; it will also be easier to program optimally—a major consideration. Simple benchmarks like cycle time or multiplication time have little relevance when trying to guarantee execution of the complete algorithm's code in short enough time.

A wide selection of DSP ICs are available from several vendors, including Analog Devices, AT&T, Motorola, NEC, and Texas Instruments, and matching the IC to the application involves subtle judgement. Because the final DSP code is often written at the assembly language level, the source code cannot be ported over to another DSP. The selection of an appropriate digital signal processor re-

quires tradeoffs in cost, memory, speed, programming effort, power consumption, and similar traditional engineering factors.

For example, signal processing DSPs have two independent memories (Harvard architecture)—one supplying the program steps (the operations) and the other supplying the data

time (or at least fast enough to be truly useful) provide:

a) fast and flexible arithmetic, including single-cycle multiplication (often with accumulation), shifting, and logic operations. There should be no need to rearrange operands or order of arithmetic operation, since these actions waste valuable time.

b) extended dynamic range on multiplication/accumulation—needed to deal with sums-of-products common in DSP—and protection against overflow, which may result from repeated accumulations.

c) single-cycle fetch of two operands, as data for repeated sum-of-products calculations. If fetching the

data pairs takes one cycle per operand, little time is left for arithmetic.

d) circular buffering in hardware, for efficient execution and minimal software burden in applications such as digital filtering, along with hardware for address pointer wraparound from end back to beginning.

e) looping and branching with zero overhead: The often inherently repetitive DSP algorithms are commonly implemented as program loops, and should execute without extra-cycle penalty for checking the end of the loop or for conditional branching out of the loop. Using an additional cycle to check the If...Then (or For...Next) loop for exit conditions is very inefficient, and most of the time the result of checking is to go back to the beginning of the loop.

Three aspects of DSP architecture are especially critical to real-time sig-

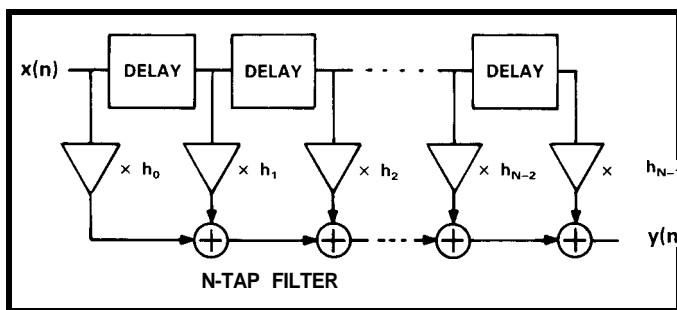


Figure 1 -A FIR filter is a typical DSP application.

(operands). In contrast, a non-signal processing DSP relies on the conventional von Neumann architecture with a single memory space shared by both operation opcodes and data.

Of course, two memories require two program counters to address the correct memory location. This leads to lots of confusion in terminology, since the simple address bus and data bus combination is now replaced by four: program memory address (PM A) bus, program memory data (PMD) bus, data memory address (DMA) bus, and data memory data (DMD) bus! And, this DMA is unrelated to direct memory access DMA!

EFFECTIVE ARCHITECTURE REQUIREMENTS

Effective DSP architectures for processing real-world signals in real

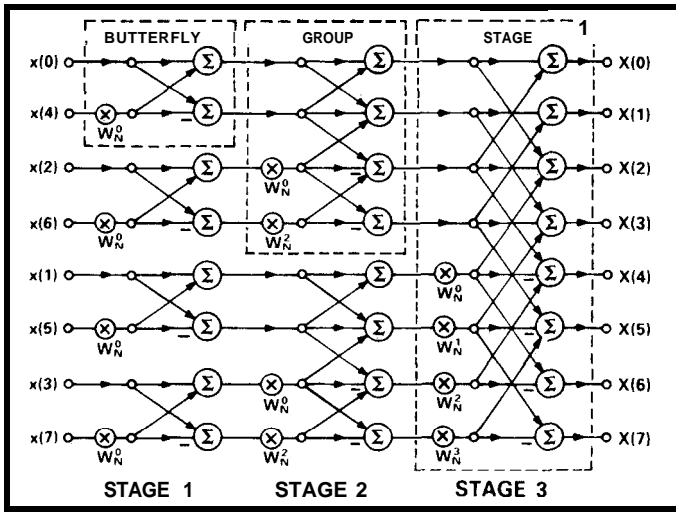


Figure 2—An eight-point decimation-in-time (DIT) FFT shows the repetitive pattern of this common DSP function.

nal processing: arithmetic processing (items a and b), data addressing (c and d), and program sequencing (e). We can look at these using the ADSP-2100 family of DSP processors from Analog Devices Inc., which is designed with these factors optimized for signal processing, in contrast to general processing and calculation

ARITHMETIC PROCESSING

The arithmetic section of the ADSP-2100 has three computational units, shown in Figure 4, linked but independent of each other. The arithmetic and logic unit (ALU), multiplier/accumulator (MAC), and barrel shifter are connected via the results (R) bus, so that the output of any unit can be used as input for itself or any other unit on the next cycle. Operands for the ALU and MAC can come from program or data memory, or various registers.

ALU operations can be done on any combination of the two X and two Y input registers, which in turn can be loaded with any combination of program or data memory, or other data registers within the processor. An example of a multifunction ALU instruction for addition is:

```
AR (the ALU result) = AX0 + AY1
AX0=DM(I0, M3)
AY1=PM(I4, M7)
```

where the first part is the addition, the second part loads one X input register from data memory, and the third part loads a Y register from program

memory. The entire operation is completed in a single cycle (80 nanoseconds for a 12.5-MHz clock ADSP-2100A).

In contrast, some DSP architectures require that one operand comes from the accumulator while the other comes from either the multiplier or from the data bus (via a shifter). When adding two numbers with these other architectures, the accumulator is first loaded with one data number, and then the second number is added to the accumulator—a two-cycle operation. In addition, for this result to be used as an input for anything other than another ALU operation, the data

must first be transferred from the accumulator to data memory. These restrictions result in a severe arithmetic throughput penalty.

The MAC in the ADSP-2100, like its ALU, has two X and two Y input registers. MAC operands may be loaded from any combination of program memory and data memory, or other processor registers; the MAC feedback and result registers can also serve as operands for any MAC operation. Two new operands can be loaded into the input registers in parallel with computation so that a new MAC operation, with new operands, can be started with every cycle (even with off-chip memory accesses).

By comparison, some DSP architectures do not have a multiplier/accumulator as a single entity, although they do have a multiplier separate from the ALU. A complete MAC operation thus requires two cycles: one to perform the multiplications, and one to perform the accumulation (in the ALU). The interdependency of ALU and multiplier means that MAC operations cannot be intermingled easily with ALU operations; the order of calculations in the final algorithm may have to be changed to avoid conflicts. In addition to requiring more system time, the instructions

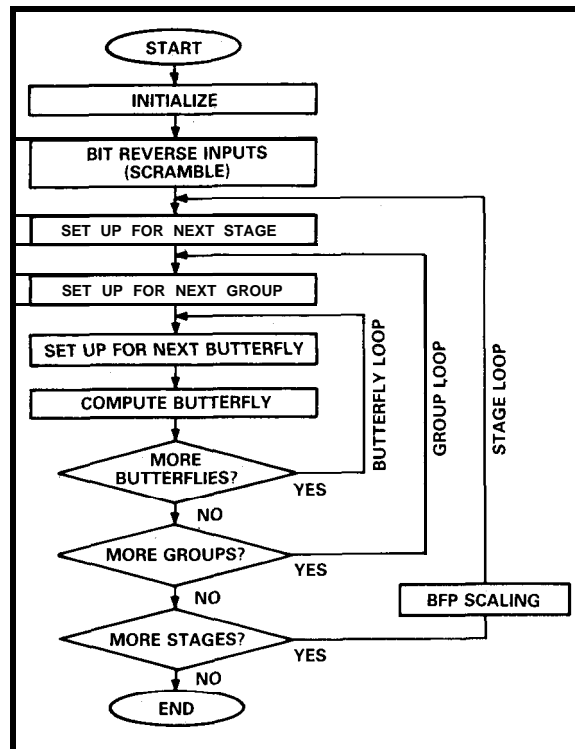


Figure 3—The DIT FFT flowchart shows the nested looping structure of the algorithm.

RISC VS. CISC VS. DSP ARCHITECTURES

Why not avoid all these signal **processing** architectural issues by using a RISC (reduced instruction set computer) architecture, **since** RISC seems a **panacea** for so many other problems?

As central processor architectures matured, their Instruction sets became 'richer' and more complex. A **CISC** design includes instructions for basic **processor** operations, plus single Instructions that are sophisticated enough to **evaluate** a high-order polynomial, for example. But **CISC** has a price: many of the instructions execute via microcode in the CPU and require numerous clock cycles, plus silicon real estate for code storage.

In contrast, the reduced instruction set computer (RISC) recognizes that in many applications, basic instructions such as LOAD and STORE-with simple addressing modes-are used much more frequently than the advanced instructions, and should not incur an **execution** penalty. These simpler **instructions** are 'hard-wired' in the CPU logic to execute in a single clock cycle, **reducing** execution time and CPU **complexity**.

RISC AND DSP APPLICATIONS

Although the RISC approach offers many advantages in general-purpose computing, it is not well suited to DSP. For example, most RISCs do not support single-instruction multiplication, a very common and repetitive operation in DSP. The DSP is optimized to accomplish its task fast enough to be 'real time' in

the context of the application, which requires single-cycle arithmetic operations and accumulations.

DSP algorithms have unique needs not found in general-purpose **computing**: circular buffering, pointer updating and fast looping **with** zero overhead, bit reversing, barrel shifting, scaling, and data-dependent **execution** branching. Each of these should execute **within** the DSP **instruction**, and not as a separate time-consuming **instruction** cycle. The computational unit **within** the DSP must be run **efficiently**, **with** data arriving from **at least two separate data address generators** and no. time penalty for data access. CISCs and RISCs support **virtually** none of these needs.

Software programming also **differs**. RISCs and CISCs are programmed in high-level languages to **minimize** software development time and **hide** the Instruction set from the programmer. For DSP, however, code **optimization** (primarily of execution time, but also of memory usage) requires that the **software** engineer use assembly language to get the satisfactory performance. **Critical sections** of the program are **examined** and recoded if needed, to reduce overall **execution time**, after simulation and run-time histograms.

In theory, any processor-even a hand-held calculator-can accomplish any software task, given enough time. However, **DSPs** are optimized for the unique **requirements** of a real-world signal processing computational needs and algorithms, **while** CISCs and RISCs are better suited for general-purpose calculations, and where real time is usually not a factor.

DATA ADDRESSING

Fast arithmetic, of course, is wasted if the required data cannot be fetched at a commensurate speed, regardless of source. To fully utilize the separate data and **program memories** of the "Harvard" architecture used in most **DSPs** (in contrast to the single interleaved program/data memory of the von Neumann computer architecture), the data addressing must support simultaneous dual operand fetches. Circular buffers are often found in DSP algorithms, and a signal processing **DSP** supports these by way

for the two-step process require more program memory.

The barrel shifter accepts as input any result register in the processor, including its own result (or its own input register). The shifter can place a 16-bit input value anywhere within a 32-bit field in a single cycle, and shift any number of input bits from off-scale right to off-scale left. Functions such as exponent detection, **normalization** and **denormalization**, and block floating-point manipulation can be realized via this shifter. All shifts, regardless of number of bits to be shifted, are performed in a single cycle.

MINDSEYE

SYNERGIZER™

Converts your PC into a Brainwave Entrainment machine.



This unique new hardware-software combination turns your PCXT/AT/386 or clone into a lab-grade, audio/visual brainwave synthesizer. The Synergizer uses precisely controlled light and sound patterns to safely induce electrical activity in the brain, providing easy access to Alpha and Theta states.

Features of the Synergizer include: independent light and sound control, multiple time ramps and sound/light levels (over 32,000), on-board stereo sound generation with waveform, filter and other sound parameter controls, and an optional Software Toolkit. The latter provides the source and object code necessary to custom program the unit in "C." The Synergizer provides more capability than any other similar device at a remarkably low price. Requires DOS 3.0 or above, 512 KRAM, and hard drive.

MINDSEYE SYNERGIZER \$395
Board, software, lights, manual.
EXTERNAL CONTROL UNIT \$95
Software-assignable controls.
STUDIO TYPE HEADPHONES \$35
Standard MiniStereo plug.
SOFTWARE TOOLKIT \$50

SYNETIC SYSTEMS
The leader in Personal Mind-Tech

ORDER PHONE
800-388-6345

Credit Cards Accepted

Information (206) 632-1722
PO Box 95530
Seattle, Washington 98145

© 1991

Reader Service #198

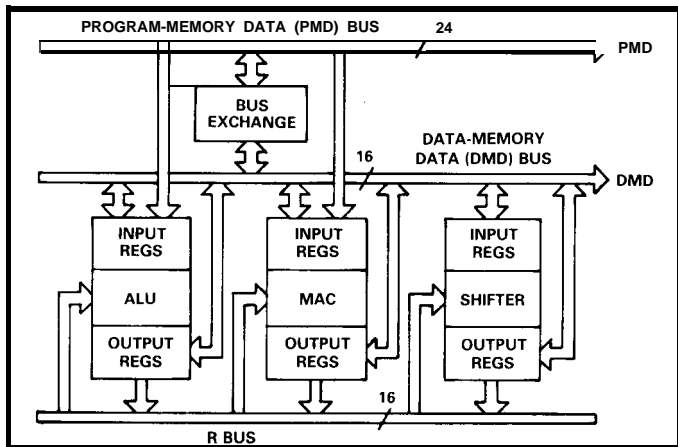


Figure 4—The ADSP-2100's arithmetic section is quite different from most non-DSP microprocessors.

of built-in address pointer wrap-around.

There are two separate address generators in the ADSP-2100: one typically supplies addresses for program memory data fetch while the other supports data memory data fetch. Each address generator has multiple registers to store pointers (addresses), address modifiers, and buffer lengths for circular (modulo) addressing. For efficient FFT execution, the address generator can "bit-reverse" with zero overhead an address as it is being sent out.

Both indirect and direct addressing are available. In indirect mode, the address-in register is updated by the contents of a modify register, as it is being put on the bus. The pairing of the various base address and modify registers is up to the programmer, useful for two-dimensional array addressing or for pointer increment/decrement. The 24-bit-wide instruction, Figure 5, includes four 2-bit fields to point to specific program memory and data memory address registers, plus their respective modify registers.

A special register is used for circular addressing. Loading a nonzero buffer length into this register automatically activates the modulus logic. The address and its modulus are maintained transparently by the address generator without explicit calculation by the programmer; like many other ADSP-2100 functions, this internal calculation has zero overhead.

Some DSP processors support both direct and indirect addressing but through a very limited set of address and modifier registers. This limits flexibility in interleaving several indirect sequences for complex algorithms since the old modify value must be stored and a new one written before the new indirect mode is used. Similar constraints exist for base address switching.

PROGRAM SEQUENCING

An efficient DSP for signal processing has little or no overhead wasted in maintaining the desired flow of control. Loops are fundamental to many signal processing algorithms (typified

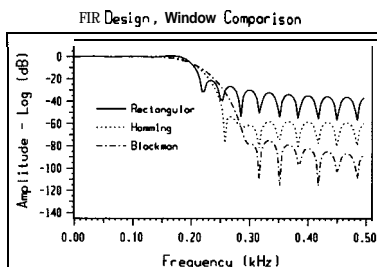
DspHq
Software

SIGNAL PROCESSING SOFTWARE

Digital Signal Processing Headquarters, DspHq, is IBM-PC based data analysis software that integrates data input, manipulation, display, and output. DSP boards and data acquisition products are optional. DspHq provides users with an easy to use application program for the evaluation, simulation, and prototype of DSP analysis routines and systems **with or without programming!**

APPLICATIONS

- ☐ Data analysis
- ☐ DSP algorithm design
- ☐ Digital filter design
- ☐ Simulation
- ☐ Real-time software development
- ☐ Communications
- ☐ Speech processing
- ☐ Biomedical engineering
- ☐ a Sonar / Radar
- ☐ Seismology



Example of graphic generated by DspHq in HPGL format.

FREE DEMO!
800-848-0436

FEATURES

- ☐ Algorithm design & system simulation
- ☐ Prototype algorithms in C or Pascal
- ☐ Integrate existing function libraries & DSP boards
- ☐ Menu driven interface
- ☐ Interactive script language with run-time user input
- ☐ Advanced graphics with publication quality hardcopy

☐ **No programming required!**

BITTWARE Research Systems • 400 East Pratt Street • 8th Floor • Baltimore • MD • 21202
Demo & Info: 800-848-0436 • BBS (3-12-24 N 8 1): 301-838-3205 • FAX: 301-879-4465

Reader Service # 117

8031/51 Tools

8031 In-Circuit Emulation

Our emulator provides most features of an 8031 In-Circuit-Emulator at a significantly lower price. It assists in integration, debug, and test phases of development. Commands include: disassembly, trace, breakpoint, alter register/memory, and load Intel Hex file. Enhanced ROM option adds real-time execute-to-breakpoint and line-by-line assembler to the ICE or development board. Also allows emulation of the 8032 processor.

8031 ICE - \$199

80C31 ICE - \$249

Enhanced ROM add \$70

NEW!

80C51 FA ICE \$329

80C652 ICE \$329

Includes Enhanced ROM

8051 Simulation

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

8031/51 Single Board Computer

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

Call us for your custom product needs.

Other products available:

MyGAL - GAL Programmer \$199

FORTH Card - FORTH development card for STD Bus \$279 (OEM-\$1 99)

HTE

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

(619) 566-1 892

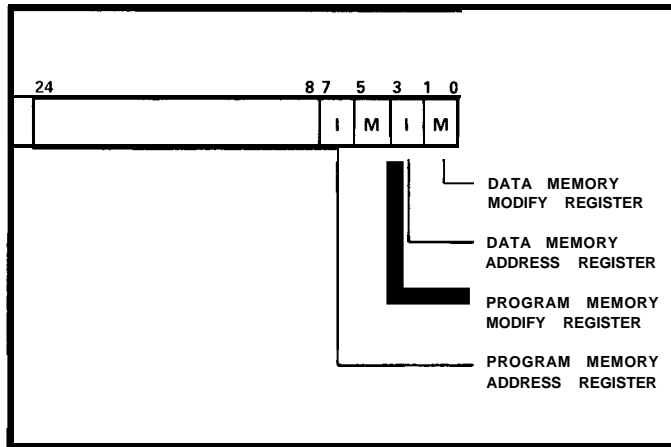


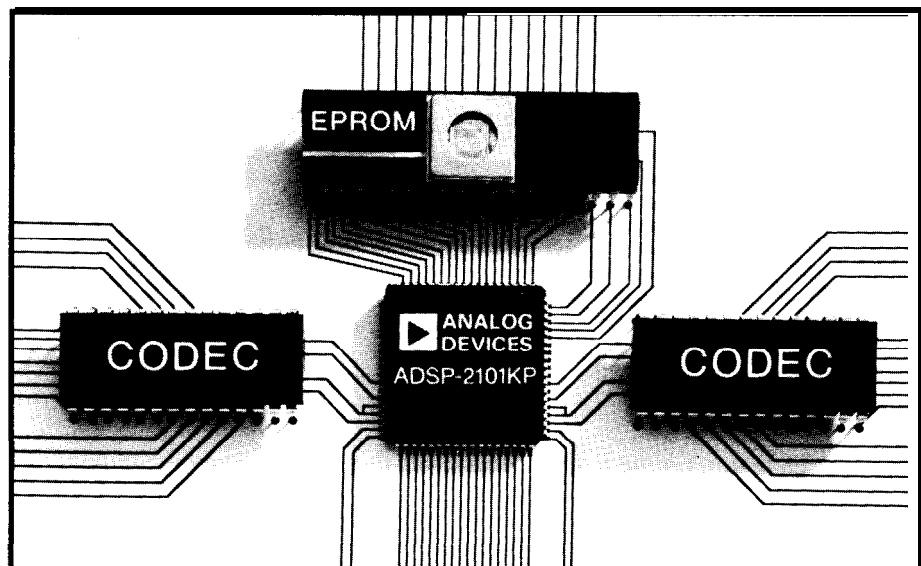
Figure 5—Indirect addressing is often used for efficiency in algorithm execution.

by the ubiquitous sum-of-products operation). When a DSP program can be expressed in loop form, the coding is simplified and shorter; further, changes require less work (such as changing the number of taps in the FIR filter). Equally critical, branching specifies conditions under which the loop terminates and program execution begins at new point.

The ADSP-2100 program sequencer, Figure 6, selects the next address for the address bus from either the program counter (for sequential addressing), the instruction word itself (for direct jumps and subroutine calls), the program counter stack (for returns from subroutines and interrupts), or the interrupt logic (to vector to the interrupt routine). All address selection and execution occurs in a single cycle; when an interrupt occurs, all processor status registers are automatically pushed onto the status stack for later recall.

When address looping is used, it is automatic and transparent. Without any extra checking cycles, the processor determines if a loop should terminate (either because it has run the specified number of cycles or a branch condition is met) along with the next instruction address. In one cycle the last instruction of the loop is executed and on the very next cycle the next instruction is executed (either within the loop or outside, upon termination).

To achieve speed, some DSP architectures use a three-level pipeline (for instruction prefetch, decode, and actual execution) in the program sequencer. They also require an extra instruction to check for loop count or branching conditions. Any deviation from the sequential flow of instructions, such as for returning to the beginning of a loop or terminating the loop, requires that the pipeline be emptied and then refilled.



\$129

(without memory)

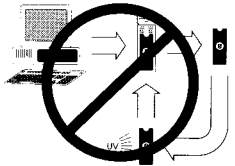
\$149

(32K x 8 SRAM)

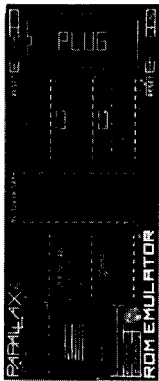
\$169

(32K x 8 NV SRAM)

Avoid the hassles of programming EPROMs



A ROM Emulator can greatly reduce the time spent writing and debugging ROM code



Emulates 2764, 27128, & 27256

Plugs into target ROM socket and connects to PC parallel port via modular telephone cable

Accepts 32K x 8 SRAM or NV SRAM

Loads Intel Hex, Motorola S, hex, and binary files

High and low RESET outputs for automatic startup after downloading

Includes all necessary software and cables **works right away!**

PARALLAX

(916) 721-8217

FAX: (916) 726-1905

Parallax, Inc.

6200 Desimone Lane, #69A
Citrus Heights, CA 95621

California residents add sales tax.
Shipping: \$4.00 for UPS ground,
\$9.00 for UPS 2nd day, \$18.00 for UPS next day.



ROM EMULATOR

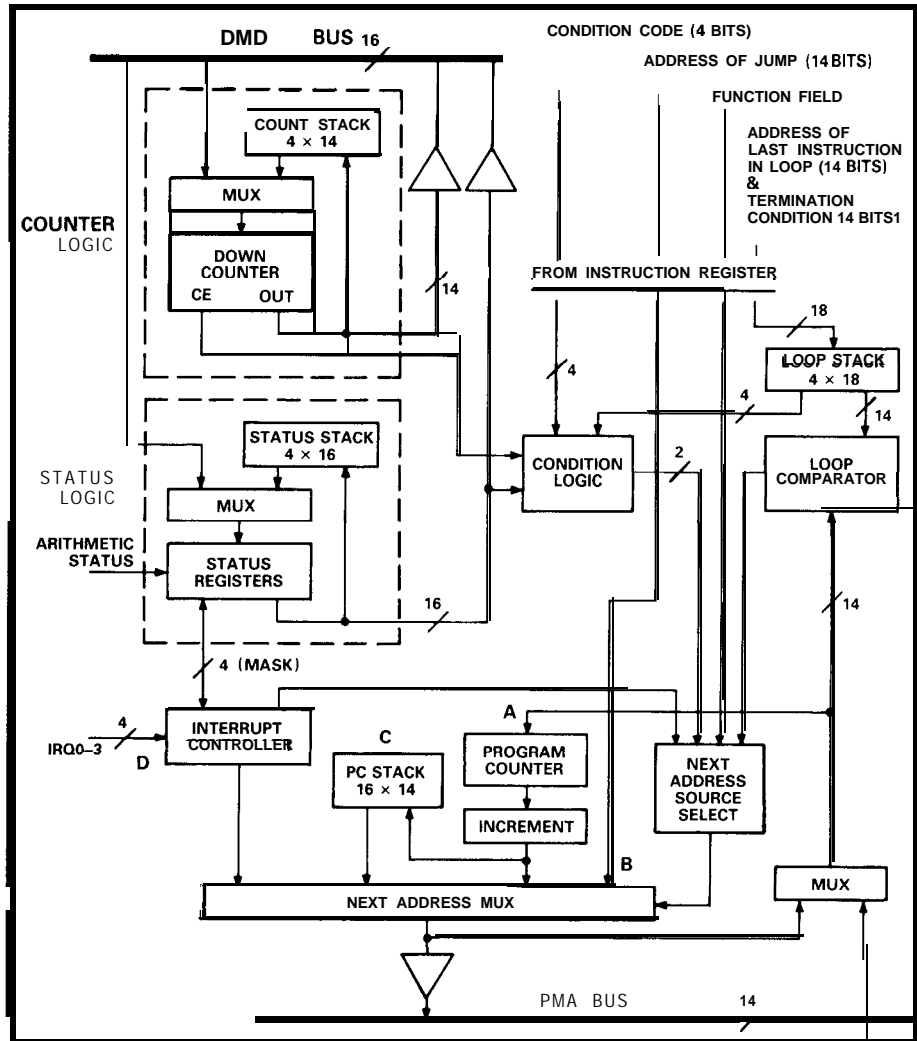


Figure 6—One of the features of the program sequencer architecture is that the next program address can come from one of four sources.

besides making analysis of program flow complex, this approach encourages straight-line, nonlooping coding of algorithms. These are inefficient to program and consume more memory than loops. When a separate, explicit instruction is needed to check loop count and branching, there is **one** cycle of overhead for each iteration.

A CHOICE FOR ENGINEERS

DSP chips are like "standard" microcontrollers in a crucial aspect: in both cases there are architectures and design philosophies to match almost any application. It's important to keep the inherent strengths and drawbacks of various DSPs in mind as you make your design decisions.

DSP chips, as high-speed, powerful computing components, can be an important addition to your applica-

tion design toolbox. Understanding the implications of DSP architecture assures that you match the tool to the job at hand. ❖

REFERENCE

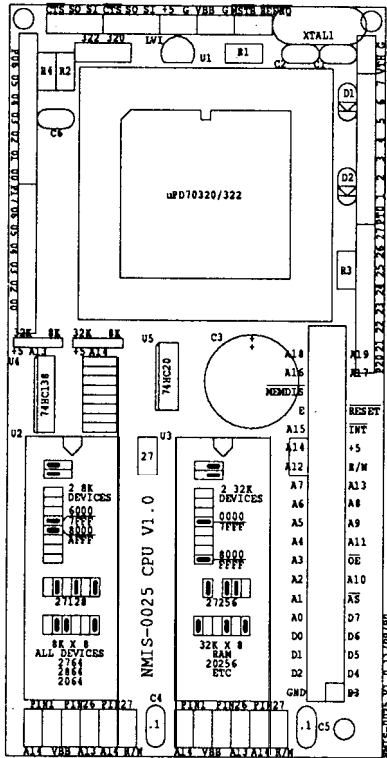
'Digital Signal Processing In VLSI,'
Richard J. Higgins, Englewood Cliffs,
NJ: Prentice Hall, 1990.

Bill Schweber is a senior technical marketing engineer at Analog Devices Inc. and holds a B.S. and M.S. in electrical engineering. He has designed microprocessor-based real-time machine controls, has been a product marketing engineer, and has authored numerous technical articles and written three text books.

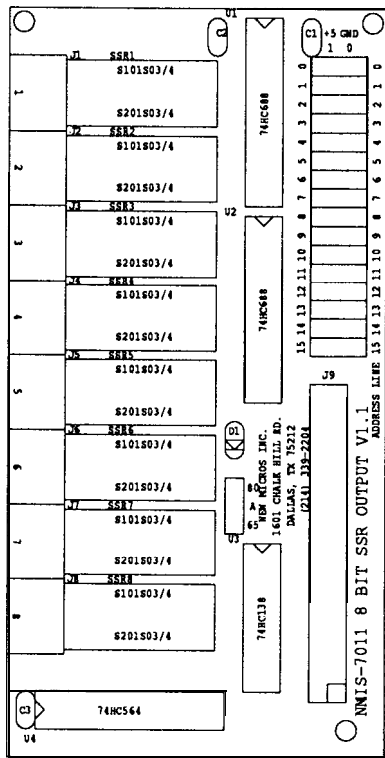
IRS

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

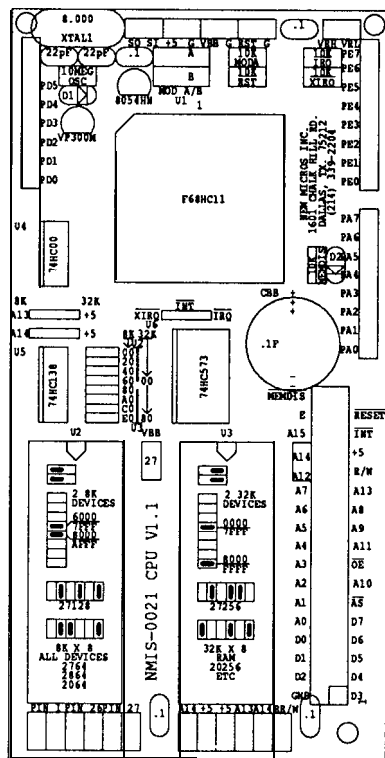
V25 NMIS-0025



SSR's NMIS-7011



68HC11 NMIS-0021



NEW MICROS, INC.
1601 Chalk Hill Road
Dallas, Texas 75212
Tel: (214)-339-2204

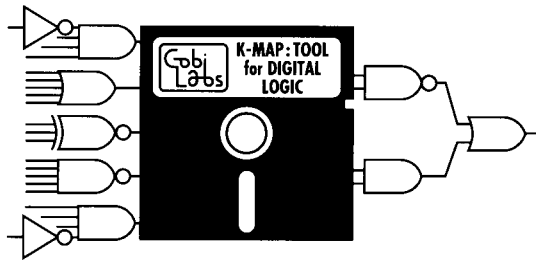
Our new V25 CPU (left) fits on our SSR board (above), and a whole host of other peripherals, (over 40 in all) just like our very popular 68HC11 CPU (right) could three years ago...

NMIS-0025 \$200,
NMIS-7011 \$99,
NMIS-0021 \$99.

More Value! Less Money! Ain't competition grand?

Reader Service #180

DIGITAL DESIGNERS MINIMIZE YOUR LOGIC

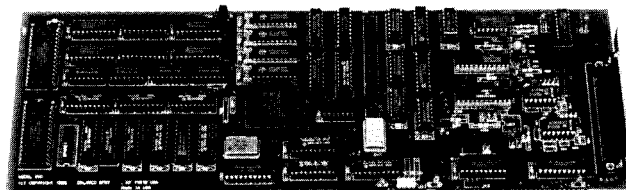


KARNAUGH MAP SOLVER FOR PC's Minimum Gates from Your Specs

- Combinational, Sequential
- Mealy & Moore State Machines
- 2 to 14 Inputs, **Any Number Outputs**
- Sum of Products, Product of Sums
- JK, D, SR, and T Flip Flops
- Expert Reference and Tutorial
- 60 Day Money Back Guarantee
- **MUCH MORE**

SAVE 40% = **\$110** + \$5 (P&H)

GOBI LABS: (407) 297-0862
BOX 616304, ORLANDO, FL 32861-6304



Model 250 for Algorithm Development,
Data Acquisition, Instrumentation. Audio.

- TMS320C25 DSP at 10 MIPS.
- Up to 192 Kwords RAM.
- Multi-Channel Analog IO - 250K Samples/sec.
- Development Software, including Assembler & Debugger.
- Applications Software includes FFT, Signal Display, Data Acquisition & Waveform Editor.
- No Gap Sampling to/from Disk at Very High Rates.
- Supports Multiboard & Standalone (EPROM) Operation.
- From \$1095. Other DSP Products Available.

DALANCO SPRY

89 Westland Avenue
Rochester, N.Y. 14618
(716) 473-3610

Reader Service #143

Reader Service #134

FEATURE ARTICLE

David P. Schulze

A PC Stopwatch

Improved Timing for Acquisition and Control

Have you ever wondered how fast it really ran? A little piece of your software, I mean. You know, those seemingly trivial loops and simple algorithms that we all take for granted? It takes 129 μs to use `INT 10H` to find out where the cursor is on my 80386 clone. It takes 69 μs to convert a text character to a graphical pattern using an algorithm I thought was great.

There have been times when simple answers to simple questions have been just out of my reach because the timing methods were either just too coarse, too crude (loop 1000 times and measure with a stopwatch),

or too expensive. Why couldn't I have a stupid counter running off a 1-MHz crystal that simply plugs into my PC?

With a handful of TTL chips and an inexpensive schematic capture package, I set out to do just that. The schematic in Figure 1 shows eight LSTTL chips that can be addressed on an 8-byte boundary anywhere in the 1K I/O address space of the IBM PC.

The 16-bit counter comes from daisy-chaining two 74LS393 counters together. The relatively slow clock speed of the 1-MHz crystal oscillator keeps the propagation delay rippling through the counters from becoming

an issue. The output of the counters is input to a pair of 74LS244 tristate bus drivers. When the PC uses a 16-bit read, both ports are read in the proper order, loading the least-significant byte of the count into the AL register.

One race condition to watch for is that the processor could catch the clock at the transition of the LSB from FF to 00. If the LSB is input by the first half of the read and the clock is allowed to continue before the MSB is read, the time reported would be incorrect. This was solved by the cluster of NAND gates that control the timer reset and clock enable.

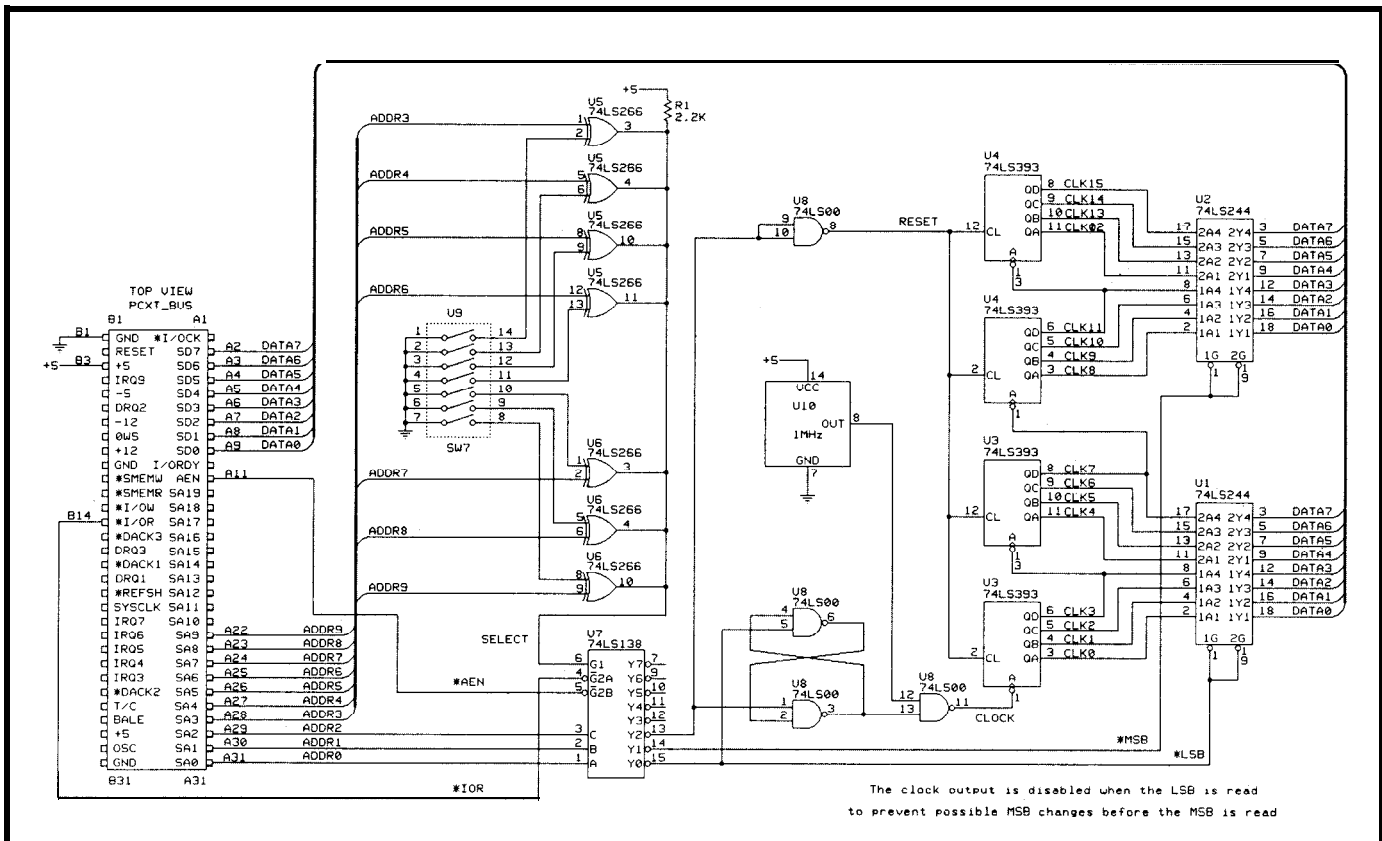


Figure 1—The PC stopwatch relies entirely on discrete logic to keep track of time intervals under program control.

The line called CLOCK is enabled when the third port is read. For example, if 3EOH is chosen as the base port address, the following would reset the timer and enable CLOCK:

```
START: MOV DX, 3E2H
        IN  AL, DX
```

To read the elapsed time, simply,

```
READ:  MOV DX, 3EOH
        IN  AX, DX
```

This instruction will freeze the timer by disabling CLOCK going into the first counter at pin 1 on U3.

The user-configurable port address of the PC Stopwatch is made of a 14-pin switch block, two 74LS266s (quad XNOR gates with open-collector outputs), and a 74LS138 address decoder. The '266s do most of the work decoding the port address by only permitting the SELECT line to pin 6 of the '138 to go high when bits 3-9 of the address bus match the switch settings. The '138 simply decodes the bottom

three address lines from the PC and converts them into the actual ports enabled for the read. Special attention should be paid to pins 4 and 5 of the '138. Notice the *IOR and the *AEN lines. The *IOR is simply the PC's port read line. This is similar to port read lines on other micros. The *AEN line, however, is much more specific to the PC. This line goes high during a DMA transfer. If it is ignored, every time the PC reads from the disk, the timer would be falsely triggered and bogus data could be input with the disk data.

As a software guy, it took me a while to put this thing together, but the result was as simple as it was effective. With the stopwatch, I can measure up to 65 ms worth of processing time with 1- μ s resolution. True, I don't always need that much resolution, but I now know exactly how much time my interrupt service routines will take. Using diagnostic C functions such as "StartWatch ();" and "time = ReadWatch ();" if it is easy to know where my timing problems are.

There are probably as many different variations of enhancements to this simple idea as there are people who need to know the performance of their software. I found that this was a good place to start. ❖

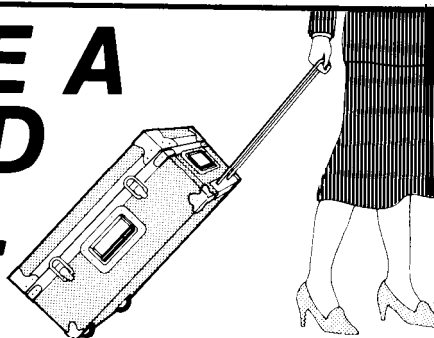
[Editor's Note: If you have developed an add-in timer for your desktop computer, or if you modify this design, we'd like to know about it. If we receive enough designs, we'll do a special article on high-resolution timing in desktop systems. Send your design to: Timer Design Project, CIRCUIT CELLAR INK, 4 Park St., Vernon, CT 06066.1

David Schulze is a specialist in real-time systems design and interrupt-driven environments. He received his B.S. in Computer Science from Eastern Connecticut State University and is noted for his work in software engineering by "Who's Who in the Computer Industry."

IRS

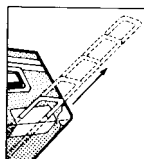
404 Very Useful
405 Moderately Useful
406 Not Useful

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.



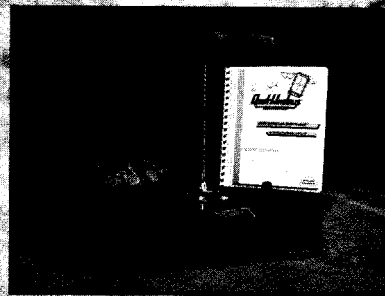
UNLOAD ON US!

Call or write to tell us about your shipping or carrying problems **WE HAVE SOLUTIONS!**

CABBAGE CASES

CABBAGE CASES, INC.
1166-C STEELWOOD

A User Interface Library Worth Its Weight in Gold!



FREE DEMO DISK! 1-800-54-BASIC

Save \$10,000 in programming time and join thousands of programmers using our **QuickWindows** user interface library. Give your programs a professional look and feel!

Works in both text and graphics modes through Super VGA. Create windows, menus, help systems, data-entry forms, and dialog boxes with icons, picture field inputs, pushbuttons, radiobuttons, scrolling list boxes, and much more. Comes with over 175 functions, 25 fonts, font editor, icon editor, and many example programs. Written entirely in assembly language. No royalties. Supports all regions of Microsoft compiled BASIC.

Gold Value at a Silver Price!
\$149

SOFTWARE INTERPHASE INCORPORATED

5 Bradley Street, Suite 203
Providence, RI 02908

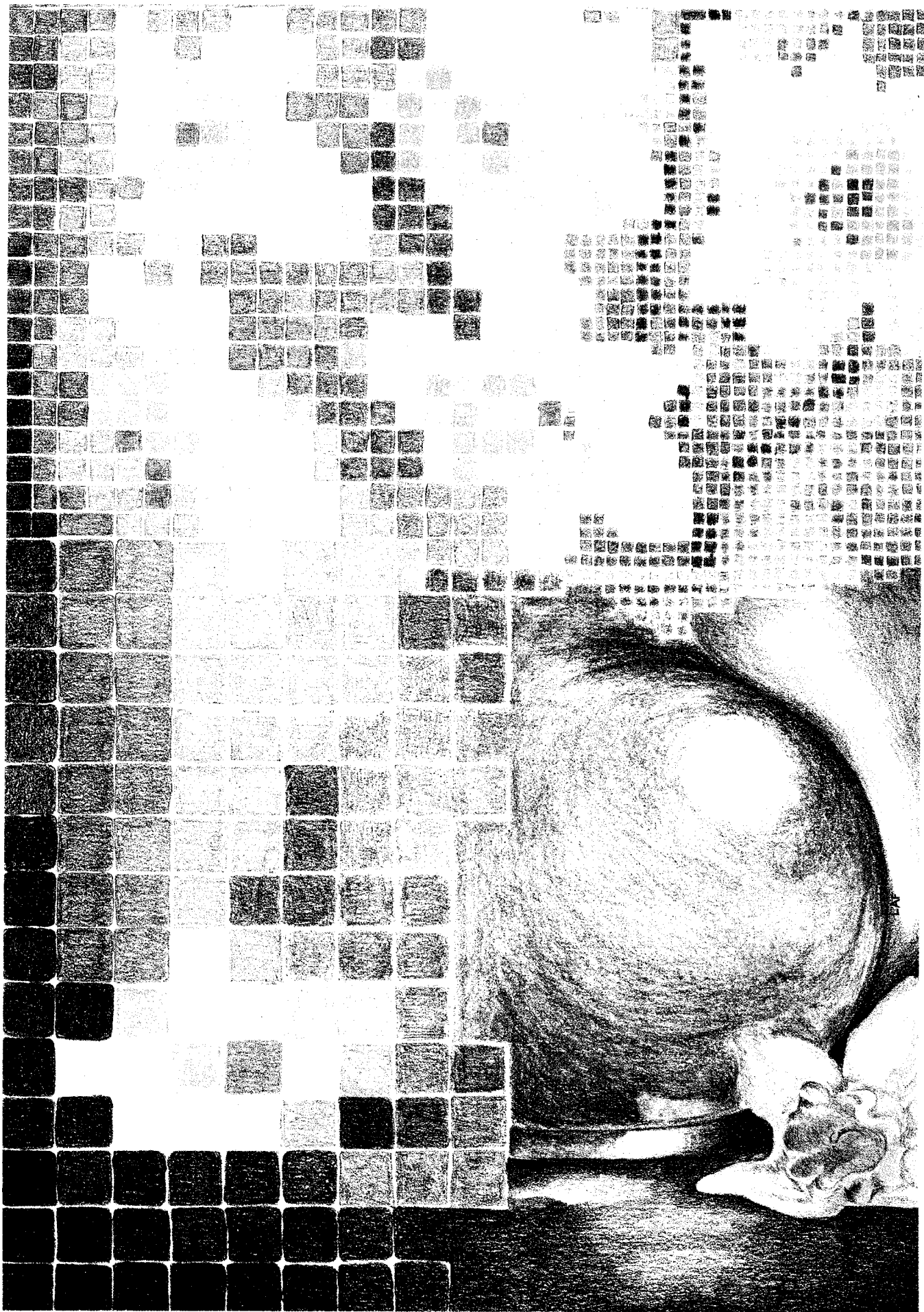


Illustration by Lisa Ann Ferry

Digital Image Processing

Software-based Digital Signal Processing

Image processing (IP) techniques have become widely used as a direct result of the computer boom of the '80s. The availability and extensive computing power of low-cost personal computer systems have made image processing tools both time and cost effective. As a result, easy access to tools of the once "mystical and closed realm" of the image processing laboratory has spawned new techniques and performance expectations. Today we take for granted the machine vision techniques employed by the automated assembly line procedure or the enhancement of the photograph from a deep-space probe we see on the evening news.

DIGITAL IMAGE REPRESENTATION

In the world of image processing, *image* defines a set of information which is a machine representation of some pictorial scene. So, our image refers to a two-dimensional light intensity function $i(x,y)$, with x and y denoting spatial coordinates, and the value of i at any point (x,y) proportional to the brightness (or grey or density level) of the picture at that point. Conventionally this "grey level" value increases as a function of the point intensity.

A digital image is one that has been discretized both in spatial coordinates and in brightness; within the computer realm, it is defined as a 2-D matrix whose row and column indices specify a point within the image and the corresponding matrix array element value specifies its grey level (or density) at that point. Although the size of these digital images varies as a function of the application, the typical image is a square array whose dimension is based on the integer powers of two. For our purposes, we will use images which are 256 x 256 in size and employ a grey scale ranging

from 0 to 255, with 0 representing black and 255 representing the brightest white. As such, a grey scale of this sort is easily accommodated using a single 8-bit word (byte) for each pixel thereby making our PC-resident image files 256 x 256 x 1, or 65536 bytes long. [Editor's Note: Software and image files for this article are available from the Circuit Cellar BBS and on Software On Disk #19. See page 107 for downloading and ordering information.]

BRIGHTNESS ADAPTATION AND DISCRIMINATION

In order to facilitate computer processing of visual data, the typical image is digitized in both the spatial and intensity domain. That is, we perform image sampling to set spatial coordinates and grey-level quantization to specify the brightness or light-intensity amplitude digitization. This information is usually stored in a square $N \times N$ sampled image array $i(x,y)$ with each element in the array containing the discretely quantized intensity level. As such, the resolution of an image is strongly linked to the number of these spatial points con-

tained within an image area and the range of levels that each point can display. As the number of grey levels decreases, the eye tends to integrate the image causing a mild form of "false contouring" to take place. But there is no "hard concrete rule" for what defines "good" image representation. The number of samples and grey levels required to produce a faithful reproduction of an original image depends on the image itself. Typically, for a 512 x 512 grid, 256-, 128-, and 64-level images are of acceptable quality. However, as a rule, the minimum acceptable system for general image processing work is a 256 x 256 grid with 64 grey levels (which is ideal for PC memory constraints).

IMAGE ENHANCEMENT

The primary objective of any enhancement technique is to process a given image such that the visual information contained within the image is presented in a more "application-oriented form" than the original. In other words, the information which is required from the image is "enhanced" and "isolated" so that it can be prop-

erly evaluated. Of course this means that the techniques used are specific to the type of information we are trying to enhance. Like any workshop, general image processing techniques are tools in the hands of the craftsman and can be used and combined to create a wide variety of applications.

Image enhancement usually means the sharpening or smoothing of an image to **bring** out desired image detail. Such techniques are often designed to manipulate the variations in the grey-level intensities-to stretch or compress an image's contrast (signal-to-noise ratio). Processing techniques of this sort are usually divided into two basic categories: frequency domain and spatial domain. In the first category, an image is decomposed into its frequency components, modified within that domain, and then reconstructed through an inverse transform into the enhanced image. The spatial domain techniques, on the other hand, refer to modifications which take place within the image plane itself, and constitute techniques that represent direct manipulation of an image's pixels. As it turns out, spatial techniques are usually faster and more efficient, and for all intents and purposes, they are as effective as the frequency domain techniques. The primary advantage of frequency domain **processing** lies in one's ability to be more precise.

MASK FILTERS

The spatial domain refers to the aggregate of pixels composing an image, that is, the light intensity values associated with their specific spatial coordinates within the image matrix. Spatial domain methods operate directly on these pixels through some sort of transformation process. Functions of this sort are usual defined by the expression,

$$p(x,y) = T[i(x,y)]$$

where $i(x,y)$ is the input image, $p(x,y)$ is the processed image, and T is an operator on i defined over some neighborhood of the spatial coordinates (x,y) . The approach most often used is

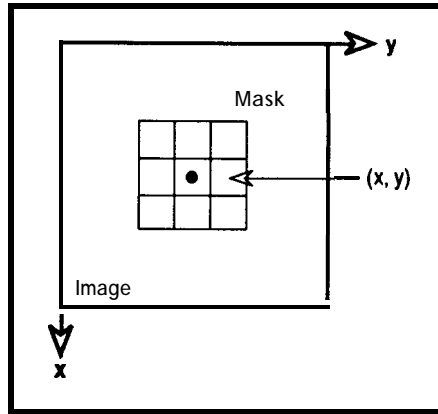


Figure 1 -A 3 x 3 neighborhood mask (kernel) about a point (x,y) in an image.

to define a neighborhood about (x,y) , a rectangular or square subimage area like that shown in Figure 1, that is moved pixel by pixel, operating at each (x,y) location throughout the whole image. Generally we let the values of $i(x,y)$ in a predefined neighborhood of (x,y) determine the value of $p(x,y)$ at those coordinates. This so-called "masking filter" is usually implemented using a 3 x 3 kernel or template (the T function) whose coefficients are chosen to detect a given property within an image.

To demonstrate how you can create such a mask filtering procedure on your PC, suppose we consider the image shown in Figure 2. Here we have a simple low-contrast video image taken from my office window. It is a picture of my Jeep "Betsy Bomber" imaged with a Panasonic CCTV camera and grabbed using my Image-Wise/PC video card. [Editor's Note: See "Image Wise/PC-The Digitizing



Figure 2-A low-contrast video image of my 'bomber truck' taken from my office window.

Continues" in *CIRCUIT CELLAR INK* issues 6-8 for details of the ImageWise/PC Video Digitizer.1 As you can see, the image has a low contrast but there are widely scattered points which are different from the background. In this case these points make up much of the edge detail. To isolate them, we can apply the mask shown in Figure 3. This mask is moved around the image and positioned at each pixel within the image where every pixel overshadowed by the mask area is multiplied by the coefficient within the corresponding mask, then summed. The total is then divided by nine, the renormalization constant. This is implemented in code in the following manner,

```
for (j=2; j < 256; j++) {
for (i=2; i < 256; i++) {
    result= a0*din[i-1][j-1]
    result= result + a1*din[i-1][j]
    result= result + a2*din[i-1][j+1]
    result= result + a3*din[i][j+1]
    result= result + a4*din[i+1][j+1]
    result= result + a5*din[i+1][j]
    result= result + a6*din[i+1][j-1]
    result= result + a7*din[i][j-1]
    result= result + a8*din[i][j]
    dout[i][j] = result/9
```

Here, the coefficients are defined by the variables a_0, a_1, \dots, a_8 (see Figure 3 for the mask coefficient locations) and the input, $i(x,y)$, and output, $p(x,y)$, images are defined by the arrays $din[][]$ and $dout[][]$, respectively. The action of this mask operating on the image in Figure 2 is shown in Figure 4. As you will notice, those points that represent a distinct variation from the background have been enhanced; mainly those points representative of sharp change with respect to the background. In this case they lie along the edges of imaged objects. So in effect, our masking filter acts like the frequency domain high-pass filter, since edges and other abrupt changes in grey levels are associated with high-frequency components. Using this mask has basically attenuated the low-frequency components within the 2-D frequency domain. If we desired to emulate a frequency low-pass filter using the above-described masking technique, we could define a kernel like that shown in Figure 5a. Here we have a low-pass averaging mask which is designed to smooth out the sharp high-frequency

structure. It can be used quite effectively to isolate the low-frequency components of an image (see Figure 5b.)

For further experimentation using mask filters, I refer you to the filtering program **FLTR.EXE**, available for your use through the Circuit Cellar BBS. Try a variety of different types of filters. **FLTR** will allow you to create, edit, and design your own masks.

HISTOGRAM EQUALIZATION . . . NONLINEAR CONTRAST STRETCHING

How many times have you viewed an image that has poor contrast, and no matter how you look at it, your eyes seem to have extreme difficulty picking out background details? What you need at this point is some technique which will stretch this background contrast in order to isolate those desired details. One such technique is called histogram equalization. It's an enhancement procedure that modifies the basic global description of the image by modifying an image's contrast in a nonlinear fashion. To best describe how such a technique can be implemented, let's discuss Listing 1 and determine how the input image (we will use the image in Figure 2) is processed in a step-by-step fashion.

In the first step we need to declare our working **environment**. Here I have defined two 256- x 256- "byte" arrays: `dintohold` the input image and `dout` to receive the output processed image. I have also defined an array `ihist` (0 : 255) (of 256 locations since our image can have 0-255 individual light-intensity grey levels or densities) which will contain the image's histogram and the array `pf` (0 : 255) which will contain the probability density transformation function.

The primary purpose of step 1 (Listing 1a) is to create the histogram of the input image, the so-called "probability density function." It is a linear profile that contains information about how many times an individual grey level value appears within the image. It is created by scanning the

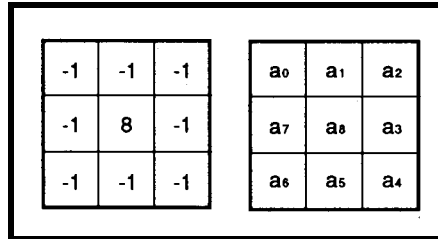


Figure 3-A mask for detecting isolated points different from a constant background.

entire image and adding up the total number of times each intensity level appears within the image and storing that number in the corresponding indexed array location of `ihist`.

Figure 6a is a histogram of the image in Figure 2. It provides us with very important information about the original image and it is often used by different 'II' tools. In this instance it enables us to determine the relative contrast within the test image. As you can see, the image of my "bomber" is low contrast and this fact is verified within its histogram. Note how the histogram profile is fairly low and drawn out. There are no prominent peaks which would indicate sharp structure. But, there is some structure within the image and our ultimate goal is to amplify these features.

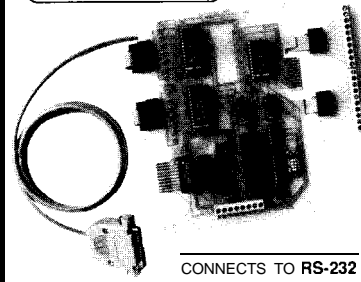
This amplification or contrast stretching can be effected by creating the "histogram transformation function" which will be used to map the original image over into the equalized output. Step 2 (Listing 1b) shows how this is done. The occurrence densities within the histogram array are summed and then renormalized by



Figure 4—The effect of the filter mask shown in Figure 3 operating on the image in Figure 2.

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS



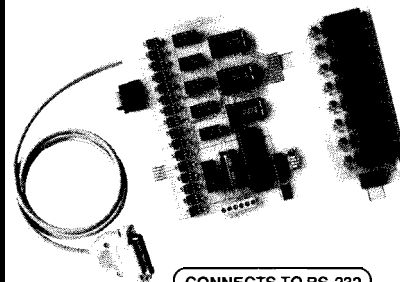
CONNECTS TO RS-232

AR-16 RELAY INTERFACE \$ 69.95

Two 8 channel relay output ports are provided for control of up to 16 relays (expandable to 128 relays using EX-16 expansion cards). Each relay output port connects to a relay card or terminal block. A variety of relays cards and relays are stocked, call for more info. RS-422 available (distances to 4,000 feet) PS-8 port selector may be used to control satellite AR-16 interfaces. (up to 16,384 relays)

RD-8 REED RELAY CARD (8 relays) \$ 49.95
RH-8 RELAY CARD (10 amp SPDT 277 VAC) \$ 69.95
EX-16 EXPANSION CARD (16 channel) .. \$ 9.95

ANALOG TO DIGITAL



CONNECTS TO RS-232

ADC-16 (16 channel) \$ 99.95

Input temperature, voltage, amperage, pressure, energy usage, energy demand, light levels, joystick movement and a wide variety of other types of analog signals. Inputs may be expanded to 32 analog or 128 status inputs using the AD-16 or ST-32 expansion cards. 112 relays may be controlled using EX-16 expansion cards. Analog inputs may be configured for temperature input using the TE-8 temperature input card. RS-422 available. PS-8 port selector may be used to connect satellite ADC-16 interfaces (up to 4,096 analog inputs/16,384 status inputs). For 32 relays, use RS-232 for satellites up to 50 feet or RS-422 for satellites up to 4,000 feet).

(terminal block and cable sold separately)
ST-32 STATUS EXPANSION CARD \$ 79.95

Input on/off status of relays, switches, HVAC equipment, thermostats, security devices, smoke detectors and other devices. The ST-32 provides 32 status inputs. (opto isolators sold separately)

TE-8 TEMPERATURE INPUT CARD .. \$ 49.95

Includes 8 solid state temperature sensors.

Temperature range is minus 78 to 145 degrees F.

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

- HIGH RELIABILITY... engineered for continuous 24 hour industrial applications. All IC's socketed.

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

- Use our 800 number to order free information packet. Technical Information (614) 464-4470.

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

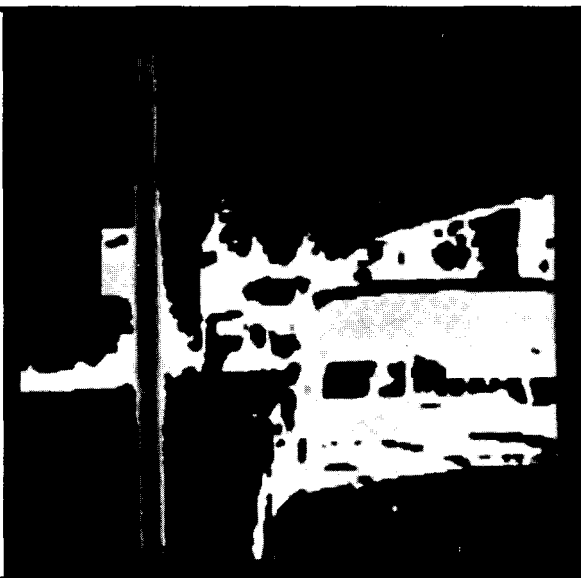
ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 604
Columbus, Ohio 43215

a)

1	1	1
1	1	1
1	1	1

b)

Figure 5—(a) A low-pass mask filter. (b) The effect of a low-pass filter acting on the image in Figure 2.



the image weight density—the total number of points within the image divided by the number of grey levels minus one (256 - 1). The output of this operation is then stored in the probability density transformation function array pf .

The contrast-stretched image is then created by mapping over the

original image using the transformation array. This is accomplished by determining the grey level for each pixel within the initial image, finding its associated transformation value within pf (at its corresponding indexed location), and then assigning this value to the output image at that spatial coordinate (see Figure 6b). This

procedure is demonstrated in step 3 (Listing 1c).

In Listing 1 I included two components not previously discussed. The first is the use of integer arrays. Much of the above coding involved the conversion of byte numbers, ranging from -128 to +127, over to "integer*2" numbers for the expression of our 0-255 grey level scale. Use of integer arrays minimizes required programming memory. Also I have not discussed the use of the clipping thresholds min and max . These are useful when you are attempting to isolate a specific range of mapped contrast values.

To gain a further understanding of what different histograms look like and how this technique can be used to modify your images, try `EQUAL.EXE` (included with the software on the Circuit Cellar BBS). `EQUAL` plots the histogram of the input image, then maps your image over into an equalized output. An example is shown in Figure 6b containing the histogram equalized image of my "bomber." Note how improved the background

BTK52 BASIC-52 TOOLKIT

The BTK52 is an intelligent front end for program development on the MCS BASIC-52 CPU. It reduces 8052 program development time substantially and can be used with any MCS BASIC-52 based target system. The BTK52 runs on any IBM-PC/XT or compatible.

- Program download from PC host to target
 - Program upload from target to PC host
 - BASIC program renumber utility, with "from," "through," "start," and "increment"
 - Full screen program editing
 - Single line editing with automatic error line number detection
 - Full on-line help facility
 - Transparent, adaptive line compression for full input line buffer utilization
 - All functions accessible with only one keystroke from the terminal emulator
- \$125

BXC51 8051/8052 BASIC COMPILER

- Fully compatible with code written for MCS BASIC-52 interpreter
 - now with integer, byte and bit extensions for code that runs more than 50 times faster than the MSC BASIC-52 interpreter
 - Full floating point support
 - In-line assembly language option
 - Compile time switch to select 8051/8031 or 8052/8032 CPUs
 - Includes Binary Technology's SXA-51 cross-assembler and Hex file manipulation utility
 - Compatible with any RAM or ROM memory mapping
 - Runs on IBM-PC/XT or compatible
- \$295

603-469-3232 • FAX 603-469-3530



Binary Technology, Inc.

Main Street . P O Box 67 . Meriden, NH 03770



GRAB IT!



Control Vision framegrabbers are high quality, versatile, affordable, real time imaging cards. The basis of many of today's OEM products, our boards excel in robotics, inspection, security and medical imaging.

- CV-512 Advanced framegrabber. Programmable resolution to 512 x 480, 256 grayscale four inputs, color killer, RGB outputs, input and output LUTS, dual independent field buffers. \$1,095
- CV-02 256 x 240 resolution, 256 grayscale, dual 595 inputs, output LUTS, RGB output, chroma filter, hardware cursor. \$595
- CV-03 64 grayscale board with many CV-02 features. \$469

Custom Software/System Integration
PC Board Design and PC Assembly

Control Vision

P.O. Box 596, Pittsburg, Kansas 66762

800/292-1160. 316/231-6647. Fax 316/231-5816



```

integer*1 din(256,256),dout(256,256)! input,output images
integer*4 ihist(0:255) ! histogram
real*4 pf(0:255) ! prob. transfer function
c
c find histogram of image
c
do j=1,256
do i=1,256
if(din(i,j).lt.0)then ! convert to 0-255, integer*2
k=256+din(i,j)
else
k=din(i,j)
endif
ihist(k)=ihist(k)+1 ! count occurrence of an
enddo ! intensity level
enddo

```

Listing 1 a- FORTRAN listing of a histogram equalization procedure. Step 1: Finding the image histogram,

```

total=0
do i=0,255
total=total+ihist(i)
pf(i)=total/257.0039 ! 256x256/255=257.0039
enddo
min=0 ! set threshold limits
max=255

```

Listing 1 b-Step 2: Creating the transformation function,

imaging and the front of my truck has become. The black areas at the top of the image represent the original "washed-out" bright clouds which were stretched past 255 and clipped.

IMAGE SHARPENING USING EDGE ENHANCEMENT

There are many times when the information required from an image

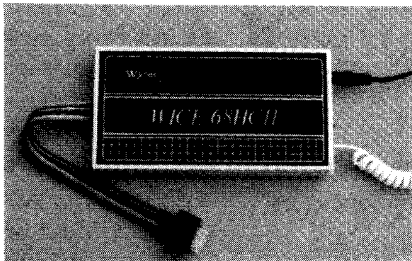
is directly related to the size and shape of the objects in view; characteristics which are often used to identify the objects. Usually this so-called "object segmentation" is implemented by enhancing the edges of the imaged objects and then tracing and segmenting each form into object categories based on their geometrical type. Procedures of this type are typical in most machine vision applications.

This edge enhancement task represents "image sharpening." It can be achieved in both a frequency and spatial manner. We have already discussed how this can be accomplished using the masking filter shown in Figure 4, which was the analog of a high-pass frequency filter. However, most image processors use the "differentiation technique" since it is usually more efficient. A technique of this sort is intuitively reasonable since integration involves averaging-an increasing of image blur; and it's natural to expect differentiation to have the opposite effect-the sharpening of the image.

68HC11

New Features:

64 K hardware breakpoints. Breaks on Address, address range, and Data RD/WR.



PC based real-time ICE. Easy to use, low cost & high performance. Complete development support for single-chip & expanded modes. Complex real-time hardware breakpoints. Symbolic debugger. Windowed user interface. Data watch windows for memory, registers & stack. On-line assembler, disassembler, single-step and trace commands. Logic analyzer trigger output. 115.2K bps RS-232C link.

30 day money back guarantee.

\$595.00*

(Introductory price)

WICE 68HC11 emulator

Regular price: \$795.00

* For first 2.5 customers only

52 PLCC to 48 DIP adapter \$55

Call: (708) 894-1440

Wytec

Suite 140

185C East Lake Street
Bloomington, IL 60108

Z8

WICE Z8 emulator (complete system) \$995

86C08,09 18 pin adapter w/analog comparators \$55

U.K. dist., Merwood Engineering, Tel: (0253)827787

Distributors for the Far East wanted

CAD Showdown Results!!

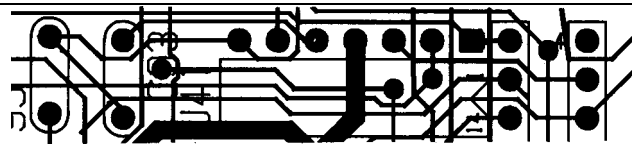
HIGH DENSITY EXPERTS!

Schematic Capture ♦ PCB Layout+ Autorouting

Top-rated DC/CAD out-routed the competition in the 1990 CAD Showdown. Routing the challenging benchmark on a double-sided board while competing routers used four to six layers, DC/CAD displayed the power and flexibility needed in a top-notch design package to tackle high density board jobs. This non-copy protected package with surface mount support includes:

- High capacity schematic capture
- Multi-strategy 1-mil parts autoplacer
- "1-mil" autorouting w/ripup & retry
- Thorough annotating design rule checker
- Full Z-way GERBER and DXF support
- Optional autoground plane support with cross-hatching
- Optional protected-mode version for 386 Users and much more!

CALL TODAY. Priced at \$495.



**DESIGN
COMPUTATION**

Rt. 33 Sberman Square Farmingdale, NJ 07727
(201) 938-6661. (201) 938-6662 (FAX)

Smart Software for Tough Board Design.

```

do j=1,256
do i=1,256
if (din(i,j).lt.0)then! convert each pixel value to an I*2
k=256+din(i,j)
else
k=din(i,j)
endif
if (k.le.min) then !clip according to set threshold limits
dout(i,j)=0
else if (k.ge.max) then
dout(i,j)=0
else
n=int2(pf(k)) ! map over to the equalized image
if (n.gt.127)n=n-256
dout(i,j)=n
endif
endif
enddo
enddo

```

listing 1 c-Step 3: Histogram linearization, creating the stretched image.

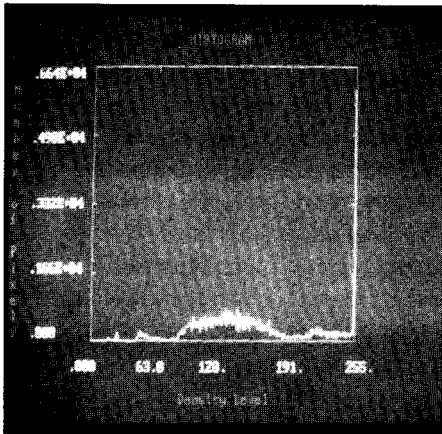


Figure 6—The screen on the left shows the histogram of the image shown in Figure 2 while the contrast-stretched histogram-equalization in on the right.

```

integer*2 din(255,256),dout(255,256) ! declare input output arrays
do j=1,255
do i=1,255
dout(i,j)=sqrt((din(i,j)-din(i+1,j+1))**2
+(din(i+1,j)-din(i,j+1))**2)
endif
enddo
enddo

```

listing 2-Robert's gradient method is easy to implement in just about any language.

The most commonly used method of differentiation in image processing applications is called the gradient procedure. Here, the gradient of an image $i(x,y)$ is defined as the vector,

$$G[i(x,y)] = \frac{di}{dx} i + \frac{di}{dy} j$$

with $G[i(x,y)]$ always pointing in the direction of the maximum rate of increase within the function $i(x,y)$. The magnitude of this vector is given by,

$$G(\text{mag}) = \sqrt{\left(\frac{di}{dx}\right)^2 + \left(\frac{di}{dy}\right)^2}$$

which equals the maximum rate of increase of $i(x,y)$ per unit distance in

the direction of G . This vector representation is the basis for several approaches to image differentiation. It is usually implemented in digital form by approximating the derivatives with differences. For example,

$$G[i(x,y)] = \sqrt{[i(x,y) - i(x+1,y)]^2 + [i(x,y) - i(x,y+1)]^2}$$

Similar results can be obtained by varying the direction of the pixel gradient relationship. One such technique which I often use is called the Robert's gradient method. It uses the cross differences between four adjacent pixels. G therefore becomes,

$$G[i(x,y)] = \sqrt{[i(x,y) - i(x+1,y+1)]^2 + [i(x+1,y) - i(x,y+1)]^2}$$

As such, the value of the gradient is proportional to the difference in the grey level between these adjacent pixels. The gradient therefore assumes relatively large values for prominent edges within the image and small values in regions where the image is fairly smooth. It is zero in regions of constant level.

Implementing such a procedure on your PC is extremely simple. You need only code the magnitude of the gradient function defined above, as shown in Listing 2. An example of the effect of this operation is shown in Figure 7. Here, the image in Figure 2 was passed through the above procedure and then thresholded. This thresholding was done mainly for display purposes and basically involved setting every pixel in the Robert's enhanced image greater than 15 to 255 and all pixels less than or equal to 15 to zero. The primary use of such a thresholding technique is to isolate and emphasize a specific image feature. In this case, the enhanced edges.

IMAGE RESTORATION

As in all image processing, the ultimate goal is to improve the overall quality of the input image. But, unlike most enhancement procedures, the typical "image restoration" technique is designed to reconstruct or recover an image which was degraded due to some observable phenomenon. A perfect example would be a defocused camera image which was taken with the camera's lens improperly adjusted. This is what is called the domain of "image reconstruction."

Suppose we were called upon to make corrections to some space telescope images which were distorted due to poor focusing (like the real ones distorted because of misaligned mirrors). For example, suppose that the optical path lengths between components of the space telescope's optical data acquisition system were changed due to some extraneous circumstance. As a result, the output of



Figure 7—The Robert's edge enhancement of Figure 2 with every pixel > 15 set to 255 and all pixels ≤ 15 set to 0.

the system would be slightly defocused, resulting in a reduction in image quality. It's obvious that adjustments to instrumentation under these conditions are impossible. So this would require that some "restoration" procedure be applied to recover such "expensive" images. This can and has been accomplished by treating the space image as a composite of the desired/required image and an associated blurring function.

A reasonable implementation of such a procedure can be achieved by assuming that the telescope's optical system can be modeled as a spatially invariant linear system with the final space image (I_i) being defined as the convolution of the our desired image (I) and a blurring function (F) resulting from the maladjustment of the lens components, such that

$$I_i(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x-t,y-r) F(t,r) dt dr$$

The effect of the system blurring function, is seen by mapping the convolutional relationship of the above equation over into one of multiplication via a Fourier integral transform,

$$I_i(u,v) = I(u,v) \times F(u,v)$$

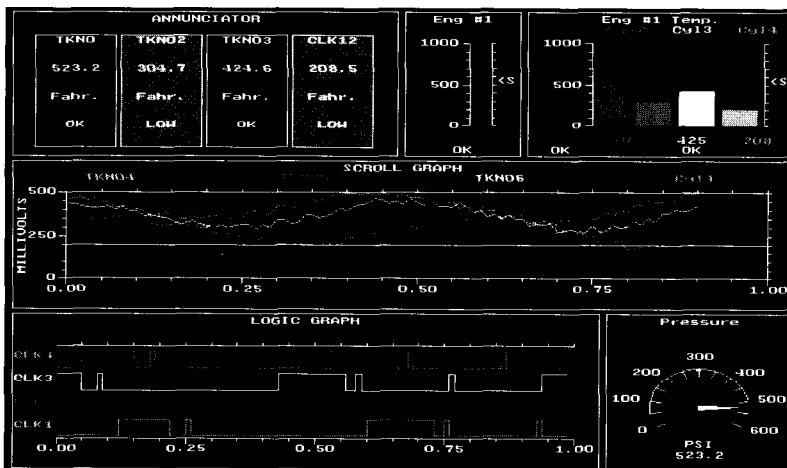
where $F(u,v)$ is the two-dimensional frequency response of the system blurring function, and $I_i(u,v)$ and $I(u,v)$ are the transforms of the space image and the required "restored" image respectively.

SO WHERE DOES THIS ALL LEAD?

An image can be corrected for basic blurring if the form of the blurring spread function (F) is known. Correction of this sort can easily be achieved through a simple deconvolution process which is best implemented in the frequency domain. So, before we continue with the description of the deblurring restoration method, it is important that you be aware of the basic features of 2-D fast Fourier transform techniques. If you're happy with black-box implementa-

Real-Time Graphics and Measurement/Control Tools

Programmers Tools for Microsoft C, Turbo C, C++ and Turbo Pascal



Call or Write for Demo Disk

Fast Real-Time Graphics

Real-time graphics routines for scrolling graphs, sweep graphs, process control bargraphs, annunciator panels, 3 types of meters and general graphics text displays. Real-time mouse routines are also included. The routines are optimized around the graphics primitives which come with the respective compiler.

Powerful Measurement and Control

PID Control (position and velocity algorithms), Thermocouple linearization for the 15 standard TC types (B, BP, BN, E, J, JP, JN, R, K, KP, KN, S, T, TP, TN), Thermocouple curvefitting for any temperature range, Fourier analysis routines including 16K point FFTs.

Source Code and Royalty Free

No additional charge for source code, use these routines in your programs without worrying about royalty fees. No other third party drivers are necessary.

Each version is \$200

Please specify compiler when ordering.

Price includes libraries, source code and a 250 page manual.

Mastercard, Visa accepted. Shipping charge \$5 within USA, \$9 Canada and \$25 elsewhere.

Quinn-Curtis, 21 Highland Circle, Needham, MA 02194 USA Tel. 617/449-6155 FAX 617/449-6109

Reader Service # 189

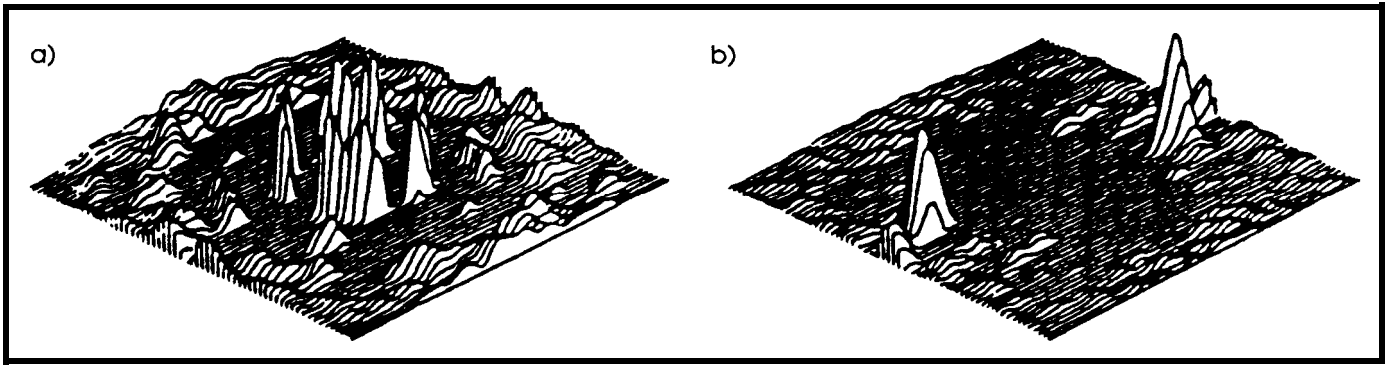


Figure 8—(a) The power Cepstrum of an image that contains defocus blur. In this picture the power Cepstrum has been inverted and clipped at zero to disclose negative spikes that indicate the out-of-focus condition. (b) The inverted clipped power Cepstrum of an image that contains motion blur. The twin peaks indicate the motion blur; the separation and orientation of the peaks about the center define the severity and the direction of the motion.

tion of the “deblurring restoration procedures,” I refer you to three application programs `FWD_FFT .EXE` for doing forward 2-D FFTs, `INV_FFT .EXE` for doing inverse 2-D FFTs, and `LAMP .EXE` which is designed to create the log amplitude of a complex FFT for display purposes.

RESTORATION

The restoration of the space image (correcting for lens blur) can be accomplished by a simple deconvolution of F from I_i with the defocusing spread function $F_b(x,y)$ being approximated by a cylinder whose radius R depends on the focus defect extent. In most cases this function in the spatial domain has the form,

$$f_b(x,y) = \begin{cases} 0 & \text{for } \sqrt{x^2+y^2} > R \\ = \frac{1}{\pi R^2} & \text{for } \sqrt{x^2+y^2} \leq R \end{cases}$$

An effective procedure would therefore, (1) create a small filled circle

at the center of a blank image of the appropriate radius, (2) then find the forward FFT of the space image and the image in (1) using `FWD_INV.EXE`, (3) then submit each to `DECONV.EXE` to deconvolve the effects of F from I_i , and finally (4) submit the output of `DECONV.EXE` to `INV_FFT.EXE` to convert the results back into the spatial domain to create I .

Suppose instead the space images where taken with an exposure time that was long compared to the movement of some celestial object displaying linear motion. This would result in a blurred smear along the path of the object. To deconvolve this effect, a linear motion spread function can be approximated by a rectangle whose orientation and length are indicative of the direction and extent of the blur. In the case of a blur of length $2d$,

$$F_m(x,y) = \begin{cases} 0 & \text{for } y \neq 0, -\infty < x \leq +\infty \\ = \frac{1}{2d} & \text{for } y = 0, -d \leq x \leq +d \end{cases}$$

OFCOURSE LIFE'S NEVER SO EASY...

Well this is all well and good, provided you can anticipate the form of the blurring function. However, more often than not, restoration attempts require some analysis of the types of problems present before blurring functions can be constructed. The actual form of the defocus and motion blurring functions are described above, but the procedure describing how the defocus radius R and/or the motion displacement d and its phase angle are measured must be addressed for any serious “restoration attempt” to succeed. They are usually determined by computing the Cepstrum of the degraded image I_i . This Cepstrum is defined as

$$C_e(p,q) = T^{-1} \{ \log |I_i(u,v)| \}$$

where T^{-1} denotes the inverse Fourier integral transform. From our basic definitions of the FFT, it follows that

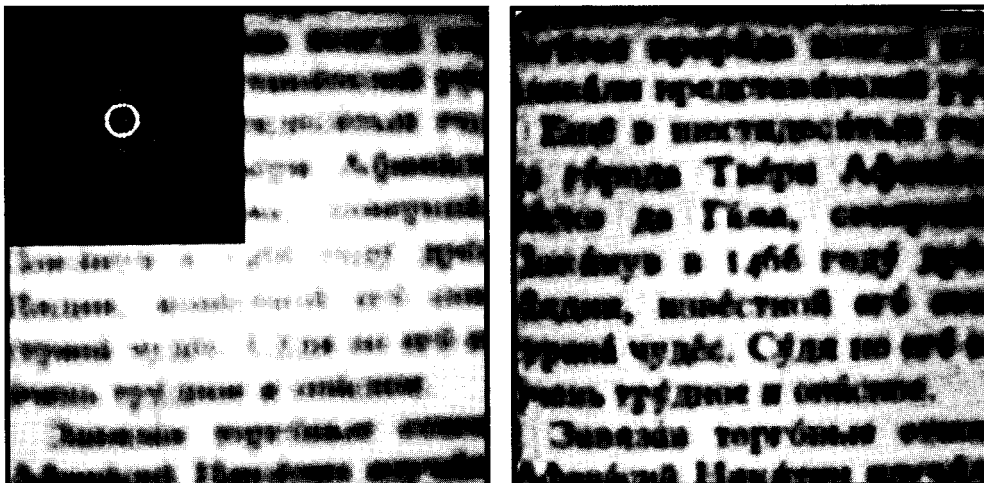


Figure 9—Text from a Soviet school book imaged by a defocus camera system is shown on the far left. The measured power Cepstrum is displayed in the upper left hand corner with its spiked ring indicating the radius of our cylindrical point spread function. Using a straightforward deconvolution technique, the effect of this blurring response function, $F_b(x,y)$, was removed with the results being displayed on the near left.

**FAST COMPLETE
ACCURATE
DRAM TEST
DIPs - SIMMs - SIPs**



64K - 256K - 1M - 4M

RAMSTAR

1ns. RESOLUTION

ACCESS SPEED VERIFICATION

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

AUTO-LOOP

Continuous Test 6.25 Mbits/sec.

ADAPTERS:

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

NEW

NTX ADAPTER \$149.00
Tests 64 Pin Dual-Edge
LaserWriter Type SIMM's.

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

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

**FREE RAMFACTS
DRAM NEWSLETTER**

1-800-RAMSTAR

COMPUTERDOCTORS
9204-B Baltimore Boulevard
College Park, Maryland 20740

MADE IN U.S.A U.S. PATENT No. 4.965799

Reader Service # 1:

the convolutional effects of the blurring functions are additive within the Cepstral domain, where neglecting the effects of noise,

$$C_{I_i}(p,q) = C_i(p,q) + C_{F(m \text{ or } b)}(p,q)$$

For motion blur of length d at a phase angle of ϕ degrees off the horizontal, $F_m(u,v)$ has the form $\sin(\pi df)/(\pi df)$ where $f = u \cos \phi + v \sin \phi$, with the higher frequencies being attenuated. The alternate lobes of $\sin(\pi df)/(\pi df)$ produce phase shifts in pi radians. For defocus blur, the frequency response is of the form $J_1(Rr)/(Rr)$ (the first-order spherical Bessel function), where R is the radius of the blur function and

$$r = \sqrt{u^2 + v^2}$$

It is similar in shape to the motion blur response but circularly symmetric.

The periodic zeros in $F_m(u,v)$ and $F_b(u,v)$ lead to large negative spikes in $C_{F_m}(p,q)$ and $C_{F_b}(p,q)$. For example, the zeros of motion blur fall along lines spaced $1/d$ apart. This periodic pattern results in a series of negative spikes within $C_{F_m}(p,q)$ spaced a distance d apart, radiating outward from the origin. The amount that this spike line has rotated about the origin indicates the direction of the motion blur (see Figure 8b). For defocus blur, the circular symmetry of the blur function results in a series of rings of negative spikes within the Cepstrum. These rings are assumed to reflect the radi-

ally periodic zeros in $J_1(Rr)/(Rr)$, even though that is not strictly true (see Figure 8a). The radius of the first ring within $C_{F_b}(p,q)$ equals the diameter of the spread function.

It is my experience that attempts to locate the spikes of $C_{F_m}(p,q)$ and $C_{F_b}(p,q)$ are often frustrated due to the effect of noise and the overlying structure of $C_i(p,q)$ and $C_{F_b}(p,q)$. It is with this in mind that an averaging scheme, based on the spatial invariance assumption, should be adopted. In order for the Cepstral spikes to be identifiable, $I_i(x,y)$ must be subdivided into 100 image subsections and an average must be taken over the square of the magnitude of the Fourier transforms before projection into the Cepstral domain. In this way, the $|F_m(u,v) \times F_b(u,v)|^2$ will be present within the average, but contributions from noise will be averaged to a more subtle background.

Figures 9 and 10 demonstrate the effectiveness of the Cepstral technique. In Figure 9a, text from a Soviet school book has been imaged by a defocused camera system. The measured power Cepstrum is displayed in the upper left-hand corner with its spiked ring indicating the radius of our cylindrical point spread function. Using a straightforward deconvolution technique, the effect of this blurring response function, $F_b(x,y)$, was removed with the results being displayed in Figure 9b. Note that the date "1466," written just below the lower right

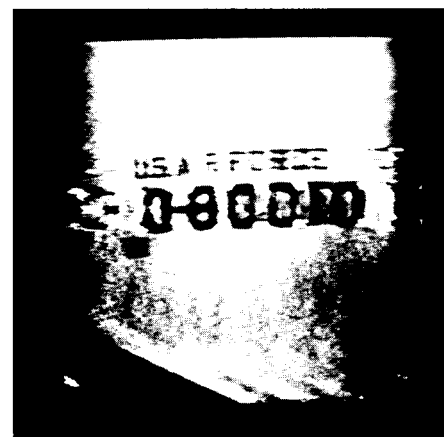
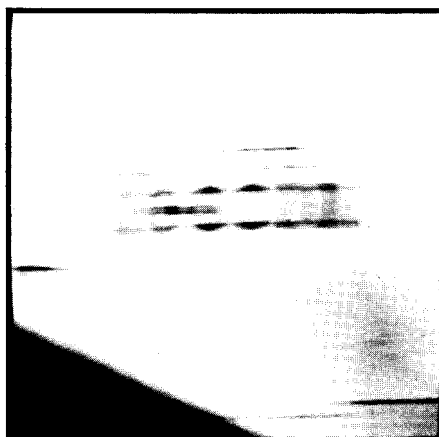


Figure 10—On the left is a photograph of the tail of a moving aircraft with the lettering smeared and unreadable. This smearing was corrected by measuring the extent of the blur and its associated phase angle using the Cepstral technique. Its spread function was then created and deconvolved from the original image to get the image on the right.

corner of the power Cepstrum plot, can now be read without too much difficulty.

For linear motion, a similar exercise was undertaken. Consider the photograph of the tail of a moving aircraft shown in Figure 10a. Notice how the lettering is smeared and unreadable. This smearing was corrected by measuring the extent of the blur and its associated phase angle using the above Cepstral technique. Its spread function was then created and deconvolved from the original image to get the image in Figure 10b. We can now read the tail labeling, "U.S. AIR FORCE O-80010."

Incorporation of the Cepstral technique into any "image restoration" procedure is simple and straightforward. It enables one to use processing tools to determine the extent and type of degradation which must be removed.

ONWARD AND UPWARD

Well, so ends my all too short description of Image Processing. I hope I've given you a quick view of the types of things that can be done using the tools of IP. Personally, I find myself applying these tools extensively in my work. I like to think of myself as the keyboard artist complete with my palette of paints and brushes, toothpicks, tape and glue, as well as the hammer for effective fitting. No matter how you apply IP tools, whether creating fantastic scenes or just correcting data, the application of these tools is unique to each application. This makes image processing one of the truly fundamental art forms. ❖

Chris Ciarcia has a Ph.D. in experimental physics and is currently working as a staff physicist at a national lab. He has extensive experience in computer modeling of experimental systems, image processing, and artificial intelligence. Chris is also a principal in Tardis Systems.

IRS

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

REFERENCES

1. Gonzalez, R.C. and P. Wintz, 'Digital image Processing.' Addison-Wesley Pub., Reading Mass, 1987.
2. M. Cannon, 'Blind Deconvolution of Spatially Invariant Image Blurs With Phase.' IEEE Trans. Acoust., Speech, and Signal Processing. Vol. ASSP-24. no. 1, 58-63, Feb. 1976.
3. J.C. Dainty and R. Shaw, 'Image Science,' Academic Press, New York, 1974.
4. T.G. Stockman, Jr., T.M. Cannon, and R.B. Ingebreetsen, 'Blind Deconvolution Through Digital Processing,' Proc. IEEE (Special Issue on Digital Signal Processing), vol. 63, 678-692, April 1975.
5. W. Lukosz, 'Uebertragung Nicht-negativer Signale durch Lineare Filter.' Opt. Acta, vol. 9, 335-364, 1962.
6. M.M. Sondhi, 'Image Restoration: The Removal of Spatially Invariant Degrada-tions,' Proc. IEEE (Spec. Issue on Digital Signal Proc.), vol. 60, 842-853, July 1972.
7. B. Bogart, M. Healy, and J. Tukey, 'The Frequency Analysis of Time Series for Echoes.' In Proc. Symp. Time Series Analysis, M. Rosenblatt, Ed. New York: Wiley, 1963, ch. 15.
8. E. Oran Brigham. 'The Fast Fourier Transform' Prentice-Hall, Inc, Englewood Cliffs, NJ, 1974, 164.

C compiler
software simulator
source level debugger
macro assembler—development board
IBM-PC MSDOS—Sun, Apollo,
Dec UNIX—VAX VMS—Apple Macintosh

AVOCET
SYSTEMS, INC.

The Source For Quality Embedded-System Tools

Avocet Systems, Inc., 120 Union St., P.O. Box 490, Rockport, ME 04856
In Maine, or outside U.S., call (207) 236-9055
TLX: 467210 Avocet CI / FAX: (207) 236-6713

Call today for free catalog 1-800484500

FEATURE

ARTICLE

Steven E. Reyer

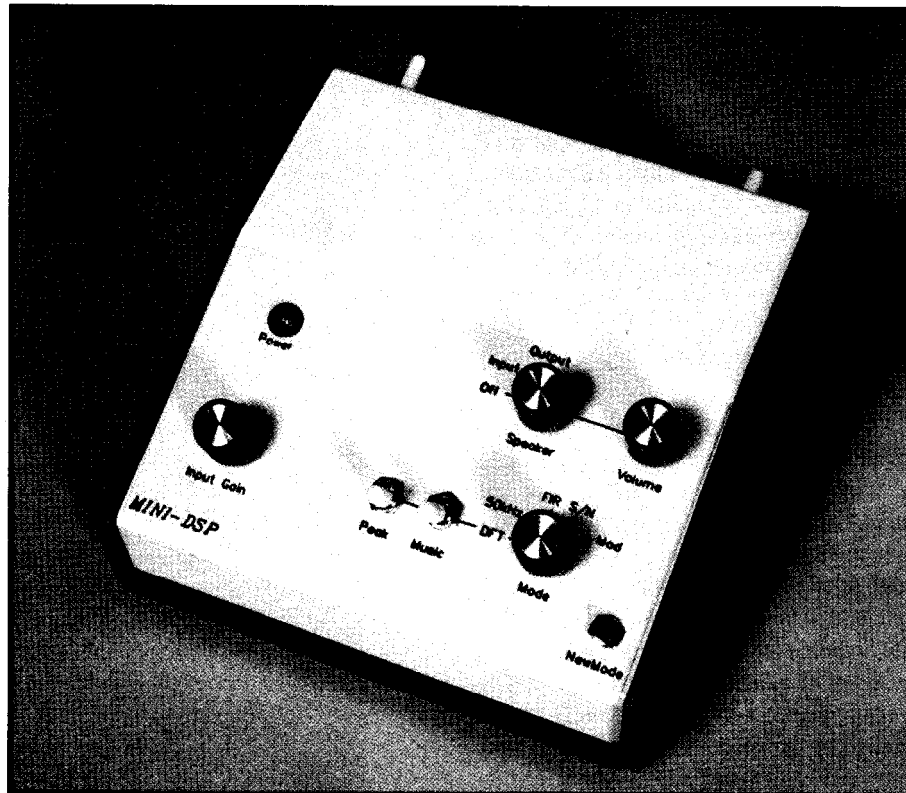
Mini-DSP

A Digital Signal Processor Experimentation Unit

Having digital computers manipulate analog signals sounds like the classic “apples and oranges” situation. Yet, whole areas of theory, algorithms, and hardware techniques are centered around it. This hasn’t happened as recently as might be thought. In fact, some digital signal processing (DSP) theories date back to the 1940s, while newer ideas have developed in just the last few years. The reason DSP is a buzzword now is that designers finally have access to the type of hardware that can crunch the numbers fast enough to make real-time systems practical. Before that, applications were largely limited to off-line processing on mainframes and minicomputers. Real-time processing, available since the early 1980s, opens up incredible new applications.

What do these systems do? A main application is to have the computer (typically a microprocessor system) “process” speech, music, and other analog signals. As discussed by others, this may include filtering, spectral analysis, correlation, and more. [1][2] A spectrum analysis can be implemented so quickly that immediate control decisions can be made from the results. Filtering can be shared between many analog signals, so one processor can replace racks full of analog componentry. A precision audio equalizer could be made from a handful of components, and would be easily upgradable. Even the familiar CD music player is an application of DSP, although it’s almost a mundane application of the art.

The trend toward this digital processing is good, since it yields systems that can often outperform



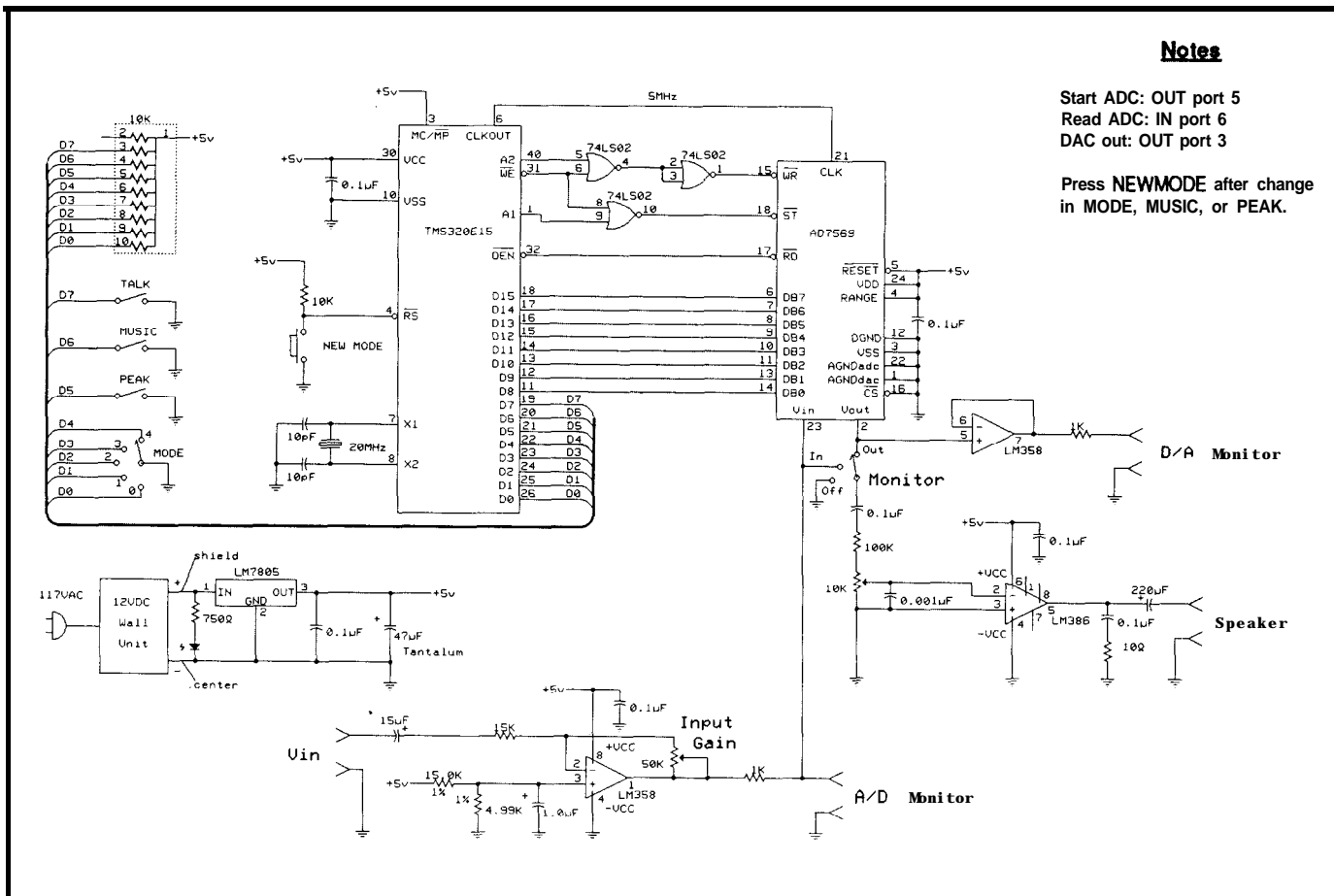
analog designs, require no “tweaking,” and will maintain the same high performance for years, since component drift is of no concern. Since the signal processing techniques exist mainly in software, a new EPROM means completely new functions.

The bad part is that the theory behind DSP is heavily mathematical. Any experimentation with DSP principles might follow an awfully long learning curve. To shortcut this, I developed the Mini-DSP system. It’s a stand-alone DSP system, incorporating both DSP hardware and software algorithms, all programmed, ready to run. To experiment with DSP means simply turning the Mini-DSP on, se-

lecting from many popular preprogrammed DSP algorithms, and hearing and seeing the results. It works in real time on music, other audio, or signal generator inputs. Since the software is in EPROM, the user could further experiment on his own. The Mini-DSP stands alone, not needing a personal computer.

THE MINI-DSP SYSTEM

The goals in the design were to implement many common DSP algorithms with a minimal number of non-exotic chips, at a reasonable cost (under \$100). The result was a 5-chip design (Figure 1), able to run in the following



Notes

Start ADC: OUT port 5
Read ADC: IN port 6
DAC out: OUT port 3

Press NEWMODE after change
in MODE, MUSIC, or PEAK.

Figure 1 — The schematic for the Mini-DSP shows the use of the TMS320E15 and AD7569 with only a handful of components around them.

modes: Spectrum Analysis or Discrete Fourier Transform (DFT); Sampling, Aliasing, Zero-Order-Hold Experiments; 70-tap Digital FIR Low-pass Filter; Signal/Noise Experiments; Double Sideband Modulation (Speech Scrambling).

Two key chips form the heart (brains?) of the design—the Texas Instruments TMS320E15 DSP, and the Analog Devices AD7569 Analog I/O System. The two chips were not deliberately designed to work together, but do—and well. Below we'll look at the digital and **analog** circuitry separately, although designing such a system means looking at both simultaneously, while incorporating algorithm considerations.

ANALOG CIRCUITRY

Generally, a DSP system requires an antialiasing filter ahead of the sampling circuit to limit the highest frequency present to less than one-half the sampling frequency. The de-

cision was made to omit this, to allow experiments with aliasing to be observed, and to allow maximum flexibility. If external filtering is desired, it can be added. Likewise, no reconstruction filter on the output is included. With the sample rates chosen, many reconstruction frequency components are either inaudible, are masked by

louder audible components, or are actually desired, as in the DFT case. Maximum flexibility is again the rule.

The AD7569 is a perfect component to use in such a system. It contains a sample/hold device (S/H), analog-to-digital converter (ADC), and digital-to-analog converter (DAC). It runs on a single +5-volt

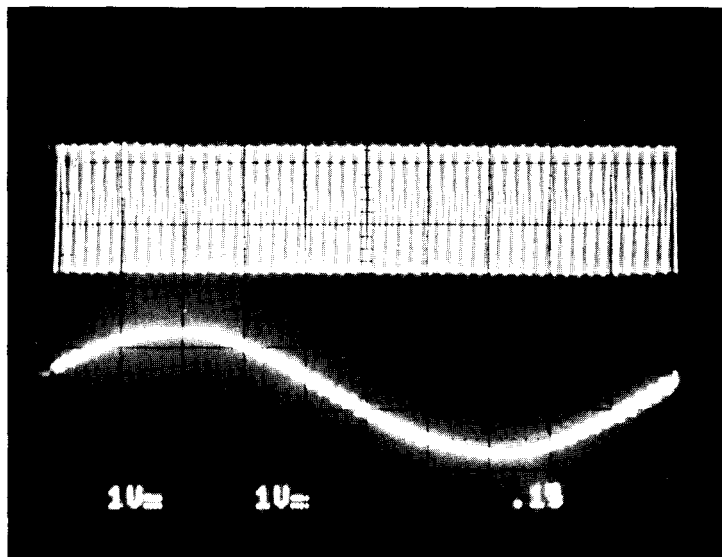


Figure 2-A 49-kHz sine wave (top) sampled at a 50-kHz rate yields an audible 1-kHz sine wave (bottom).

supply, and will convert an analog voltage to an 8-bit sample in 2 μ s. Its internal logic can be clocked by a 5-MHz signal which, fortunately, is available on the TMS320E15. The AD7569, with its unipolar supply, works with analog inputs and outputs centered around +1.25 volts, and not exceeding the range 0 to +2.5 volts.

An op-amp circuit, using an LM358 with precision resistors, provides offset and gain adjustment. The LM358 is a good choice here, since it operates well off a single +5-volt supply, able to produce outputs down to zero volts. A 1 k isolation resistor feeds the scope jack. The input analog processing has a frequency response from 1 Hz to over 100 kHz, so audio will pass well, and experiments can take place with high-frequency aliasing. The DAC output similarly feeds a scope jack with the other half of the LM358 as a buffer. This signal, with DC blocking, drives a typical LM386 power amp and speaker.

The only other analog circuit is the power supply, a 12-volt "wall wart" regulated down to +5 volts. The only voltage required internal to the system is +5 volts.

DIGITAL CIRCUITRY

The digital circuitry is a bit strange. Switches directly on a bus? Yes. With care it'll work just fine. We'll get to that.

The AD7569 is connected to the eight most-significant bits of the 16-bit data bus, with ADC and DAC data flowing over these bits. The AD7569 puts out binary data (0 volts = 00H, 2.5 volts = FFH) which has to be changed to 2's complement in software by complementing the MSB. Control is provided by loosely decoding port numbers to provide ADC start, ADC read, and DAC write signals. The 'E15 is run off a 20-MHz crystal, and has a reset switch which starts new modes.

Now for the switches. The 'E15 will only output to the data bus during two instructions: TBLW (table write to external RAM), and OUT (port output). No TBLWs are used, and if, during OUT, only zeros are written to the least-significant bits on the bus, the switches

```

* MODE 1 is a straight-through A/D - D/A system
* test at 50-kHz sample rate

MODE1      LDPK      0           Page 0
MODE1A     ZAC       X           Make sure X=0
           SACL      X           Save
           OUT       X, PA5      Start ADC
           CALL      WAIT2       Wait a while
           IN        X, PA6      Get sample
           LAC       X           Get X and mask
           AND       MASK        Mask off LS byte
           SACL      X           Save
           OUT       X, PA3      To DAC
           B         MODE1A      Back for more

* For use in MODE1, total wait is 20 microseconds
* (100 states): 15 in MODE1, 85 here

WAIT2      LACK      27          Counter
WAIT3      SUB       ONE        Count down
           BNZ       WAIT3       Finished?
           NOP                          Padding
           RET                          Finally done

```

Listing 1 - A simple program showing part of the instruction set of the TMS320E 15.

can never short circuit a logic high level bus line to ground. When an IN (port input) is performed, the least-significant bits will indicate switch position. Talk about a cheap and dirty input port! The 10k resistors are in a resistor pack. The 0.1- μ F bypass capacitors are important, since the digital and analog circuitry are in close electrical and physical proximity.

THE DSP ALGORITHMS

All the textbook learning in the world can still benefit from actually

hearing and seeing the results in real time. This philosophy led to the choice of algorithms. The DSP techniques included are good ones to experiment with when learning about DSP. They also serve to demonstrate the extraordinary capabilities of even a small DSP system.

1. Sampling, Aliasing, Zero-Order Hold

This software example is a simple loop that takes a sample from the ADC and outputs it to the DAC. At the 50-kHz sample rate, with 8-bit samples,

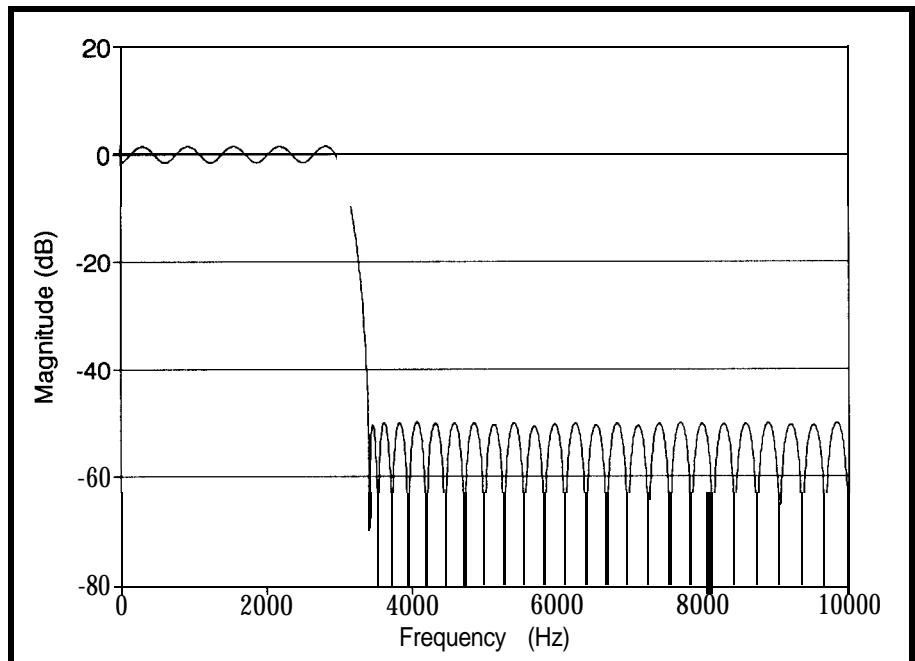


Figure J-Magnitude frequency response of the 70-tap FIR digital low-pass filter

audio is easily reproduced beyond human hearing range. The interesting part is to note the differences **between** input and output on a scope and/or spectrum analyzer, verifying the effects of sampling. Replicas of the input audio will appear about multiples of 50 kHz. The zero-order hold effect of the DAC reduces the level of the replicas. Any input signal above 25 kHz will "alias" back into the range

below 25 kHz. For instance, a 49-kHz input will produce an audible 1-kHz tone (Figure 2).

The simple program shown in Listing 1 serves to exhibit some features of the instruction set.

2. Reduce & Precision Sampling

This is a sampling example, as above, but now the precision is gradually reduced from 8 bits down to 1 bit.

Five seconds at each precision audibly reinforces the theory of six decibels per bit **signal-to-noise ratio (SNR)**. At or below 5 bits, the noise (hiss) is very obvious. But the surprise is just how recognizable is a 2-bit or even a 1-bit representation. The software then reverts back to 8 bits, and repeats.

3. FIR Digital Filter

One of the most common applications of DSP, and one that the 'E15 architecture is best suited for, is finite impulse response (FIR) digital filtering. Almost any analog filter can be outperformed by a digital filter. And the FIR type, which is equivalent to numerical convolution, can have perfectly linear phase, as well. The filter programmed has 70 taps, a sample rate of 20 kHz, a passband with 1.28-dB ripple out to 3000 Hz, and a stopband down 50 dB from 3400 Hz out to half the sampling frequency at 10 kHz. The filter's frequency response appears in Figure 3.

The 'E15 really shines here, doing 70 multiplies and 70 adds in the intersample interval of 50 μ s. Actually, the chip is coasting, with

FINITE IMPULSE RESPONSE (FIR)
LINEAR PHASE DIGITAL FILTER DESIGN
REMEZ EXCHANGE ALGORITHM

BANDPASS FILTER

FILTER LENGTH = 70

```

***** IMPULSE RESPONSE ***** *2^15=
H( 0) = -0.39324740E-02 = H(69)  -129
H( 1) = -0.12983940E-01 = H(68)  -425
H( 2) = -0.208003703-01 = H(67)  -682
H( 3) = -0.26237300E-01 = H(66)  -860
H( 4) = -0.22017620E-01 = H(65)  -721
H( 5) = -0.10052870E-01 = H(64)  -329
H( 6) =  0.472867303-02 = H(63)   155
H( 7) =  0.13156240E-01 = H(62)   431
H( 8) =  0.10599380E-01 = H(61)   347
H( 9) = -0.41856990E-03 = H(60)   -14
H(10) = -0.10643430E-01 = H(59)  -349
H(11) = -0.11533010E-01 = H(58)  -378
H(12) = -0.201812903-02 = H(57)   -66
H(13) =  0.98988650E-02 = H(56)   324
H(14) =  0.13563880E-01 = H(55)   444
H(15) =  0.48546820E-02 = H(54)   159
H(16) = -0.939682903-02 = H(53)  -308
H(17) = -0.164733603-01 = H(52)  -540
H(18) = -0.87726440E-02 = H(51)  -287
H(19) =  0.84543290E-02 = H(50)   277
H(20) =  0.20221020E-01 = H(49)   663
H(21) =  0.143755103-01 = H(48)   471
H(22) = -0.65115430E-02 = H(47)  -213
H(23) = -0.25056310E-01 = H(46)  -821
H(24) = -0.22764450E-01 = H(45)  -746
H(25) =  0.27096650E-02 = H(44)    89
H(26) =  0.31907500E-01 = H(43)  1046
H(27) =  0.36872510E-01 = H(42)  1208
H(28) =  0.52376820E-02 = H(41)   172
H(29) = -0.43869390E-01 = H(40) -1437
H(30) = -0.67281310E-01 = H(39) -2205
H(31) = -0.26753120E-01 = H(38)  -877
H(32) =  0.80170720E-01 = H(37)  2627
H(33) =  0.21102490E+00 = H(36)  6915
H(34) =  0.30049300E+00 = H(35)  9847

```

	BAND 1	BAND 2
LOWER BAND EDGE	0.0000000	0.1700000
UPPER BAND EDGE	0.1500000	0.5000000
DESIRED VALUE	1.0000000	0.0000000
WEIGHTING	1.0000000	50.0000000
DEVIATION	0.1584962	0.0031699
DEVIATION IN DB	1.2778920	-49.9790200

Figure 4—The 70 coefficients used in implementing the FIR digital low-pass filter.



Develop Image-based applications with Victor Image Processing

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 device control, image processing, display, and TIFF/PCX/GIF/PW/PIF/bin file handling routines.

Your software can have features such as: live video on VGA, resize and zoom, display multiple images with text and graphics. Image processing functions include brightness, contrast, matrix convolution filters: sharpen, outline, etc, linearization, equalization, math and logic, overlay, resize, flip, invert, mirror. Size/number of images limited only by memory. Display on EGA/VGA up to 1024x768x256. Hold multiple images in expanded memory and print half-tone3 at any size on laser printers.

And, to get you up and running quickly, we've introduced our popular Zip Image Processing software for rapid testing and prototyping of image processing and display functions.

Victor and Zip support ImageWise and other popular video digitizers. Victor supports Microsoft C, QuickC, and Turbo C, no royalties. Victor Library \$195. Source available.

ZIP Colorkit creates color images

It's easy to produce stunning color images with the ZIP Colorkit -- capture three images with a gray scale video digitizer using red, green, and blue filters and save the images. Colorkit loads the image files and uses a unique optimizing algorithm to calculate the optimum color image. Supports PW, PIF, PCX, TIFF, GIF, and TGA file formats. ZIP Colorkit, \$75.

ZIP Utility Pack

Translate images between popular file formats: TIFF-PCX-GIF-ASC-WKS-PW-PIF-EPS. Analyze images with your favorite spreadsheet by converting into the worksheet or ASCII formats. Rotate, flip, mirror. ZIP Utility Pack, \$39.

VICTOR LIBRARY a..... \$195
with 8-bit frame grabber \$399
ZIP Colorkit ,..... \$75
ZIP Utility Pack \$39

Call (314) 962-7833 to order

VISA/MC/COD
CATENARY SYSTEMS
470 BELLEVIEW
ST LOUIS MO 63119
(314) 962-7633

8031 Single Board Computers

Experience the power of low cost embedded control in your next project!

The Control-R (pronounced "Controller") series of 8031 Single Board Computers are designed and manufactured to provide the features you need at a price that won't blow your budget. We are **committed** to providing cost effective products to speed construction of your prototype systems and make one of a kind products **trully** affordable. All SBCs operate from 5 volts DC and come assembled and tested with complete documentation and sample programming information.

Control-R I — 11.0592MHz 8031 SBC with 8K EPROM **socket**, optional RS232 serial port and header connection to ports 1 & 3 of the 8031 16 I/O lines make it perfect for small control and data aquisition applications. **\$39.95**

Control-R II — 11.0592MHz 8031 SBC has all the features of the Control-R I plus an 8K SRAM **socket** and expansion bus with address, data and RST, INT1, WR, RD, PSEN, ALE, T1 and power. **\$64.95**

MAX232 — RS232 option for either computer **\$6.95**

SRAM — 8Kx8SRAM (type 6264) for Control-R II **\$10.00**

PseudoSam51 — MSDOS cross- assembler for the 8031 **\$50.00**

\$3.00 Shipping (UPS Ground) for all US orders. Illinois residents must add 6.25% sales tax.

PHONE: (217) 529-7679
Visa MasterCard COD

Cottage Resources Corp.
Suite 3-672
1405 Stevenson Drive
Springfield, Illinois 62703

* MODE 2 is a digital filter (FIR) with the following specifications:

* Fs = 20000 Hz
 * Passband: 0-3000 Hz, 1.28 dB ripple
 * Stopband: 3400-10000 Hz, 50 dB attenuation
 * Filter length: 70 taps
 * Linear phase
 * Delay: 34.5 samples = 1.725 ms

* Start computing the convolution at 20-kHz rate

```

AGAIN1    CALL    WAIT2    Wait 85+2(CALL)
          NOP          Pad for 20 kHz
          NOP
          NOP
          NOP
          IN          XN,PA6    Get new sample
          LAC        XN          Fix for 2's comp
          XOR        BIAS
          SACL       XN
          ZAC
          SACL       X          Ready to start ADC
          OUT        X,PA5      Start ADC
          LT         XNM69      Start convolution
          MPY        HO
          LTD        XNM68
          MPY        H1
          LTD        XNM67
          MPY        H2
          LTD        XNM66
          MPY        H3

          etc...

          LTD        XNM2
          MPY        H2
          LTD        XNM1
          MPY        H1
          LTD        XN
          MPY        HO
          APAC
          ADD        ONE,15    Final addition
          X          To round
          LAC        X          Save (x 0.5)
          XOR        BIAS      Get result
          AND        MASK      Convert 2's -> DAC
          SACL       X          Make LS byte = 0
          OUT        X,PA3     Save
          B          AGAIN1    Out to DAC
                                Back, next sample
  
```

listing 2-The FIR digital low-pass filter assembly language program.

time left over. The limitation of 70 taps was largely due to memory constraints, not time.

The actual implementation of the filter has an additional passband gain of 0.5 (-6 dB) to minimize overflow saturations. With no antialiasing filter, even a square wave could be sampled at the input. After filtering, its fundamental component could exceed the original signal's peak value. The TMS320E15 arithmetic can be made to saturate, instead of overflow, but even then, distortion will result. The gain factor helps minimize distortion.

The technique for designing the filter coefficients is computer aided. The IEEE has available a program which takes a frequency domain

magnitude specification, and produces an optimized set of coefficients for the desired filter length. This method is often called the Parks-McClellan FIR filter design technique [3], and gives a theoretical response which is close to ideal. The coefficients of a linear phase FIR filter are symmetric about their midpoint, so only half the total coefficients need be stored in memory (Figure 4). A look at the program listing shows the FIR filter programmed in-line, rather than using a loop (Listing 2). The powerful indexing features of the 'E15 would have allowed a very tight, three-instruction loop to implement the bulk of the filter calculations, but would have been time inefficient. In-line code is about 100% faster, but uses more EPROM.

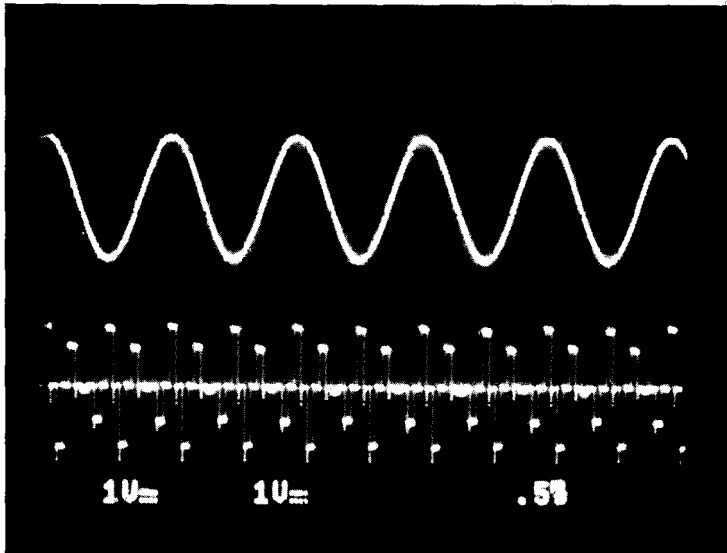


Figure 5—Effect of the speech scrambler on a single 1-kHz sine wave. The original signal is on top while the result of modulating the signal is on the bottom.

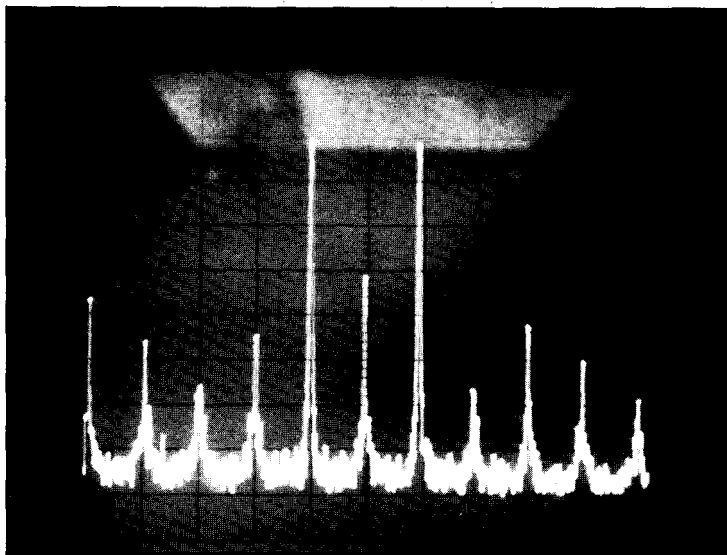


Figure 6—Spectrum of a 1-kHz sine wave after passing through the speech scrambler algorithm. Range is 0 to 10 kHz, with 10 db per vertical division.

The digital filter's perfectly linear phase response yields a delay of 34.5 samples at all frequencies, or a constant 1725 μ s. The zero-order hold effect of the DAC adds an additional half-sample time delay, or 25 μ s. The combination of its very sharp magnitude response and linear phase would be unobtainable in analog circuitry.

4. Modulation (Speech Scrambling)

A primitive speech scrambler can be made by modulating speech audio out to a center frequency of 5 kHz, thus effectively inverting the spectrum. That is, low frequencies will appear high, and vice-versa. To implement this digitally, the algorithm is to multiply by samples of a 5-kHz cosine wave. That is, with a sampling

rate of 20 kHz, the corresponding 5-kHz samples of a cosine are +1, 0, -1, 0. Actually, a sample rate choice of 10 kHz is even more convenient, since the cosine samples of +1 and -1 every period imply a simple alternation of sign applied to the incoming samples. The 20-kHz rate was chosen to remove the replicas from sampling further from the human hearing range (Figures 5 and 6).

The audio is quite effectively scrambled, and the significant lower sideband portion of it only occupies the same 5-kHz bandwidth as the original speech. Thus it could be transmitted over a simple audio channel, such as a telephone system. A similar system on the receiving end would unscramble the sound. This works well

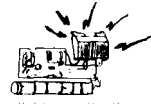
Courteous Service • Discount Prices • Fast Shipping

ALL ELECTRONICS

P.O. Box 567 • Van Nuys, CA 91408

** New ** FLASH UNIT

This NEW compact flash unit comes from a U.S. manufacturer of cameras. It operates on 3 vdc and measures 2 1/2" x 1 1/4". Ideal for use as a strobe, warning light or attention get-
% Complete with instruction on how to wire.
CAT# FSH-1 \$3.75 each 10 for \$35.00



Special New Reduced Price PHOTOFLASH CAPACITOR

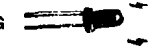
Subicon CE photoflash capacitor. 1.79" dia X 1.1" high. These are new capacitors that have been prepped with 1.4" black and red wire leads soldered to the terminals. 10 Mfd 330 Volt CATW PPC-210 .125 each • 10 for \$11.00 • 100 for \$100.00 Large quantities available. Call for pricing.



FLASHING L.E.D.

Diffused L.E.D. with built in flashing unit. PULSE RATE: 3 Hz @ 5 Volt/20 ma. Unit continually flashes when 5 Volts is applied. Operates between 4.5 Volts and 5.5 Volts. T.1 3/4 size. IDEAL AS AN INDICATOR.

RED CAT# LED-4
GREEN CAT# LED-4G
YELLOW CAT# LED-4Y
\$1.00 each • 10 for \$9.50 • 100 for \$90.00



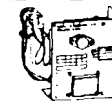
HEAVY-DUTY NICKEL CADMIUM "C" BATTERY

Yuasa 1800C special purchase of new, rechargeable nickel-cad batteries. 1.2 volts, 1800 MAH. PRICE REDUCED ON 10 OR MORE. CAT# HDNCBC 10 pieces for \$42.50 (\$4.25 each) 100 pieces for \$375.00 (\$3.75 each)



POWER SUPPLIES

5Vdc 3 AMP ACDC Electronics #5N3-1 New, prepped power supply with wires and connectors soldered to the inputs and outputs. Open frame style. 4.94" X 4.03" X 2". Input: 115 Vac, UL and CSA listed. Regulated. CAT# PB-53 \$10.00 each



12 Vdc 5 AMP ACDC Electronics #12N5 or equiv. Input: 100-240 Vac (wired for 115 Vac) Output: 12 Vdc @ 5 amps. Open frame style 7" X 4 3/4" X 3" high. Regulated. CATX PS125 \$37.50 each

24 Vdc 2.4 AMP Power-One Inc. #HC-24-2.4 Input: 115/230 Vac (wired for 115 Vac) Output: 24 Vdc @ 2.4 amps. Open frame style. 5.62 X 4.87" X 2.50" CSA listed. CAT# PS-2424 \$30.00 each

TOLL FREE ORDER LINES

1-800-826-5432

CHARGE ORDERS to Visa, MasterCard or Discover

TERMS: Minimum order \$1 0.00 Shipping and handling for the 48 continental U.S.A. \$3.50 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include state sales tax (6%, 6 1/2%, 7%) Quantities Limited. NO C. o. D. Prices subject to change without notice. Call Toll Free, or clip this coupon

FREE 60 Page Catalog

Containing over 4,000 ITEMS

ALL ELECTRONICS CORP.

P.O. Box 567 • Van Nuys, CA • 91408

Name

Address

City

State

Zip

Order Service # 103

February/March 199 J 43

au
hu
pa
inj
sp
fec
pu
of
of
ref
kt

I
U
D
W
D
D

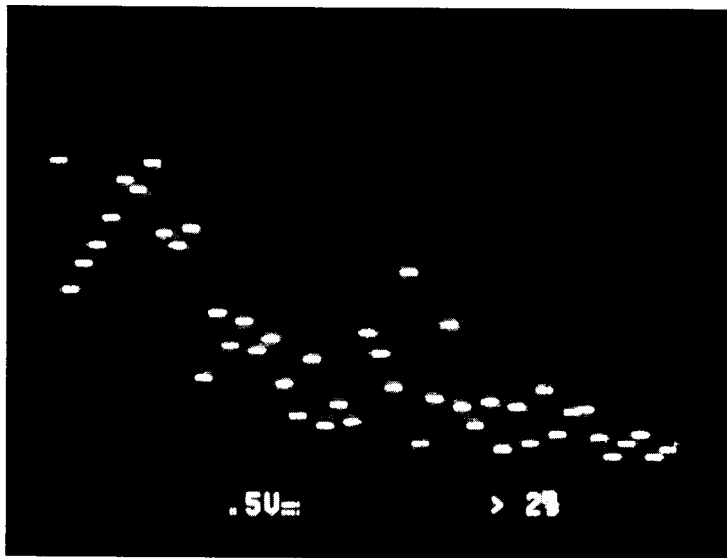
Figur
digit

for privacy purposes, but is by no means secure.

Nevertheless, it serves to demonstrate modulation, which is a useful technique in modem design, radio signal generation, and frequency-division multiplexing applications.

5. Spectrum Analysis

The final major application programmed turns an oscilloscope into a spectrum analyzer. The discrete Fourier transform (DFT) is the heart of the method. It takes 100 samples at a 10-kHz rate and performs a DFT, computing magnitudes of the 50 harmonics representing the components



up to 5 kHz (at 100-Hz intervals). These magnitudes are output through the DAC, which drives a scope. A leading sync pulse will trigger the scope, then displaying the spectrum of the input signal. Computations are fast enough that the display is refreshed about 25 times per second, causing minimal flicker.

A music or speech spectrum can be observed as it changes in real time (Figure 7), a mesmerizing experience, and experiments with a signal generator verify the harmonic makeup of a square wave and triangle wave (including the associated aliasing). A sinusoid above 5 kHz will appear in the spectral display below 5 kHz, a perfect example of aliasing.

Since music has relatively little power at any one frequency, a "MUSIC" setting provides a gain boost

of four, internal to the software, creating a more useful display. A "PEAK" setting causes peak values at each harmonic to be held. This is reset by resetting the processor.

A look-up table provides the sines, cosines, and square roots necessary for the algorithm.

DIGITIZED SPEECH

After all of the above software was written in assembly language, only 1K of the 4K-word (16-bit) EPROM was used up. As long as I had 3K words left, I filled it up with some digitized speech. Upon changing

modes, you now hear my wife's voice saying "thank you," in a 0.6-second episode of sampled speech. The sampling rate for this was around 10 kHz, and two 8-bit samples were packed into each 16-bit word. This feature is switch selectable with the "TALK" switch. Finally the EPROM was full.

USER ALGORITHMS

The above techniques were all programmed into the 'E15 to allow the user to get familiar with DSP techniques without the difficulties of fighting hardware and software problems in addition to the mathematics itself. They provide a great variety and flexibility with which to learn. However, with the on-board EPROM, the user can experiment with other algorithms, as well. With the proper assembler

and EPROM adapter, the 4K words of EPROM can be the host to any type of DSP technique. One thing to keep in mind here, however, is that certain algorithms can generate high-frequency components which can alias. For example, the modulation example above had a carefully chosen carrier frequency so modulation products were still, for the most part, below half the sampling frequency. An operation such as full-wave rectification could generate aliased high frequencies. That is, not all analog techniques have a painless transition to the DSP world. Much care must be taken in these cases. While the difficulties of the hardware and language were surmounted by the ready-made Mini-DSP, so were the subtleties of the mathematics.

WHERE TO FROM HERE?

Using a device like the Mini-DSP should be the springboard to using DSP methods in actual engineering applications. While audio processing is, perhaps, the most fun, applications in data analysis, monitoring, control, and adaptive techniques abound. Give it a try! ❖

REFERENCES

1. D. McConnell, 'Digital Signal Processing-An Introduction, Part 1,' Circuit Cellar INK, issue 13, Feb/Mar 1990, 30-41.
2. D. McConnell, 'Digital Signal Processing-DSP Applications with the TMS320C25, Part 2,' Circuit Cellar INK, Issue 14, Apr/May 1990, 48-56.
3. J. McClellan, T. Parks, L. Rabiner, 'A Computer Program for Designing Optimum FIR Linear Phase Digital Filters,' IEEE Transactions on Audio and Electroacoustics, vol. AU-21, no. 6, Dec. 1973, 506-526.

Steven Reyer holds a Ph.D. in the area of digital signal processing. In addition to being a professor of Electrical Engineering at the Milwaukee School of Engineering, he does consulting in the fields of DSP and communications systems.

IRS

- 4 10 Very Useful
- 4 11 Moderately Useful
- 4 12 Not Useful

FEATURE ARTICLE

Mark E. Nurczyk

Analog Circuit Design

Stripping Away the Mystery for Digital Designers

One of the most difficult aspects of any microcomputer-based project is the design of the analog signal conditioning circuitry. The traditional method is to hand calculate initial values, build a prototype, and spend lots of time and money putting it through its paces to determine if it works as desired.

Now you have one circuit that works, but will the thousandth? When you build many circuits from real-life components, what will their performance be across the complete tolerance range of all components? To answer these questions you have two options: build one thousand or so circuits, then redesign from what testing these units reveals to you; or simulate the circuit action using a computer program.

To illustrate the proper steps in analog design, the Student Version of PSpice will be used. PSpice, an electronic circuit simulation program, allows you to determine circuit performance before picking up wires, resistors, or soldering iron. PSpice will not design your circuits for you; only you have the intellect for that. It will show you how your circuit will theoretically behave. It will allow you to use your computer as an oscilloscope. PSpice will allow you to see the voltage waveform at each circuit node (a point where two or more components come together) or the current through each component. It will save you from difficult and tedious hand calculation, allowing you to be more creative.

The steps necessary for proper design of analog circuits are many and varied. The procedures required are: define the input protection network to protect from real-world tran-

```
110 LOGA = 1.24798
120 C = 2.38559
130 D = 3195.89
140 FOR TZONE = 33 TO 122
150   KTZONE = (TZONE - 32) * 5 / 9 + 273.15
160   CLOGT = C * LOG(KTZONE)/LOG(10)
170   RTZONE = 10^(LOGA - CLOGT + D/(2.3026 * KTZONE))
180   PRINT USING "   ####";TZONE;
190   PRINT USING "   #####";10000*RTZONE
200 NEXT TZONE
```

listing 1 -A small BASIC program is used to develop a PSpice thermistor model.

```
*
* THERMISTOR SWEEP
*   +--+Control voltage
*   || +-Reference resistor
*   ||| t--Simulated component
*
.SUBCKT ZX 1 2 3 4 5
EOUT 4 6 POLY(2) (1,2) (3,0) 0 0 0 0 1
FCOPY 0 3 VSENSE 1
RIN 1 2 1G
VSENSE 6 5 0
.ENDS
RM 11 0 1
x3 20 0 11 1 0 ZX
VTH 20 0 PWL
+ 1 84845 2 82173 3 79592 4 77100 5 74694
t 6 12370 7 70125 8 67956 9 65862 10 63837
t 11 61881 12 59990 13 58163 14 56397 15 54690
t 16 53039 17 51442 18 49899 19 48406 20 46963
t 21 45566 22 44215 23 42908 24 41643 25 40419
+ 26 39235 27 38088 28 36979 29 35904 30 34964
+ 31 33858 32 32883 33 31958 34 31060 35 30191
t 36 29349 37 28534 38 27744 39 26979 40 26237
t 41 25519 42 24823 43 24148 44 23494 45 22859
t 46 22245 47 21648 48 21070 49 20510 50 19966
t 51 19438 52 18926 53 18430 54 17948 55 17480
t 56 17027 57 16586 58 16158 59 15743 60 15340
t 61 14949 62 14569 63 14199 64 13841 65 13493
t 66 13154 67 12825 68 12506 69 12195 70 11894
t 71 11601 72 11315 73 11038 74 10769 75 10507
t 76 10252 77 10004 78 9764 79 9529 80 9301
t 81 9079 82 8864 83 8654 84 8450 85 8251
t 86 8057 87 7869 88 7686 89 7507 90 7333
t 91 7164 92 6999 93 6839 94 6683 95 6531
t 96 6382 97 6238 98 6097 99 5960 100 5827
t 101 5696 102 5570 103 5446 104 5325 105 5208
t 106 5093 107 4982 108 4813 109 4161 110 4663
t 111 4562 112 4464 113 4368 114 4274 115 4183
+ 116 4094 117 4007 118 3922 119 3839 120 3759
+ 121 3680 122 3603
.TRAN 1 122
.PRINTTRAN V((7))

.lb:64
```

listing 2 -A piecewise linear voltage source is used to synthesize a thermistor's resistance change with temperature.

sients; choose the proper circuit topology for accurate circuit operation; and check the circuit sensitivities to determine component tolerances. Total circuit performance will be examined with all components taken through their full tolerance range. Finally, the circuit's stability will be checked to be sure that it does not oscillate.

The circuits that I will develop throughout this article were devised to illustrate the design process. They will work as shown, but may not be optimal for your purposes. When you get your copy of the Student Version of PSpice, you can experiment with different values for each of the components or the circuit topology. You are free to **exercise your wildest flights of fantasy**; you cannot destroy expensive and difficult to obtain components. Then you will come up with a circuit that is just perfect for you.

Grab some Jolt Cola, a couple of cases of Twinkies, and a pizza or two (no use not eating right), and let's begin.

FIRST, YOU MAKE A LIST...

To begin a circuit design you must have a place to start. If you don't have a good foundation, you can't build a proper structure. A specification that defines how the circuit is supposed to work must be developed and written down. If you don't write it down, you may forget what you're trying to do. Halfway through, you might end up with a circuit that doesn't perform the task required. A specification for a temperature sensing circuit is below:

- A. Accuracy
 - 1. 0-60°F: ±5°F
 - 2. 55-85°F: ±2.5°F
 - 3. 80-115°F: ±5°F
- B. Resolution
 - 1. 10-60°F: 2 bits/°F
 - 2. 55-85°F: 4 bits/°F
 - 3. 80-115°F: 2 bits/°F
- C. Electromagnetic Interference
 - 1. Must withstand 15,000-V static discharge
 - 2. Must withstand 600-V 1.2 x 50-μs pulse discharge
- D. Must be able to find open or shorted thermistor
- E. Ambient temperature 0-140°F

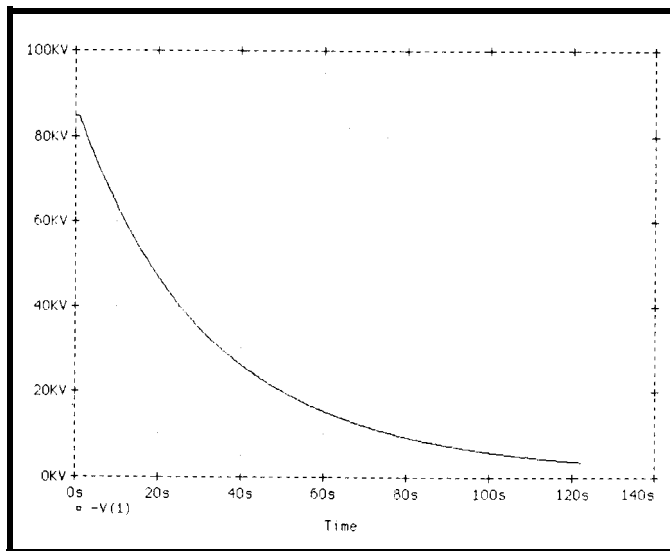


Figure 1—The PSpice output of the thermistor mode/ shows the characteristic exponential response of a real thermistor.

THE RIGHT SENSOR

Now that we have defined the requirements, let's pick a sensor. There are many devices that can be used to determine temperature. There are bandgap semiconductors, thermocouples, RTDs, or thermistors. For this design exercise we will use a thermistor. Why? They are easy to use, they are easy to purchase, they have a high resistance change versus temperature change, and it will make for an interesting article that will teach you more.

The thermistor that I'll use is a Keystone D97 rated 10,000 ohms at 25°C. In the databook, Keystone supplies a formula that defines the thermistor resistance versus temperature. To develop a thermistor model, I wrote a small BASIC program, a portion of which is shown in Listing 1.

The values in lines 110-130 are constants supplied by Keystone for the temperature range 33-122°F. Line 150 changes the temperature from Fahrenheit to Kelvin. Lines 160 and 170 solve the Keystone equation.

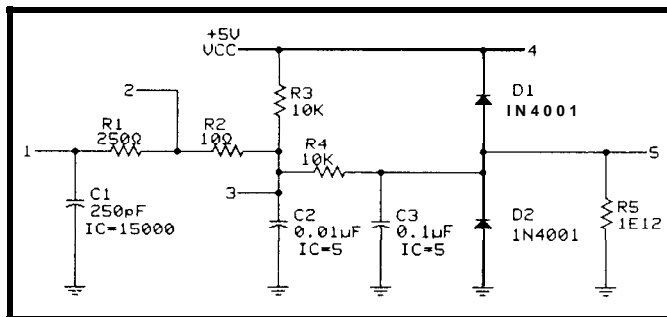


Figure 2—Static discharge protection is added to the input of the temperature measurement system.

```

Circuit Cellar Ink input circuit *** Static discharge ***
C1 1 0 250P IC=15000
C2 3 0 .01U IC=5
C3 5 0 .1U IC=5
R1 1 2 250
R2 2 3 10
R3 3 4 10K
R4 5 5 10K
R5 0 1T
D1 5 4 D1N4001
D2 0 5 D1N4001
vcc 4 0 DC 5
.TRAN 2.5U 500U UIC
.PROBE
.MODEL D1N4001 D(RS=.04 CJO=55PF IS=1.383-09 N=1.7 VJ=.34V TT=5U
M=.38 BV=75V)
.END
  
```

listing 3—The corresponding PSpice mode/ for the static discharge protection circuit.

There are two ways to use the results of the BASIC program to build a thermistor model suitable for PSpice. One is a DC model, the other is a voltage source model used in the transient analysis mode of PSpice. The DC model is preferred. DC analysis is the fastest analysis mode in PSpice. [Editor's Note: All software for this article, excluding PSpice and MiniTK, is available on Software On Disk #19 or on the Circuit Cellar BBS. See page 107 for ordering and downloading information.] PSpice is very computationally intensive and uses lots of computer time. There is a great incentive to use the fastest running analysis. However, the DC model is very difficult to synthesize. It involves using a diode's junction voltage change with forward current to create an exponential response. Then you have to multiply by an appropriate constant and add an offset. It can take many weeks of trial and error to properly determine the values. Alternatively, you can use a PieceWise-Linear (PWL) voltage source in a transient analysis to model the thermistor resistance.

Listing 2 shows how a PWL voltage source may be used to synthesize a thermistor's resistance change with temperature. Each pair of time-voltage values specifies a point on a curve. The magnitude of the voltage between each time-voltage pair is the linear interpolation of the values defined by each pair. For this thermistor model, the time in seconds corresponds to the temperature in degrees Fahrenheit. The voltage at each pair corresponds to the resistance of the thermistor at each temperature. By using seconds as the time change, the resistance changes so slowly that there will be no slew rate limiting due to the finite frequency response of the op-amps.

In Listing 2, subcircuit ZX comes from Paul W. Tuinenga's excellent book "SPICE: A Guide to Circuit Simulation and Analysis Using PSpice." The subcircuit takes a voltage in its control voltage pins, multiplies it by the value of the device connected to the reference pin, and presents a floating impedance on its simulated component pins. The simulated impedance is the product of the voltage and the reference device. Figure 1 displays the output of the thermistor model. It shows the characteristic exponential response of a thermistor. The Y-axis is resistance and the X-axis is temperature in degrees Fahrenheit.

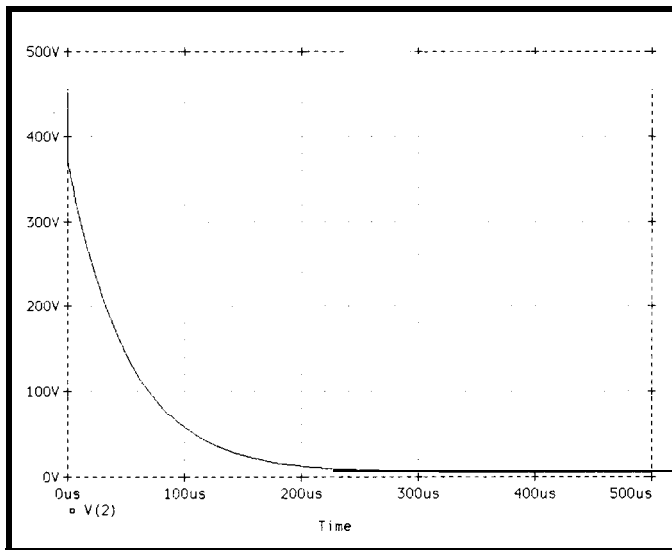


Figure 3-A graph at the voltage at node 2 in Figure 2 shows a fast, high-voltage spike on the input.

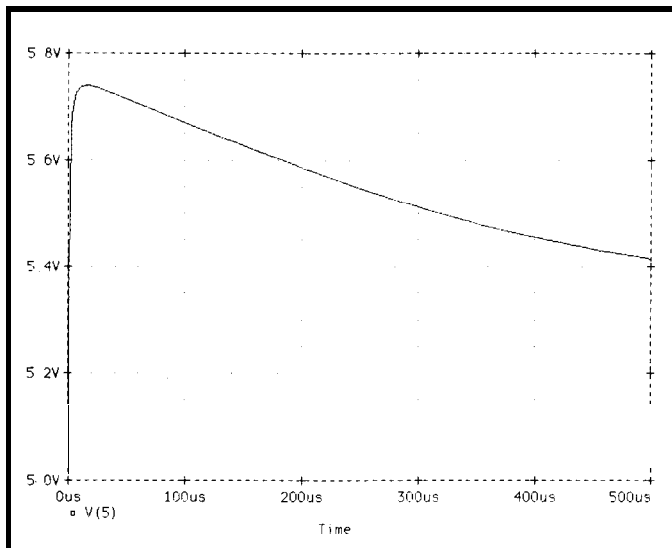


Figure 4-Moving to the output of the protection circuit shows the voltage reaching the input op-amp is within tolerances.

INPUT PROTECTION NETWORK

Many phenomena are present in the real world that will disrupt or destroy an input circuit. Among these are lightning discharges, static electric discharges, or miswiring. Circuits can be designed not to be affected by this interference. The schematics in Figures 2 and 5 and PSpice Listings 3 and 4 show a protection network with standard electrical disturbance circuits. These schematics and listings can be used to develop circuits that reject these transients.

Figure 2 is used for static discharge protection design. The PSpice input file is shown in Listing 3. C1 and R1 define a simple model of a human being charged up to 15,000 volts. Those of you who go through cold, dry winters know the joy of touching your home thermostat after walking across a carpet. The input circuit, which will become part of our circuit, consists of all the rest of the parts. R2 provides a small, finite impedance that provides the first stage of filtering with C2. R3 is one fourth of a bridge circuit (to be described later) that will work with a thermistor connected from node 2 to ground. R4 limits the input current and provides additional filtering in conjunction with C3. R5 simulates the input resistance of an LMC660 op-amp, which will be used in this design. Diodes D1 and D2 limit the voltage at node 5 to no more than one diode drop above the power supply or below ground.

Figure 3 shows the voltage waveform at node 2. The voltage starts at about 450 volts and decays to 5 volts in about 300 microseconds. A gaze at the LMC660 specification sheet indicates that the maximum input voltage is 0.7 volts above the power supply voltage.

Figure 4 shows the voltage waveform at node 5. The voltage starts at about 5.7 volts and decays to about 5.4 volts in about 300 microseconds. This shows that the protection circuit is effective in limiting the voltage at the input of the op-amp to within tolerances.

Since the LMC660 is restricted to 16 volts DC, clearly 450 volts is above the maximum. We will use power supplies of 5 volts DC. The rest of the circuitry has to limit the voltage at node 5 to no greater than 5.7 volts.

The essential protection devices are the capacitors. Once a capacitor attains a specific voltage it contains an exact electric charge. From elementary physics, the charge q of a capacitor is related to its voltage by

$$q = c v$$

In our closed system of Figure 2, the total charge on the capacitors does not change, so the total charge is

$$(C1 \times 15000) + (C2 \times 5) + (C3 \times 5) = 4.3 \times 10^{-6} \text{ coulomb}$$

When the capacitors charge equalizes the resultant voltage is

$$V = 4.3 \times 10^{-6} / (C1 + C2 + C3) = 39 \text{ volts}$$

The actual resultant voltage is lower, modified by the discharge paths through R3 and R5. It is, however, greater than that allowed on the inputs of the op-amps that will be used. Diodes D1 and D2 were added to shunt the overvoltage to power and ground, respectively, with the current through them limited by R4. Figure 4 shows the voltage at the op-amp input with the diodes in place. As you can see, the op-amp will be saved. A similar result could have been obtained if C3 were larger. A value of around 6 μ F will keep the voltage at node 5 to about 5.6 volts. I'll stick with the diodes; large electrolytic capacitors have less than ideal transient characteristics.

Figure 5 shows the input circuit modified to design against a 600-V 1.2 x 50- μ s pulse discharge. The PSpice input file is shown in Listing 4.

Where do these 600-V 1.2 x 50- μ s pulsedischarge transients come from? They are induced into the input wires when the input wires are bundled together with wires carrying large inductive loads. For example, if you run the thermistor wires from the circuit board to the thermistor in the same

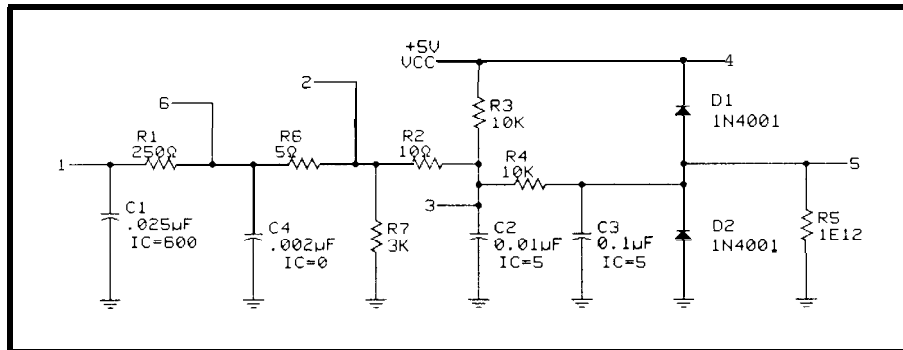


Figure 5—The protection circuit is modified to protect against a 600-V 1.2 x 50- μ s pulse.

```

Circuit      Cellar  Ink  input  circuit  ***  Conducted  susceptibility  *y
C1          1      0      .025U  IC=600
C2          3      0      .01U   IC=5
C3          5      0      .1U    IC=5
C4          6      0      .002U  IC=0
R1          1      6      250
R2          2      3      10
R3          3      4      10K
R4          3      5      10K
R5          5      0      1T
R6          6      2      5
R7          2      0      3K
D1          5      4      D1N4001
D2          0      5      D1N4001
vcc         4      0      DC 5
.TRAN       2.5U  500U  UIC
.PROBE
.MODEL     D1N4001 D(RS=.04 CJO=55PF IS=1.38E-09 N=1.7 VJ=.34V TT=5U
M=.38 BV=75V)
.END
  
```

listing 4—The corresponding PSpice model is also modified for the extra protection.

HURDLE

8051 64180 68HC11 6x01/3
8096 3770 80X86 H85XX
KX86 683XX

MICROCONTROLLER

BOUNDARIES

with **Byte-BOS™** real-time

Multitasking Operating System

Byte-BOS Real-Time Multitasking Operating System (BOS) is a powerful multitasking operating system designed specifically for embedded microcontroller applications.

BOS is written in C with an assembly language kernel tuned to a specific microcontroller. Application code written in C using BOS on one microcontroller can be used on any microcontroller supported by BOS.

BOS reduces integration time by supporting "on board" peripherals and popular C compilers. BOS includes timer support, an asynchronous communications package, system "make" files, and working application code.

BOS supports a wide variety of microcontrollers including the 8051, 8096, 80188/86, 6801/3, 68HC11, 68332, 68302, 68340, 6301/3, 64180, H8500, and 37700 families.

BOS, now available for the PC, supports PC peripherals and DOS/BIOS calls. The BOS PC system can be used in a desktop/embedded application or for cross development and debug of software for all microcontrollers using BOS.

BOS is available as "no royalty" source code. BOS supports one timer and serial port, and is configured to the C compiler of choice. The complete system sells for \$199 and includes a user manual, "make" files, and application code.

Byte-BOS™ Integrated Systems 415-543-3626

wire bundle that carries power wiring to a motor, whenever the motor turns on and off it may induce pulse transients of this type onto the thermistor leads.

Another source is near-proximity lightning strikes. On lines inside a building the pulses rarely exceed 600 volts. On lines that run outside a building, use the FCC specification for telephone lines: 2,000 volts. Yes 2,000. These are the transients your mother warned you about!

Figure 6 shows the input voltage wave form at node 2 due to a positive 600-volt pulse. As you can see, the voltage raises to 350 volts in about 1 microsecond and decays to 200 volts in 50 microseconds. Plainly, 350 volts is also above the LMC660 input specification. The rest of the circuitry has to also limit this pulse to no more than 5.7 volts.

Figure 7 displays the voltage at the input to the op-amp with the diodes in place. As for the static discharge case, the op-amp input voltage is properly reduced.

How else can we protect the circuit? C3 could be made larger and the diodes deleted. At 600 volts, 4 μF will keep the voltage at node 5 to 5.8 volts. If the input transient were increased to 2,000 volts, 14 μF would be needed to keep node 5 to 5.8 volts. But 14 μF and 10,000 ohms would limit the input frequency to about 1 Hz. For a temperature input this may be acceptable frequency performance; just make sure to evaluate all options fully. And remember, large electrolytic capacitors have less than ideal transient characteristics.

Of course, the real-world isn't all positive voltages. These same protection networks have been tested with negative transients and been found to be equally effective.

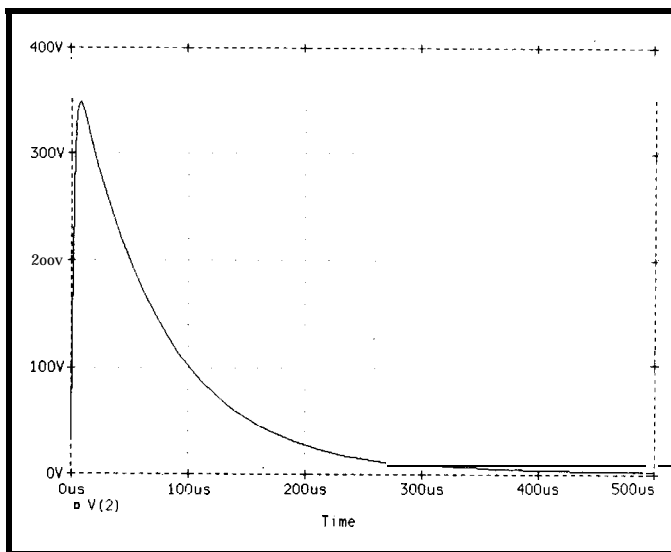


Figure 6—The voltage at node 2 due to a positive 600-V pulse raises to 350 volts in about 1 microsecond.

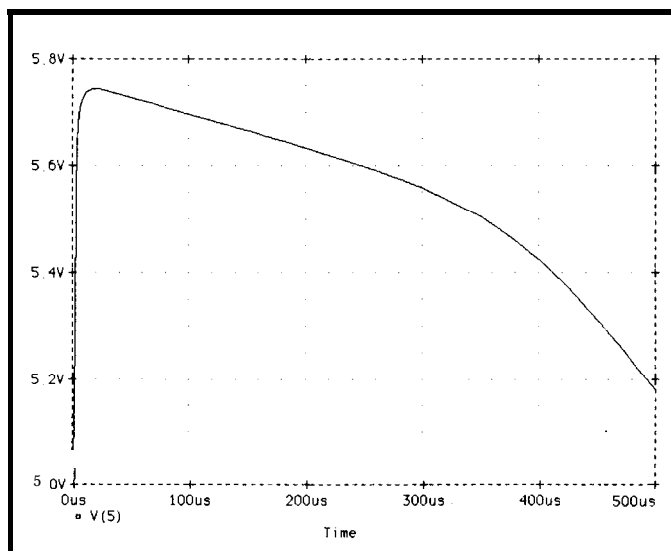


Figure 7—The voltage at the input to the op-amp is like the static discharge case, within circuit tolerances.

SENSOR BRIDGE

Linear sensors are the easiest to apply. But, as depicted in Figure 1, a thermistor is anything but linear. To help linearize the thermistor output signal, a full-bridge circuit will be used. Stripped to its essentials, the bridge circuit is shown in Figure 8. R1, R2, and R3 are fixed resistors that we will choose. RTH is the thermistor. The output of the bridge will be VFIX-VIN. The bridge output voltage will be multiplied by an appropriate gain, using an op-amp in a differential amplifier topology, to scale the op-amp output voltage into the full range of zero to five volts. This will give the

greatest resolution using an eight-bit A/D converter referenced to five volts.

For the greatest linearity, R2 and R3 should equal the thermistor's resistance at the midpoint temperature. R1 should equal the thermistor resistance at the temperature where we want VFIX-VIN to equal zero.

For the low-resolution temperature range of 0–115°F, the midpoint temperature is $(115 - 10) / 2 + 10$, or 62.5°F. For the high-resolution temperature range of 55–85°F, the midpoint temperature is $((85 - 55) / 2) + 55$, or 70°F. We only want to use one thermistor. We don't want to have the trouble of correlating the accuracy of two separate thermistors, or the cost of manufacturing two individual input circuits. This creates a problem: which midpoint do we use?

The most critical application is the high-resolution temperature range. To optimize the circuit's performance we shall pick the midpoint temperature to be 70°F. Therefore, R2 and R3 should be 11894 ohms. The closest 1% resistor value is 11.8k. R1 will be different for each resolution range.

To pick the proper values for R1, the following MiniTK formulas are used:

$$\begin{aligned} \text{VIN} &= 5 \times (\text{RTH} / (\text{RTH} + \text{R2})) \\ \text{VFIX} &= 5 \times (\text{R1} / (\text{R1} + \text{R3})) \\ \text{GAIN} &= 5 / (\text{VFIX} - \text{VIN}) \\ \text{VOUT} &= (\text{VFIX} - \text{VIN}) \times \text{GAIN} \end{aligned}$$

Where to start? We already know that R2 and R3 should be 11.8k. Next we have to find R1. To make the math easier to understand, we will pick a positive slope with increasing temperature. This means that VFIX-VIN will be zero at a low temperature.

First, look at the low-resolution temperature range. Taking into account the tolerances of the parts, pick

SPICE

SPICE (Simulation Program with Integrated Circuit Emphasis) was developed at the University of California, Berkeley, to be a general-purpose circuit simulation program for nonlinear DC, nonlinear transient (voltage vs. time), and linear AC analysis. Circuits simulated may contain resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, four types of dependent sources, transmission lines, and the four most common semiconductor devices: diodes, BJTs, JFETs, and MOSFETs.

Features include:

- *Simulating electronic circuits
- *Flexibility to change designs and test new ideas
- *Checks and reports the tolerance of component parts
- May check circuit ideas before building a breadboard
- Can try out ideal operation by using ideal components to isolate limiting effects in a design
- Can simulate test measurements that are
 - *Difficult (due to electrical interference)
 - *Inconvenient (special test equipment is unavailable)
 - *Unwise (the test circuit would destroy itself)

We will be using the student edition of PSpice. The student edition is limited to ten transistors and about 20 circuit nodes. A package containing the student edition of PSpice, the current PSpice manual, and the book 'A Guide to Circuit Simulation and Analysis Using PSpice' is available for \$70 from

MicroSim Corp.
20 Fairbanks
Irvine, CA 92718

The program is also available on the Circuit Cellar BBS and many other bulletin board systems.

MiniTK

MiniTK is a subset of the popular TK Solver Plus analysis program. TK accepts equations or mathematical models in their natural form and generates their solutions regardless of the selection of known variables, the order of equations, or the location of unknowns on the left or right side of the equal sign. MiniTK is available for \$20 from

Universal Technical Systems, Inc.
1220 Rock St.
Rockford, IL 61101

(Editor's Note: Please note that these programs are not included on the Software On Disk for this issue.)

the lowest temperature to be -5°F instead of 10°F . The value of the thermistor at -5°F is 103,030 ohms; the closest 1% value is 102k. Under these conditions, V_{OUT} is -0.0067 volts; reasonably close to zero. Next, let's pick a high temperature about the same number of degrees above 115°F that -5°F is below 10°F . For fun, I picked 131°F . In MiniTK, make R_{TH} equal to the thermistor resistance at 131°F , or 29870 ohms. The gain required for the differential amplifier to produce a five-volt output is 1.43836. Using a BASIC program that I wrote (GAINRAT.BAS), the two closest 1% resistors for that ratio are 1.30 and 1.87 ohms. This gives a gain ratio of 1.43846, or the correct ratio within 0.007%. The output voltage at 131°F will be 4.9937 volts—very close to five volts.

Similarly, the resistor values for the high-resolution range can be found. We'll concentrate on the boundary temperatures of 45°F and 95°F . At 45°F , the thermistor resistance is 22,859 ohms. We will pick R_1 to be 22.6k, the closest 1% value. At 95°F , where R_{TH} is 6531, the required gain is 3.2975. Using GAINRAT.BAS, the gain resistors for the high-resolution circuit will be 1.37 and 4.53 ohms. These values give a gain of 3.3065, an error of 0.275%.

OP-AMP CIRCUIT

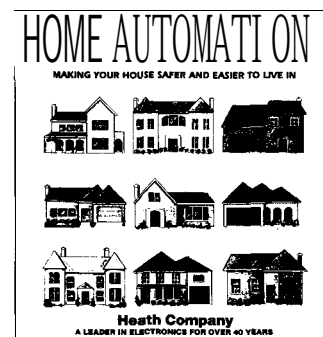
The circuit is shown in Figure 9. The bridge consists of resistors R_2 , R_{TH} , R_6 , and R_7 . X_1 , an LMC660 op-amp, matches the impedance of the input stage to the differential amplifier X_2 . The resistors R_6 and R_7 are

Safety, Security, Convenience, Entertainment and Energy Management

Home Automation from Heath, the catalog that has it all...

Enter the world of Home Automation. Remote lighting and appliance controls. Security alarms and lighting. Automated thermostats. Video monitoring systems. Whole-house security systems. They're all yours in the Heath Home Automation Catalog. To receive your FREE copy, call today toll-free.

1-800-44-HEATH
(1-800-444-3284)



Heath Company
Marketing Dept. 026-024
Benton Harbor, MI 49022

CL-807

Reader Service #146

one-tenth of the predicted values. This is allowable since the ratio of the resistors is the critical factor and not their absolute values. The lower values of R6 and R7 allow R4, R5, R8, and R9 to have reasonable impedance values (<500k) without unduly loading the R6/R7 network. In the final design, the amplifier X2 and its associated resistors R4–R9 will be duplicated using the values in parentheses. Both differential amplifiers are fed from the output of X1 at node 5. The PSpice input file is presented in Listing 5.

If we concatenate Listing 2 and Listing 5 it will be possible to determine the circuit's output as the temperature is swept from 0 to 120°F. If a .PRINT v(7) is inserted into the input file, then, after PSpice is run, the swept parameter as well as the associated output voltage will be listed in the PSpice output file. Listing 2 sweeps the thermistor model through temperature. The output file, in this case, will contain the temperature in degrees Fahrenheit and the value of the voltage at node 7.

After removing the nonessential information from the output file with a word processor, the information contained in the output file can be read into a spreadsheet for further processing. For example, the output voltage can be divided by the voltage per bit ($5\text{ V} / 256 = 0.0195\text{ V}$) of an 8-bit analog-to-digital converter to determine the bit value generated at each temperature. Using the resistor values for the low-resolution circuit, Figure 10 is the result. Figure 11 shows the results for the high-resolution circuit.

The output looks a lot more linear than the thermistor function shown in Figure 1. But how shall we have the microcomputer interpret the output? Two ways spring to mind: table look-up and line-segment equations.

For a table look-up, the A/D bit value is used as an address index. Stored in the table are the scaled engineering unit values that correspond to the A/D bit values. Table look-ups are fast, but a table uses a lot of memory. Tables of these types can use from 256 to 768 bytes of memory per variable depending on the precision required. Memory use of this magnitude may

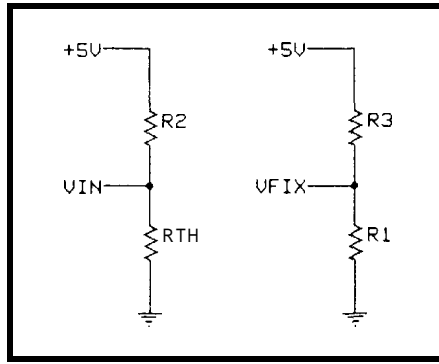


Figure 8-A full-bridge circuit is used to help linearize the thermistor output signal.

not be critical in desktop PC programming, but can be crucial in masked microcomputers with limited resources. Many masked microcomputers only have 4,096 bytes of program memory. Two or three variable tables can use up to one-half of the available memory leaving little space for the application program.

Line segment equations solve an equation of the form $Y = mX + B$. The input is the A/D bit value and the output is the variable in scaled engineering units. The slopes and intercepts for many line segments that approximate the input function are determined. All that is stored are the slopes, intercepts, and the bit values over which each set of values is valid. This method is computationally intensive and fairly slow. However, it only uses a few bytes per segment.

The data that was placed into the spreadsheet for Figures 10 and 11 can be used to determine the slopes, intercepts, and endpoints needed for these line segments. Most spreadsheets have a linear regression facility in their advanced feature section that will

determine the slope, intercept, and percentage of fit to a straight line. The approximate endpoints for the line segments can be determined by observation. Look again at Figure 10, the output of the low-resolution circuit. The curve is slightly "S" shaped. The points at which the curve changes direction are the endpoints of each line segment. My eye places these reversals at 20 and 100 degrees.

My spreadsheet gives the results shown in Table 1. These line-segment equations are only approximations of the curve of Figure 10. Next, we have to determine the errors in perceived temperature to see how close to the truth we are. In our spreadsheet we can define cells to solve the $Y = mX + B$ equations using the slopes and intercepts in Table 1 and the bit values from Figure 10. The answers will be what the microcomputer perceives as the temperature for each bit value. What would be more interesting for us would be to subtract the computed temperature values at each bit value from the actual temperature at each bit value. Figure 12 shows the results of these machinations.

What does Figure 12 tell us? If every part of our circuit has perfect tolerance then the microcomputer will perceive the temperature as 1.4° higher than actual at 0°. It will believe that the temperature is -0.6° lower than actual at 40°, and 0.25° higher at 110°.

We can improve on this by increasing the number of line segments. Pick the new endpoints where the slope reversals exist in Figure 12. Table 2 can be prepared using the regression analysis of the spreadsheet.

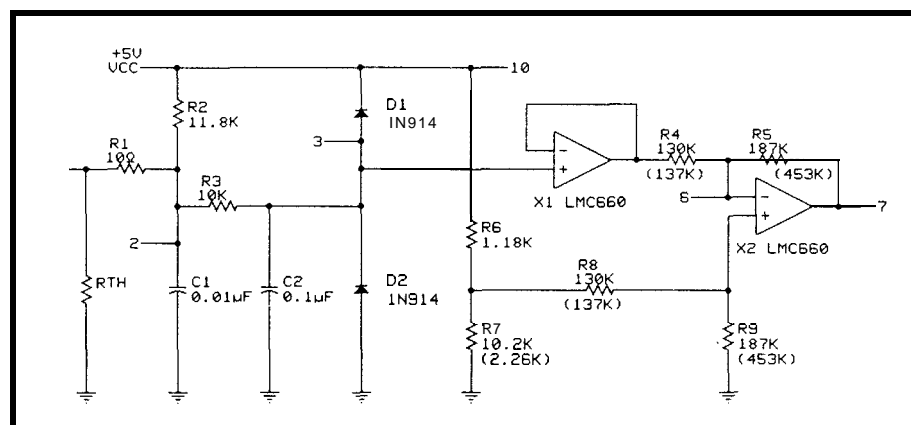


Figure 9—The high- and low-resolution circuits differ only by a few component values.

```

Thermistor full bridge circuit
*
* VOLTAGE SOURCES
*
V1 10 0 DC 5
*
* RESISTORS
*
R1 1 2 RTHERM 10
R2 10 RONE 11.8K
R3 2 3 RTHERM 10K
R4 5 6 RONE 130K
R5 6 7 RONE 187K
R6 18 8 RONE 1.18K
R7 8 0 RONE 10.2K
R8 9 RONE 130K
R9 9 0 RONE 187K
RTH 1 0 RTHERM 3759
*
* CAPACITORS
*
C1 2 0 .01U
C2 3 0 .1U
*
* OP AMPS
*
X1 3 5 5 LMC660
X2 9 6 7 LMC660
* OPAMP MACROMODEL SUBCIRCUIT
SUBCKT LMC660 12 3
*
* | | |
* I I +-OUTPUT
* | t----INVERTING INPUT
* +-----NONINVERTING INPUT
* ; INPUT IMPEDANCE
RIN 1 2 1T
* GAIN AND PHASE CONTROL
GM 0 4 1 2 1 ; GAIN=100K, OPEN LOOP CUTOFF FREQ=14 HZ
RG 4 0 100000
CP 4 0 .11368U
* OUTPUT SECTION
EOUT 5 0 4 0 1 ; OUTPUT RESISTANCE = 50.9 OHMS
ROUT 5 3 50.9
.ENDS
*
* DIODES
*
D1 3 10 D1N4001
D2 0 3 D1N4001
.MODEL D1N4001 D(RS=.04 CJO=55PF IS=1.38E-09 N=1.7 VJ=.34V TT=5U
M=.38 BV=75V)
*
* MONTE CARLO SECTION
*
.MC 200 DC V(7) YMAX
.MODEL RONE RES(R=1 DEV=1%)
.MODEL RTHERM RES(R=1 DEV=5%)
V100 1000 0 DC 1
R100 1000 0 1
.PRINT DC V(7)
.DC V100 0 1 1
.END

```

Listing 5—The PSpice model for the op-amp circuit is a bit more complicated than for the protection circuits.

Using the same techniques, Figure 13 was prepared for the high-resolution circuit. The results are shown in Table 3.

To achieve proper circuit performance, the correct tolerance for the bridge and gain setting resistors must be determined. PSpice can calculate the sensitivity, or change in value, of any node voltage due to component value variations. A sensitivity analysis calculates the component sensitivities at the DC bias point; it will not

work for components that determine frequency response of a circuit.

The output node of Figure 9 is node 7. To determine the sensitivity of node 7, the statement `SENS V (7)` is placed into Listing 5. The output file will contain the results shown in Table 4.

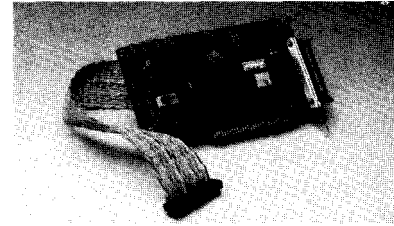
WHAT DOES IT MEAN?

The first column of Table 4 contains each component's reference

Real Time Devices

"Accessing the Analog World"

Quality U.S.-manufactured PC Bus cards and software for single user, OEM, or embedded applications.



ADA2100 - \$395

- 4 Diff./8 S.E. 12-bit 20 μ s A/D inputs
- 12 μ s or 6 μ s A/D optional
- Programmable gains of 1, 2, 4, 8, & 16
- Three 8-MHz timer/counters
- Two 12-bit D/A outputs
- 16 digital I/O lines
- Dedicated GND adjacent to each analog signal
- Supports twisted-pair or ribbon cable
- BASIC, Pascal, C, & Fortran source code!

NEW PRODUCTS

AD3100 8 Diff./8 S.E. 12-bit 5 μ s A/D inputs; 200 kHz thruput; advanced A/D architecture, FIFO-based memory; pacer clock; external trigger; DMA transfer: three 8 MHz T/Cs; 16 digital I/O lines; excellent for signal processing & instrumentation \$589

VF900 2 Diff. inputs; 20-bit V/F type A/D; variable resolution & conversion speed; 16-bit @ 16 Hz; superb stability; self-calibration; excellent for chromatography \$410

FG400 dual 12-bit 2048-point FIFO-based function generator with output summation; up to 5 MHz data rate; external trigger; programmable output filter; repetitive or single cycle operation \$585

POPULAR PRODUCTS

AD1000 8 S.E. 12-bit 20 μ s A/D inputs; 12 or 8 μ s A/D optional; three 8-MHz T/Cs; 24 digital I/O lines. \$275

ADA1100 AD1000 with 2-channel D/A and resistor-configurable gain \$365

ADA2000 8 Diff./16 S.E. 12-bit 20 μ s A/D inputs; 12 or 8 μ s A/D optional; two 12-bit D/A outputs; prog. gains of 1, 2, 4, 8, & 16; three 8-MHz T/Cs; 40 digital I/O lines \$489

AD500 8 S.E. inputs; 12-bit integrating A/D; prog. gains of 1, 10, & 100 \$259

ADA100 Single-channel AD500 with differential input and I-bit D/A output \$219

DA600/DA700 Fast-settling 2/4/6/8 -channel 12-bit D/A; double buffered \$192/359

DG24/48/72/96 TTL/CMOS digital I/O cards with up to 96 lines; 8255-based; Optional buffers and pull-up resistors .. \$110/256

TC24 Five 5-MHz timer/counters; uses powerful AM95 13A chip; 24 digital I/O lines from TTL/CMOS-compatible PPI chip ... \$218

ATLANTIS Data acquisition software; rates to 25 kHz; background/foreground operation; pull-down windows \$150

PEGASUS Data analysis software; FFT; linear regression; graphical integration; hypertext help; mouse support \$250

Custom/OEM designs on request!

Real Time Devices, Inc.
State College, PA USA

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

designator. The second column holds the component's value. The third column is the calculated "gain" of each component. For resistors, it describes the node voltage change for each one-ohm change. For other components, it describes the node voltage change for each unit change of the component's value. The final column shows the node voltage change for each percent change of the component's value.

Examining the last column of Table 4 will answer a very difficult question: If we wish to keep the output of the amplifier to within one least-significant-bit error for an eight-bit analog-to-digital converter, what minimum tolerances are needed for each of the parts? The answers are R1, 1,095%; R2, 1.1%; R3, 330,000,000%; R4, 4%; R5, 4%; R6, 2.84%; R7, 2.94%; R8, 0.74%; R9, 0.74%; and finally the thermistor, 1.1%. Unfortunately the thermistor is only economically available in 5% tolerance. The output change due to just the tolerance of the thermistor will be 4.5 bits. Except for R8 and R9, the rest of the gain-setting resistors can be standard 1% parts. Resistors with tolerances better than 1% are difficult to find and purchase. R8 and R9 will be implemented with 1% resistors. They can cause errors of up to 1.35 bits.

One of PSpice's greatest values is its ability to perform Monte Carlo analysis. A Monte Carlo analysis varies the value of specified components through a statistical analysis. Essentially, for each run through the analysis, a new random number between -1 and +1 is generated based on a bell-shaped distribution for each part with a tolerance specified in the circuit listing. The random number is multiplied by the part tolerance and finally by the part's value. Each part that has an assigned tolerance gets a unique random number in each run. The result is then used in a PSpice DC, AC, or transient analysis. If many runs are made, say 200 or so, then the change of voltage at any circuit node due to component tolerances over a production run can be determined.

In Listing 5, each resistor has a model called out which defines its individual tolerance. In the Monte

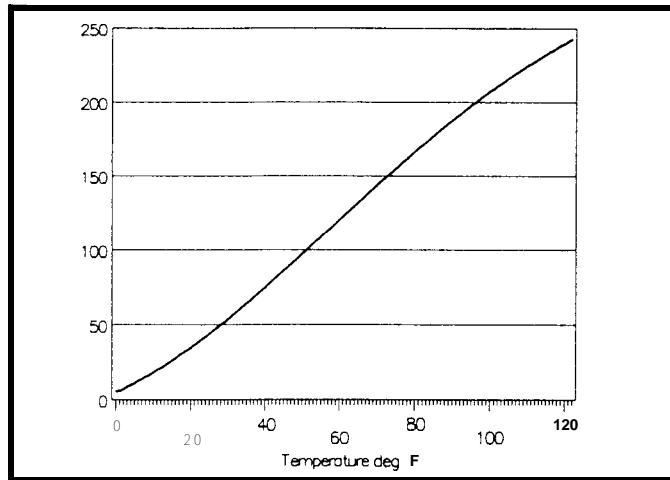


Figure 10—The output from PSpice for the low-resolution circuit is nearly linear.

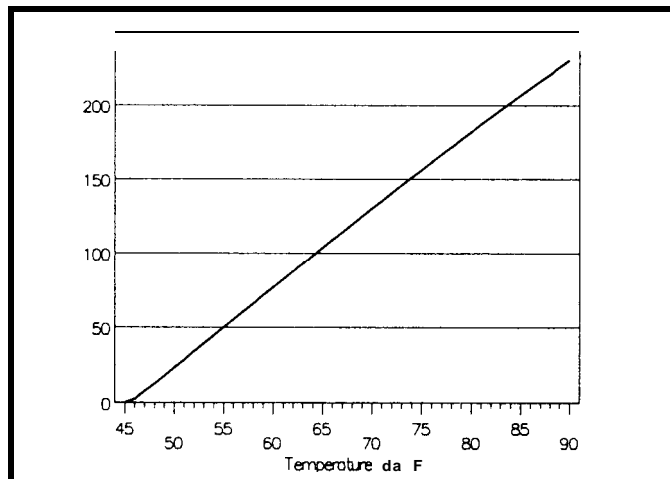


Figure 11—The output from the high-resolution circuit is even closer to linear.

Line Segment	Intercept	Slope	Percent	End Points °F
1	-2.470	0.6659	99.55%	0-20
2	5.803	0.4489	99.95%	20-100
3	-29.490	0.6224	99.87%	100-122

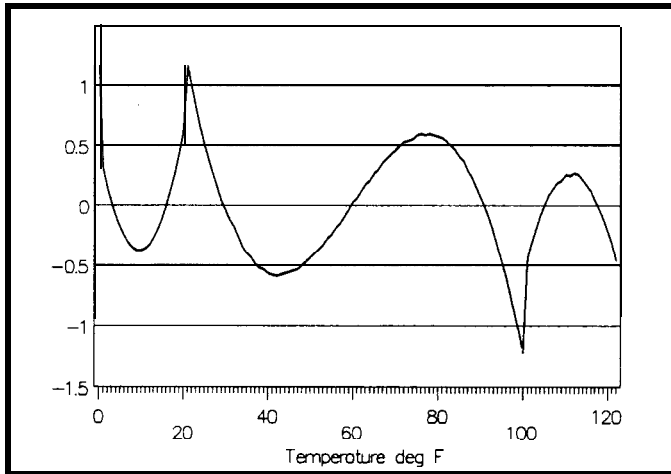
Table 1—Results from spreadsheet analysis give line-segment approximations to the output of the low-resolution circuit.

Carlo section of the listing, the models are defined. A Monte Carlo analysis of 200 runs is called out, with V(7) being the node of interest. A Monte Carlo analysis can only be carried out during an active analysis mode run such as DC, AC, or transient analysis. If we want to define the circuit components without sweeping any of the values, a dummy change variable has to be used. V100 and R100 perform that function. They are not part of our signal conditioning circuit, but they can be swept in a DC analysis to satisfy the requirement of an active analysis mode.

A lot of data will be generated and a lot of time will be used in a Monte Carlo analysis. The data will be writ-

ten to an output file. Again, we can clip out the data of interest and place it into a spreadsheet for analysis. Figure 14 is the result of 200 Monte Carlo runs with RTH held to 77°F, 10,000 ohms ±5%. It shows a statistical distribution of the output voltage at node 7. At 50 on the X-axis, you have an equal probability that the output voltage will be above or below 3.11 volts. At 100, the probability is that the output voltage will be entirely above 2.98 volts. At 75, the probability is that 25% of the units will have an output voltage below 3.075 and 75% of the units will have an output above 3.075. When the graph's data is changed into engineering units, the output will vary from +2.5° to -3° about 77°F.

Figure 12—The differences between the actual temperature and the computed temperature for the low-resolution circuit are within 1.4° across the entire range.



Line Segment	Intercept	Slope	Percent	End Points °F
1	-3.546	0.7658	98.89%	0-9
2	-0.878	0.6060	99.94%	10-20
3	3.547	0.4884	99.93%	21-42
4	7.924	0.4312	99.99%	43-76
5	-1.124	0.4865	99.93%	77-100
6	-21.433	0.5554	99.96%	101-112
7	-41.765	0.6746	99.97%	113-122

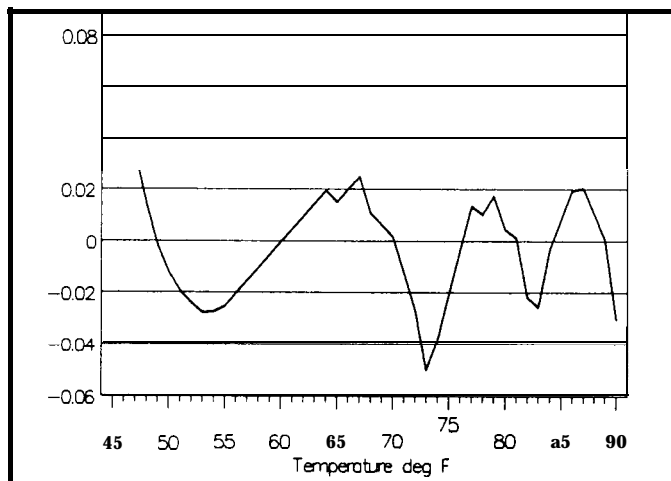
Table 2—Spreadsheet regression analysis can be used to analyze the errors.

Just for fun I checked circuit performance with different tolerances for the parts. If we were able to get 0.1% tolerance parts for the bridge and gain-setting resistors, the worst case performance at 77°F would be +1.98° to -2.55°. If we were able to get 0.01% tolerance parts for the bridge and gain-setting resistors, the worst case performance at 77°F would be +0.91° to

-1.17°. Finally, if we were able to get 0.1% tolerance parts for the bridge and gain-setting resistors and a 1% tolerance thermistor, the worst case performance at 77°F would be +0.26° to -0.85°.

Each Monte Carlo case of 200 runs takes about 10 minutes on my true blue IBM PC and the next step burns up hours and hours of CPU time. For

Figure 13—A diagram similar to that above for the high-resolution circuit shows even closer results.



Line Segment	Intercept	Slope	Percent	End Points °F
1	45.643	0.1869	99.99%	46-73
2	44.20442.004	0.1967 0.2079	99.99% 99.99%	73-90 74-82

Table 3—The results are analyzed with a spreadsheet for the high-resolution circuit.

HOME AUTOMATION BLOWOUT SUPER SPECIALS!

Mini-Controllers Mastervoice
ONLY \$9 LIST \$12.99 50% off! Voice control and more. Call for info and pricing.
 LIMIT 4 X-10 Powerhouse or Stanley

Module Madness!
 Lamp modules, wall switch modules, and appliance modules (2-pin) at an unbelievable low price! Only 864 modules reserved at this special pricing, so order now!
ONLY \$8 LIMIT 5. MIX & MATCH LIST \$12.99

GE Homeminder ONLY \$109
 We believe we have the last units available in the universe! Originally sold for almost \$600! Schedule your home using graphic icons on your TV! Remote phone interface, memo pad, security mode, message system, password protection, more. Complete with lamp module, appliance module, infrared remote control, and cables. Limited quantities and going fast, so order now!

Stanley Wireless Motion Detector
 X-10 Compatible! Indoor/outdoor operation. Works with X-10's RC5000 Base Transceiver.
LOWEST PRICE EVER! WITH BASE ONLY \$29 LIST \$39.99 LIMIT 24
ONLY \$44 LIST \$64.99 LIMIT 12

X-10 Universal Modules
 Momentary or continuous dry contact output and/or audible signal. Control sprinkler systems, garage door openers, drapery motors, door strikes, etc.
ONLY \$18 LIST \$24.99 LIMIT 16

TW523 Two-Way Powerline Interface
ONLY \$29 LIST \$39.99
 This two-way interface is great for development of your own PC-based or standalone X-10 home control projects. Purchase of this module includes permission to transmit patented X-10 code format. Complete with comprehensive X-10 technical data.

Useful Home Automation Plans
HCC-PLN1 Add local dimming (from the switch itself) to X-10 Dimmer Switch! Capability exists for Leviton's specs. Cut two parts and add jumper. Complete plans \$6.
HCC-PLN2 Control your Hi-Fi's volume with X-10 dim/bright buttons! Complete plans \$6.
HCC-PLN3 Cable television secrets: watching pay channels. For informational purposes only. Plans \$9.
HCC-PLN4 Modify X-10 modules for momentary on or off. Modify mini-controllers for momentary on or off control; just add a jumper wire! Complete plans \$6.
HCC-PLANS Get all four plans for only \$21!

One For All Remote
 Replaces 13 remotes! TV, cable, VCR, Hi-Fi, satellite converter, more! X-10 Compatible when used with infrared command center.
ONLY \$87 LIST \$129 infrared Command Center \$29.95
 LIMIT 3

Leviton Decora Wall Switch
 Use to control fluorescent lighting, ceiling fans, motors, etc. X-10 compatible. White, Ivory or Brown. Model 6291. List \$72.50. HCC \$48

Sundowner Controller
ONLY \$15 LIST \$19.95
 LIMIT 4
 Photocell turns up to 4 lights on at dusk, and off at dawn. Also functions as a mini-controller.

GUARANTEED LOWEST PRICES! Call for details.
ORDER REQUIREMENTS \$100 minimum per order. No minimum quantities. We accept Mastercard, Visa, check, or money order. Limited time specials.

HOME CONTROL CONCEPTS
 ORDER HOTLINE (ORDERS ONLY)
1-800-828-8537
 Info & Customer Service
 1-61 9-693-8887

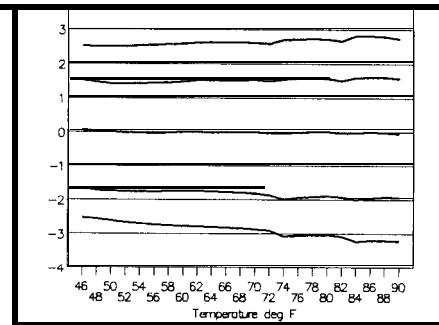
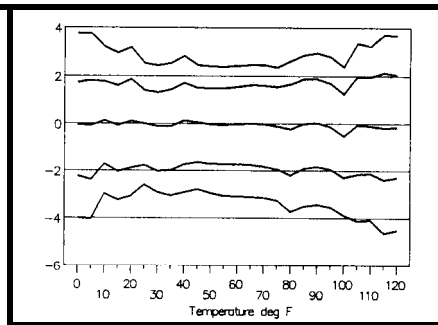
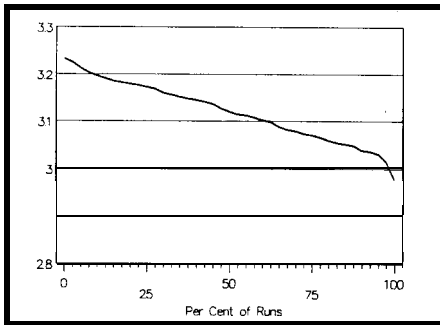


Figure 14—Two hundred Monte Carlo runs show a statistical distribution of the output voltage at node 7 of the circuit.

Figures 15 & 16—Statistical data for the low- and high-resolution circuits show the nominal output error (center), the worst-case performance (top and bottom), and the 80th percentile results (second from top and second from bottom).

the Monte Carlo analysis to work properly we need to explicitly set the thermistor's resistance value for each temperature. For the low-resolution circuit, let's make a PSpice input file for every 5°F, or 23 files. For the high-resolution circuit, we will make an input file for every 2°F, another 23 files. Next, create a batch file that repeatedly calls PSpice, feeding in the input files one at a time. It will take about eight hours for all the computing to be completed. It gives you enough time to finish reading the latest issue of *CIRCUIT CELLAR INK*.

Now we have 46 files of data taking up about two megabytes of room. We need to extract the relevant data from the output files and get it into a

spreadsheet for further analysis. When the data is in the spreadsheet, we will throw away 97.5% of it to produce our next graphs. We will only keep five lines of data: the center line, which is thenominalcase; theoutsidetwolines of data that represent the worst case performance; and the two lines that are twenty lines away from the worst case data. These last two lines represent the 80th percentile data. Graphs of this data are shown in Figures 15 and 16.

The center line defines the performance of the two circuits if all parts have their nominal values. The next two lines define the circuit's performance for 80% of the circuits built in a large production run. Theoutsidelines

define the circuit performance for 100% of the circuits built.

A TREMENDOUS ADVANTAGE

Wow, what a lot of work! As you stagger away from the computer console, after your first solo design effort, you must realize that you are not finished yet. All you have done is determine the theoretical performance of the circuit. Next, a prototype must be made. You have to check the performance to both normal and abnormal input conditions. After all, we made some very extreme simplifications in our op-amp models to make them fit into the student edition of PSpice.

But now you have a tremendous advantage: you know exactly how the circuit is supposed to work. If the circuit does not work as PSpice says it should there are three reasons why: the physical layout is incorrect, the circuit is interconnected wrong, or the parts are faulty.

I have designed a similar input circuit for a **microcontroller** using these techniques. The software was coded usinglinesegmentsasdeveloped here. All of the units worked as specified and there **havebeen** no circuit changes throughout production. ❖

Mark Nurczyk is a Registered Professional Engineer with 19 years experience in analog and digital design. He works for a large OEM designing microcomputer- and analog-based machine controls.

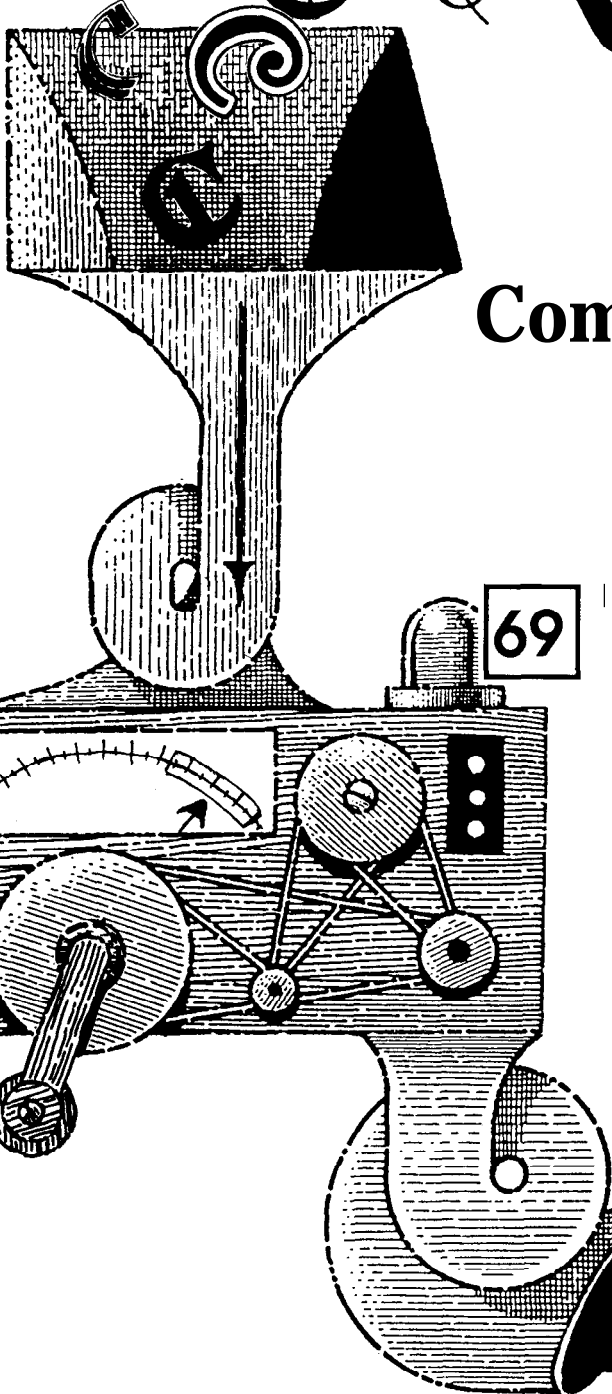
DC SENSITIVITIES OF OUTPUT V(7)

ELEMENT NAME	ELEMENT VALUE	ELEMENT SENSITIVITY (VOLTS/UNIT)	NORMALIZED SENSITIVITY (VOLTS/PERCENT)
R1	1.000E+01	-1.784E-04	-1.784E-05
R2	1.180E+04	1.513E-04	1.786E-02
R3	1.000E+04	-5.902E-13	-5.902E-11
R4	1.300E+05	-3.763E-06	-4.892E-03
R5	1.870E+05	2.616E-06	4.892E-03
R6	1.180E+00	-5.808E-04	-6.854E-03
R7	1.020E+04	6.510E-05	6.640E-03
R8	1.300E+05	-2.020E-05	-2.626E-02
R9	1.870E+05	1.416E-05	2.647E-02
RTH	1.000E+04	-1.784E-04	-1.784E-02
X1.RIN	1.000E+12	-5.088E-25	-5.088E-15
X1.RG	1.000E+05	-3.301E-10	-3.301E-07
X1.ROUT	5.090E+01	-3.763E-11	-1.915E-11
X2.RIN	1.000E+12	1.171E-23	1.171E-13
X2.RG	1.000E+05	7.618E-10	7.618E-07
X2.ROUT	5.090E+01	-6.379E-11	-3.247E-11
V1	5.000E+00	6.248E-01	3.124E-02
D1 SERIES RESISTANCE			
RS	1.600E+01	6.220E-20	9.952E-21
D1 INTRINSIC PARAMETERS			
IS	1.000E-13	-2.217E+04	-2.217E-11
N	1.000E+00	0.000E+00	0.000E+00
D2 SERIES RESISTANCE			
RS	1.600E+01	-5.310E-20	-8.497E-21
D2 INTRINSIC PARAMETERS			
IS	1.000E-13	2.217E+04	2.217E-11
N	1.000E+00	-0.000E+00	-0.000E+00

Table 4—The DC sensitivities for the circuit are computed by PSpice.

IRS

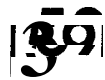
4 13 Very Useful
4 14 Moderately Useful
4 15 Not Useful



Compilers for the S051

Oh Say, Can You C?

by M. Scoff Martin,
Tim McDonough,
& Curtis Franklin, Jr.



High-Level Languages for Microcontrollers

by Ed Nisley

69

Using High-Level Languages on
Embedded Controllers

by Ken Davidson

72

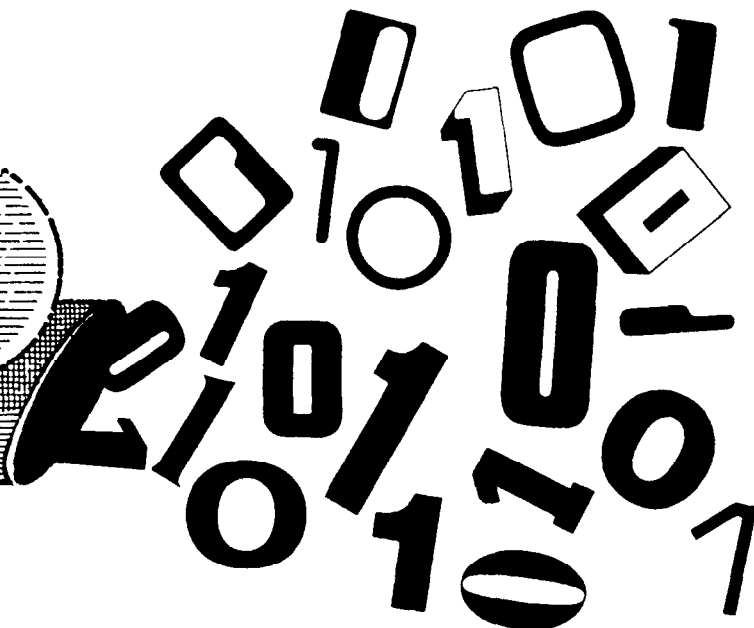


Illustration by Trish Fabish

M. Scott Martin, Tim McDonough, & Curtis Franklin, Jr.

Oh Say, Can You C?

CIRCUIT CELLAR INK Evaluates Three C Compilers for the 8051

In the beginning was the language, and the language was assembly. And the programs were dense and without structure. Sometime later, Kernighan and Ritchie said "Let there be C" and there was C. And the C programmers were divided from the assembly language programmers, and the C programmers were called "frustrated" and the assembly language programmers were called "exhausted." And the ANSI C committee was formed, and it looked about and saw what had been created, and called it Good, and cast it into stone. Then came Real-World Programming Problems, and they appeared in the form of a serpent; and the serpent tempted the programmers to chuck it all and get MBAs instead..

OK, so maybe it didn't happen exactly that way, but the struggle to find a more productive programming language than assembly can certainly tempt programmers to take some fairly drastic actions. In the last few years, the language of choice for almost every cry programming task has been C, and C cross-compilers have appeared for processors ranging from the 68030 to the 8051. We decided to take a look at three of the compilers available for the 8051 family: Avocet, BSO/Tasking, and Franklin.

GREAT EXPECTATIONS

It's difficult to know what to expect from any software development tool, but the difficulties are more pronounced when you're trying to develop software for an embedded system. C carries with it a particular set of difficulties, most of which have to do

with the twin bugaboos of standards and portability.

The ANSI standard committee has developed a position on C that most software developers applaud as reasonable and practical. Unfortunately, the majority of software developers work on desktop or large-system applications, not embedded and control applications. The result is a stan-

**One key question:
Is this package
worth my time
and money?**

dard that leaves something to be desired as a basis for embedded applications development. In particular, workstation and mainframe programmers tend to deal with a limited I/O universe and need a great deal of power and flexibility in dealing with data structures. The ANSI standard supports these requirements. Embedded applications, though, generally have fairly simple data structure needs and an almost infinite variety of I/O. To meet the needs of the embedded system programmer, a compiler writer must deviate from the standard. The nature of the deviation is different in each compiler, reflecting the experience and prejudice of the chief compiler architect. Whether you agree with those deviations will, in large part, determine how you view the compiler.

Portability is a most seductive issue. The concept of writing code for one system and easily porting it to an almost unlimited number of other systems promises to save countless programming hours. Unfortunately, the differences between I/O complements and disparities between micro-controller feature sets are simply too great to allow effortless porting.

THE BIG ISSUES

There are two questions that loom large when any assembly language embedded application programmer thinks about a change to a high-level language: "Will the language let me do what I need?" and, "Will the program fit into available memory?" In the case of the C packages we looked at, the answer to the first question is almost certainly yes. The second question's answer will depend entirely on your system and your application. When you stop to think about it, there's not much difference (in these respects) between C and assembly language.

If there's not such a huge difference, then why switch? There are several possible responses, including conforming to company standards, "modernizing" your tools, portability, and less software development time, but the answers all boil down to this: In general, productivity increases when you use C, compared to when you use assembly language. When you remember that your time is usually the most expensive component in any design, the arguments in favor of switching can look compelling.

We approached the packages as working programmers and engineers,

and tried to answer one key question: Is this package worth my time and money? For details of our evaluation philosophy and procedures, see page 71. For the evaluations themselves, just keep reading..

AVOCET AVCASE C

*Principal evaluator—
Tim McDonough*

When independent consultants and engineers in small firms think about cross-compilers, Avocet is often the first name that comes to mind. This Maine-based company has been producing cross-assemblers, cross-compilers, and development tools for many years. We looked at a C development system consisting of AvCase C1.216, AvCase 8051 Assembler 1.215, AvCase Simulator/Debugger 1.06, AvCase 8051 Make 1.206, and AvMake. Tests were run on an 10-MHz XT clone with 28-ms 40-mega-byte hard disk. Code was tested using Cottage Resources Corporation's Control-R I, Control-R II, and Data-log-R I 8031 single-board computers. All three boards use an 11.0592-MHz crystal in the clock circuit.

The simulator was of marginal value in these tests since it requires a minimum of 550K of memory and my development machine is an XT-type system. I run a pretty sparse system and have about 560K available for applications. The `READ, ME` files indicate that a version is under development that is overlaid to provide more room for the code you're debugging.

Within the AvCase environment, a Norton Guides database makes pop-up help available. Since Norton Guides is a TSR, it eats up precious RAM that the simulator would like to have. Altogether, the AvCase environment looks to have been designed with the assumption that the user will have an 80386 with several megabytes of memory at his disposal. The trend is certainly in that direction, but I hate to see companies abandoning users of older machines.

An 8086 native mode compiler is provided to test algorithms with a C compiler that is as close as possible to

the 8051 cross-compiler. Nice touch. I don't know how close it really is but since it makes `.asm` files and then links them, you can make some tiny executables for MSDOS machines.

The compiler has options to generate special code for in-circuit emulators (such as those from Nohau) to break after the breakpoint set by the user instead of before. The code generated with this option puts a `NOB` before each instruction that would in-

You should be able to shoehorn a C program into even the most limited system.

crease code size, but would seem to make debugging easier if you had the space to spare in your ROMs. I didn't have the equipment to test this feature but the fact that they supported it seemed good.

A variety of memory models are supported that use internal RAM only, external RAM, internal and external stack locations, and shared address space Code and Data memory. The flexibility of the various models is important given the variety of configurations hardware designers thrust upon programmers. Additional flexibility is provided through conditional assembly and the production of fully relocatable code.

One of the advantages of any high-level language is that data structures are more easily implemented than in assembly language. While most embedded applications don't call for complex structures, AvCase C supports all of the standard C data structures, including bit fields, unions, and structs.

C is based on the concept of a small "kernel" and complex libraries that can be linked in as needed. AvCase

C offers good support in its libraries, providing 73 functions in the standard library. Two of the most common functions for fast and easy I/O are `printf()` and `scanf()`; both are available in the AvCase C library. In addition to the standard I/O and integer math functions, a complete floating-point package is provided. As important as the library functions are, they can become useless and frustrating unless you know exactly what code goes into each function used. In this case, Avocet has done things correctly.

Source for the math routines and all library functions has been provided. Source language availability not only makes debugging much easier, it allows programmers to modify library functions to meet their peculiar needs. Most of Avocet's library routines are written in C, but a few have been written in assembly language, presumably to make them faster.

CODE SIZE

If the smallest program you can generate takes up 400K, you have a compiler that's of little use in the 8051 world. Of course, the designers of cross-compilers know this, and they've obviously worked hard to keep code size to a minimum. In the case of some of the more common library functions, the "size cost" is as follows:

<code>printf()</code>	411 bytes
<code>sqrt()</code>	2243 bytes
<code>atoi()</code>	467 bytes
<code>strcpy()</code>	107 bytes
<code>strlen()</code>	66 bytes
<code>cos()</code>	2867 bytes

Obviously, if you link every possible library into your program, you're going to end up with a file that's considerably larger than it needs to be. If you're judicious in your library use, and careful in choosing link and compile options, you should be able to shoehorn a C program into even the most limited system.

Just for fun, I decided to write the smallest program possible. I came up with:

```
void main( void )
{
}
```

This gem of the programmer's art compiles and links to 130 bytes.

Even with the best C code, there are some situations where you simply must drop down into assembly language for performance, precision, or size reasons. AvCase C does not provide for in-line assembly language, but it does allow assembly language functions to be linked in, called, or added through assembly language macros.

DOCUMENTATION

Documentation was provided in three separate manuals. The editor, Make, Assembler, and Linker manuals were very good. The manual on the compiler itself has a nice tutorial section to get the developer up and running quickly, but a lot of information such as "Where are `STDIN` and `STDOUT`?" has to be gleaned from the source code. Some of the source code I examined could have been commented a bit more thoroughly, too. It

would be nice to have a discussion of the various header files and more information on 8051 specifics of the compiler.

The overall quality of the Avocet documentation was very good. I didn't think that the C compiler documentation was quite as thorough as that for the utilities, but it was adequate and supplemented by a copy of "The C Programming Language" by Kernighan and Ritchie in an updated version that reflects the ANSI standard.

I didn't need to call on Avocet Technical Support during my evaluation, but telephone support is available at no charge. Avocet also has a reasonable update policy.

...AND OVERALL

Serial I/O can be a moderately complicated affair on the 8031. Using the `8051.h` and `stdio.h` files I was able to get the code in Listing 1 compiled, burned, and running on a Con-

trol-R I single-board computer in under five minutes. The code size produced (3781 bytes) seems very reasonable and overall performance is very good.

I was particularly pleased with the small memory model of the compiler. This model uses only internal RAM for stack and variables (128 bytes in an 8031). Serial communications and general bit manipulations of ports 1 and 3 were easily **accomplished** even on a bare bones 8031 circuit consisting of the CPU, address latch, 8K EPROM, and a MAX232. I had always operated under the assumption that it was "assembly only" for such a minimal system and was glad to learn that in some cases I was wrong. The larger memory models performed well also and provided the developer with a fairly broad choice of supported hardware configurations.

Avocet 8052 C Compiler, Version 1.3—\$995 includes compilers, assembler, linker, utilities

BCC52

BASIC-52 COMPUTER/CONTROLLER



The BCC52 Computer/Controller is Micromint's hottest selling stand-alone single-board microcomputer. Its cost-effective architecture needs only a lower supply and terminal to become a complete development or end-use system, programmable in BASIC or machine language. The BCC52 uses Micromint's 80C52-BASIC CMOS microprocessor which contains a ROM-resident 8K-byte floating-point BASIC-52 interpreter.

The BCC52 contains sockets for up to 48K bytes of RAM/EPROM, an "intelligent" 2764/128 EPROM programmer, three parallel ports, a serial terminal port with auto baud rate selection, a serial printer port, and is bus-compatible with the full line of BCC-bus expansion boards. BASIC-523 full floating-point BASIC is fast and efficient enough for the most complicated tasks, while its cost-effective design allows it to be considered for many new areas of implementation. It can be used both for development and end-use applications.

PROCESSOR

- 80C52-BASIC, 8-bit CMOS microcomputer
- jumper-selectable conversion to 80C31/80C32 functionality
- 8K bytes ROM (MI BASIC interpreter)
- 256 bytes RAM
- three 16-bit counter/timers
- 32 I/O lines
- 11 MHz system clock
- 6 interrupts

Memory

- expandable to 62K bytes
- five on-board sockets
- up to four 6264 (8Kx8) static RAM
- either an 8K 2764 or 16K 27128 EPROM

Input/Output

- console I/O RS-232 serial port
- line printer RS-232 serial port
- three 5-bit programmable TTL-compatible parallel I/O ports using a 8255 PPI
- alternate console RS-422/RS-485

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

BCC52	BASIC-52 Controller Board mm 8K RAM	Single Qty.	100 Qty.
BCC52C	Lower-power all-CMOS version of the BCC52	\$189.00	\$149.00
BCC52I	Full industrial temperature range	\$294.00	\$220.00
BCC52CX	CMOS, Expanded BCC52 w/32K RAM	\$259.00	\$159.00

MICROMINT, INC. 4 Park Street, Vernon, CT 06066

Elegant, concise, fast & standardized

FLOATING POINT

libraries for embedded applications

Based on the IEEE 754 standard, FPAC (32 bit) and DPAC (64 bit) libraries are mature, well documented, and fully tested. The libraries are fully ROMable and include the following:

- Basic Operations
- ASCII Conversion
- Square Root
- Integer Conversion
- Trigonometric
- Logarithmic

US Software supports most Intel, Motorola, Zilog and Hitachi micros, including 80X86, 80386, 680X0, 80960, 8051, 8096, 68HC11, 280, 6809 and 6301.

For additional information, please contact:



U S SOFTWARE

United States Software Corporation
 14215 NW Science Park Drive
 Portland, Oregon 97229
 800-356-7097
 503-641-8446
 503-644-2413 < F A X >

C Compilers and AvCASE Simulator—\$1895 purchase includes unlimited support by telephone, fax, in-house BBS, and UUCP
 Avocet Systems, Inc.
 120 Union Street
 P.O. Box 490
 Rockport, ME 04856
 (800) 448-8500

FRANKLIN C COMPILER 5.1
Principal Evaluator—
M. Scott Martin

I tested C Compiler 5.2 using a Dell 310 (80386 ISA computer) and a Micromint RTC stack consisting of an RTC31, RTC-SIR, and RTC-IO. The RTC31 contains RTCMON and 32K RAM. I was able to assemble the manual materials and get the software installed in approximately one hour. The system, as delivered, took up about 2 MB of hard disk space. Before I start answering the specific questions that arose during the evaluation, I'll describe what I tried to accomplish with the Franklin package.

FROM THE OLD TO THE NEW

I've been using the Avocet assembler and simulator for approximately three years. As a result, I've developed a number of libraries for developing applications for the 8031. Some of the first testing I did was to determine what would be required to adapt these libraries to be callable from a Franklin C program.

The Franklin linker will not accept Avocet-generated .obj files, but I was able to modify the original assembly language files so the Franklin linker would work. The changes made were primarily related to syntax differences regarding include files, assembler directives, variable declarations, and removal of "PROC . END-PROC"-type constructions. Once these changes were made, I could reassemble the source files using the Franklin assembler and link them as **well**. The routines which require parameters to be passed from the caller must be further modified to accommodate

the Franklin C-t-assembly interface. There is no in-line assembly support in the Franklin C compiler.

Parameter passing to a callable assembly routine requires that "local" storage (which is not truly local) be established in each routine for the parameters to be passed into. For example, consider a function named `sir_mess-out` which writes a null-terminated string out to the RTC-SIR serial port. My original routine expects DPTR to be pointing to the desired, null-terminated string upon entry. If the string is named `S1`, the C calling sequence `sir_mess_out (&S1);` will work only if the assembly routine is modified to contain three bytes of local storage to receive the pointer from the caller. (Franklin uses three bytes for pointers: one byte to identify the type of memory in which the data resides [i.e., `idata`, `xdata`, `code`, etc.], and the remaining two contain the physical address.) The data area for this routine must be defined as follows (in the assembly source file):

MICON-196KC CONTROLLER - HARDWARE & SOFTWARE DEVELOPMENT KIT
"THE COMPLETE DIGITAL CONTROL SOLUTION"™

MICONA CORPORATION is offering for the first time in one package everything necessary to implement a COMPLETE DIGITAL CONTROL SOLUTION at a very affordable price.

The MICON-196KC will be appreciated by hardware and software engineers and till provide students with an inexpensive way to gain familiarity and expertise in the world of microcontrollers.

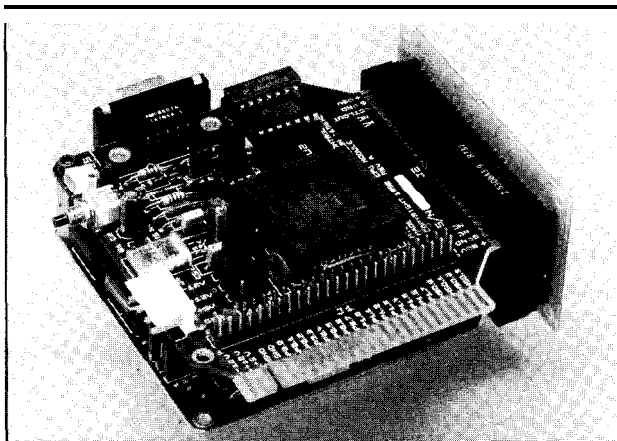
MICON-196KC Controller is ideally suited for education either at public institutions or for private study.

High-performance real-time control, with such applications as high-speed closed-loop motion control, midrange digital signal processing and intelligent data acquisition, etc. can be achieved with the MICON-196KC Digital Controller.

The MICONA's Complete Digital Control Solution comprises of 2 kits:

1. The Hardware-Core Digital Controller Kit contains:

- 3.5" by 3.5" CPU Module featuring the INTEL 80C196KC 16 bit embedded controller operating at 16 MHz, with eight ADC channels with sample and hold at a 0.8 microseconds rate, three(3)



- pulse-width-modulated outputs (DAC), one DMA channel, six high-speed outputs for pulse and waveform generation, four high-speed capture inputs with 1-microsecond resolution, one full-duplex RS-232 serial port, and five 8-bit I/O ports.
- 3.5" by 3" MEMORY Module with 64 K memory space configurable as RAM, EPROM, or a combination of both.

-PROTO Module, and two(2) BUS Modules.

2. The Software Development Kit contains:

- a PC based Universal Assembler Including 80C196 Machine Language,
- a powerful Software Development tool - SYSMON - and.

- a User's Guide with HARDWARE SCHEMATICS and APPLICATION DEMO PROGRAMS.

MICON-196KC - THE COMPLETE DIGITAL SOLUTION - sells for: \$345.00

Power & communication hardware kit COM-2000 consisting of a compact power supply and cables sells for: \$95.00

Assembly language tutorials: ALT-80C196 sells for: \$65.00

Quantity & Student discount starts at: 10%

MICONA Corp.
 MICON Division
 1885 Surveyor Ave.
 Simi Valley, Ca 93063

TEL:(818) 348-4992
 FAX:(818) 348-0960

```
SIRDATA SEGMENT DATA
PUBLIC ?sir_mess out?BYTE
RSEG
SIRDATA
?sir_mess out?BYTE:
MESS PTR: DS 3
```

One of the routines `sir_mess_out` is called, `DPTR` may be loaded by manipulating the parameter area where the caller has loaded the parameters as follows:

```
?sir_mess_out?BYTE+0 indicates the type of pointer
?sir_mess_out?BYTE+1 contains the high-order byte of the address
?sir_mess_out?BYTE+2 contains the low-order byte of the address
```

Having to modify each of my library routines this way would require a significant amount of effort, and code size will suffer somewhat from both the requirement of a separate parameter space for each called function and the coding necessary to manipulate the parameters within the routine.

Franklin provides the source code for `getchar()` and `putchar()` functions from their library and I did not attempt to modify these routines to support SIR communications, but their documentation suggests that these routines be used for that purpose. According to their documentation, modifying the `getchar()` and `putchar()` routines will redirect all I/O such as from `printf()` to the new destination, but I didn't try it. Source code for other library functions is not available.

DOCUMENTATION

There is quite a bit of documentation in the manual, but not enough for me. The package appears to possess many powerful features which are not documented clearly, so I have been unable to exploit them. For example, I was not able to get the Franklin monitor to work-1 had to fall back on RTCMON. Fortunately, I can properly ORG code using linker switches

to work with RTCMON, but it would have been more convenient to work with the monitor contained in the package. Another example of a problem is that the current manual doesn't come with an index, but I'm told the new one will.

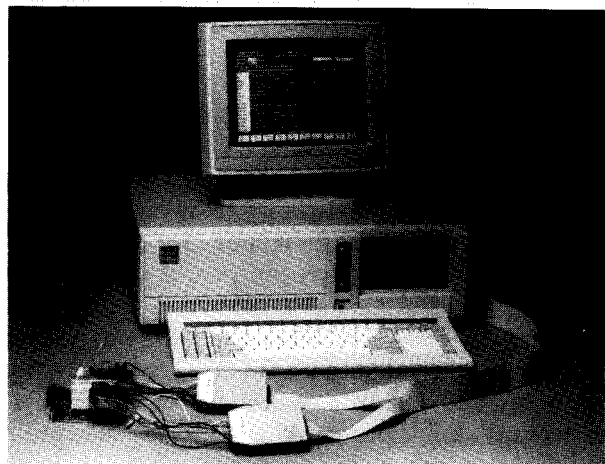
I had a number of conversations with Randy at Franklin Technical Support regarding their products, and I have no complaints with the tech support as it has always been courteous and capable. The down side is that the technical support doesn't come cheap-Franklin charges \$20 per hour for telephone time with their technicians.

LIBRARIES

As with all C compilers, the Franklin compiler's libraries are crucial to its utility. There are both positive and negative aspects to the package that Franklin presents. First the positive.

The standard library supplied with C Compiler 51 is complete, with

PC-Based Logic Analyzers



Sophisticated Logic Analysis at Unsophisticated Prices

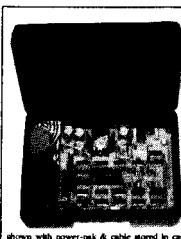
- *ID160 (50 MHz) for \$695
- *ID161 (100 MHz) for \$895
- 50 MHz or 100 MHz Sampling • 8K Trace Buffer • 32-channel Operation
- *Multi-Level Triggering *State Pass Counting
- *Event Timer/Counter *Performance Histograms *Hardcopy Output
- *Disassembles popular E-bit micros *and much more !
- *30 Day Money Back Guarantee



INNOTECH DESIGN, INC.
6910 Oslo Circle, Suite 207
Buena Park, CA 90621
Tel: 714-522-1469 FAX: 714-527-1812

MS-DOS EPROM PROGRAMMING SYSTEM NEEDS NO INTERNAL CARD

- EPROMS
- 2708, 2758, TMS2716*, 2716
 - 27C16, 2516, 2532*, 2564*
 - 68764*, 68766*, 2732, 2732A
 - 27C32, 2764, 2764A, 27C64
 - 27128, 27128A, 27C128
 - 27256, 27C256, 27512
 - 27C512, 27513*
 - 27011*, 27C011* (1 MEG)
 - 27010*, 27C010*
 - 27C1000*, 27C1001*
 - 27C020*, 27C2001* (2 MEG)
 - 27C040*, 27C4001* (4 MEG)



- EEPROMS
- 2804, 2816A, 28C16
 - 2817A*, 2864A, 28C64
 - 28256*, 28C256*, 52B13*
 - 52B33*

- MicroControllers
- 8741A*, 8742*, 8748*
 - 8748H*, 8749*, 8749H*
 - 8751*, 87C51*, 8752*
 - 8753*, 8744*, 68705*

*ADAPTER REQUIRED
Diagrams included with manual
Assembled adapters are available

CONNECTS TO YOUR SYSTEM'S

PARALLEL PRINTER PORT

- *A FAST, EASY-TO-USE SYSTEM WORKS WITH ANY DESKTOP OR LAPTOP MACHINE
- *EXPANDABLE, FLEXIBLE DESIGN SUPPORTS CURRENT AND FUTURE DEVICES
- *SUPPORTS 8, 16 & 32 BIT DATA FORMATS INCLUDING WORD SPLIT & DOUBLE WORD
- *READS AND CONVERTS INTEL HEX, MOTOROLA S-RECORD AND BINARY FILES
- *HARDWARE PROTECTED AGAINST DEFECTIVE AND INCORRECTLY INSERTED DEVICES
- *NO SOFTWARE INSTALLATION NECESSARY - PROGRAM TOTALLY SELF CONFIGURING

SYSTEM SOFTWARE COMMANDS

- | | | |
|----------------------------------|----------------------------------|--------------------------|
| *PROGRAM EPROM(S) FROM DISK FILE | *SAVE EPROM(S) OR BUFFER TO DISK | *COPY EPROM(S) |
| *READ DISK FILE INTO BUFFER | *PROGRAM EPROM(S) FROM BUFFER | *VERIFY EPROM ERASED |
| *READ EPROM(S) INTO BUFFER | *COMPARE EPROM(S) WITH BUFFER | *SELECT BUFFER EDITOR |
| | | *SELECT DEVICE TYPE |
| | | *DEVICE CHECKSUM |
| | | *SET BUFFER (0, 1, 2, 3) |

PLUS AN INTEGRATED BUFFER EDITOR WITH 18 BYTE LEVEL COMMANDS

SYSTEM INCLUDES: PROGRAMMING UNIT, POWER PACK, CONNECTING CABLE, OPERATION MANUAL & SOFTWARE \$289

SOFTWARE AVAILABLE ON 3 1/2" OR 5 1/4" DISK (PLEASE SPECIFY)
CALL ABOUT OPTIONAL ADAPTERS - A SOFT TRAVEL CASE IS AVAILABLE FOR \$19.00

TO ORDER SEND CHECK, MONEY ORDER, WRITE OR CALL:



ANDRATECH
P.O. BOX 222
MILFORD, OHIO 45150
(513) 831-9708
FAX (513) 831-7562



CALL OR WRITE FOR MORE INFORMATION - ADD \$5.00 FOR SHIPPING - \$4.00 FOR C.O.D.

67 functions including those most commonly used. The only functions I noted as absent were `malloc/free` types. As far as I can tell, there are no dynamic memory allocation functions in the Franklin libraries.

If your applications deal with either integer or floating-point math, the Franklin libraries will support you. If, on the other hand, you need to modify one of the library routines—even if you simply need to know what's happening inside a function for timing or debugging purposes—you're out of luck. The only source code provided is for `getchar()` and `putchar()`: Both functions were written in C. Franklin's documentation provides examples showing how to initialize timers and other I/O to make `getchar()` and `putchar()` work with your system.

CODE SIZE

The Franklin compiler provides three levels of optimization. Since I didn't have access to library code to test the effectiveness of the optimization, I simply left the switch set to highest level of optimization for all evaluations. The code size cost for some commonly used functions was:

<code>printf()</code>	1020 bytes
<code>atoi()</code>	286 bytes
<code>strcpy()</code>	156 bytes
<code>strlen()</code>	97 bytes
<code>rand()</code>	72 bytes
<code>cos()</code>	1597 bytes
<code>log()</code>	1456 bytes
<code>sqrt()</code>	1043 bytes

I feel that the code size is reasonable for all the functions shown. The functions are probably a little longer than they would be if I had hand-coded them in assembly language, but, then again, I didn't have to spend the time to code them.

DEBUGGING

I was easily able to get the simulator and debugger up and running. After doing that, I found that there are a couple of things that I don't like about the simulator. First on the list of problems is the fact that, once you

have loaded a program you wish to debug, no source code is shown in the code window. To view source code, you must disassemble your program, which can be done at the command line of the simulator/debugger as `u <address>`. When you do this, you get C source lines identified by line number only and the resultant assembler mnemonics. Setting breakpoints, watchpoints, and other debugging traps would be far easier if the C source were displayed along with the resultant assembly instructions when the code is unassembled.

It excites me to think that I could develop a program on the PC and simply post the working code.

The listings produced by the compiler and linker provide quite detailed information. The `.lst` output from the compiler shows which lines of C go with which line numbers: If you have a fresh hard copy of your program, identifying positions for breakpoints is not a problem. The `.M51` output from the linker resolves all addresses and provides symbol names, sizes, classes, and so on.

The debugger can create rather complex procedures to associate with watchpoints, providing great flexibility of action when a watch condition becomes true. An example might be listing the value of a variable on the screen every time a variable is read, written, or both. The second item on my list is that this is an area where the manual falls short since the commands, their uses, and effects are not thoroughly explained. It took quite a bit of exploring and experimenting to start to understand the full power of these features. Please understand, I like the concept of flexible procedures

in debugging, but I don't feel that the user should have to stumble around before finding their proper uses.

A built-in editor comes with the debugger for creating and maintaining the procedures. By saving the procedures to disk after using them, you can gradually create a powerful library of custom debugging features. This is especially useful in long debugging cycles, when you can create complex debug features, find bugs, go back to the editor/compiler for another pass, and quickly reinstall all of the previously created debug features for the next round.

INTERFACING TO WORK

Any change to a major new tool will require alterations in your work style. One of the changes the Franklin compiler caused in my routine hit hard because it touched on one of the most personal of all engineering/programming tools—my editor.

I use the Quick C editor for creating source code; I've used it for years. I like its interface to the mouse, and I've grown accustomed to its idiosyncrasies. With the Avocet assembler, I can edit a program then shell out to MAKE it using the assembler. If there are errors, fixing them is a quick and simple procedure. When I tried this with the Franklin package, I ran out of memory. Furthermore, the error message I got from the compiler was confusing—it took me quite a while to figure out the problem. The error reads: Fatal Error Allocating Memory... I thought there was some problem with the 8051 code I'd written. I made a number of changes in my program, altering the allocations of code and data between internal and external RAM, to no avail. I finally discovered that my editing cycle was to blame. To be fair, the manual states that there could be a problem since larger programs can take up to 512K in which to compile. I should have read the manual, but they should have made it easier to read. There should certainly be a way to distinguish between errors in a compiled file and errors with the compiler's operation!

Includes AT-compatible computer and Borland C++



Create custom-designed software and build a high-paying career with NRI's new training in PC Software Engineering Using C

With businesses spending close to \$80 billion a year on computer software, the demand is skyrocketing for trained professionals with the power to create custom-designed software applications. Now, with NRI's new training in PC Software Engineering Using C, you get the hands-on experience you need to succeed in this top-growth field. Now you can become one of today's highly creative software developers and cash in on the money-making opportunities of this multibillion dollar industry.

learn to do it all

NRI training lets you master the in-demand skills used to develop today's software applications. It's training that includes a powerful IBM PC/AT-compatible computer, programming software, and hands-on experience in C, today's hottest computer language. And it's the only training that prepares you to take on the real-world challenges of software engineering-at home and at your own pace.

With NRI training in PC Software Engineering Using C, you get every basic skill you need to engineer software that meets today's business and personal needs.

Your training includes an AT-compatible computer plus advanced training in C

Your NRI training gives you hands-on experience with a powerful AT-compatible computer system, including 2400 baud internal modem, 30 meg hard drive, VGA color monitor, and advanced programming software — including Borland C++ — all yours to keep.

Right from the start you get the basics you need to grasp important programming funda-

mentals. Then you move quickly ahead to discover the power of programming in C. As you learn to write tight, fast C code, you'll uncover the unique features that make the C language a key tool for building your software applications. And, with the Borland C++ software included in your course, you can take your skills even further to explore this powerful compiler's many features including overlapping windows, graphics, and all the other tools you need to build complete, reliable C++ programs.

By the time you complete your course, you'll know how to create a variety of business and personal applications including spreadsheets, databases, and text editors. Soon you're ready to write fast, functional programs that give you the competitive edge to advance on the job or get started in a money-making business of your own.

learn software engineering from start to finish

NRI training in PC Software Engineering gives

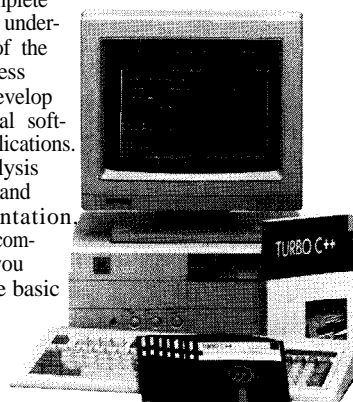
you a complete hands-on understanding of the total process used to develop professional software applications. From analysis to design and implementation, you're in command as you master the basic building

blocks for success as a software engineer. In the process, you'll discover a whole new world of career possibilities: networking, artificial intelligence, object-oriented programming, and more!

Your career in software engineering begins with your FREE catalog from NRI

NRI training is your blueprint for success in the exciting field of software engineering. Send for your free catalog and see how NRI gives you the experience and the know-how, the computer, and the programming software you need to advance on the job or build a high-paying new career. If the coupon is missing, write to NRI Schools, McGraw-Hill Continuing Education Center, 4401 Connecticut Avenue, NW, Washington, DC 20008.

AT is a registered trademark of IBM Corp. All Borland products are trademarks or registered trademarks of Borland International, Inc.



Only NRI gives you an AT-compatible computer with modem, VGA color monitor, 30 meg hard drive, and Borland's C++.

NRI Schools

McGraw-Hill Continuing Education Center
4401 Connecticut Avenue, NW
Washington, DC 20008



Check one FREE catalog only:

- PC Software Engineering Using C
- PC Systems Analysis
- Computer Programming

NAME _____ (please print) AGE _____

ADDRESS _____

CITY/STATE/ZIP _____ 4551-021

My approach to evaluating the Franklin C compiler was to try it while working on a current project. I didn't try to write fancy benchmarks. I did take an assembly language program and port it to C. My barcode reader software initially took 11 days to write in assembly language. I ported it to the Franklin C compiler in two days. The resulting program, which included liberal use of `sprintf()`, is comparable in size to the original assembly language version, and has much more debugging information going to the terminal. From this standpoint, I'm quite impressed. I simply wouldn't have believed that the final code could be so close to the final size before I tried it myself. Since the original algorithms were written (and debugged) in C on a PC, porting the code was amazingly simple. Since I have powerful C language debugging tools on the PC, it excites me to think that I could develop and debug the non-hardware-dependent portions of a program on the PC and simply port the working code.

If I were to buy the Franklin compiler, I would need some time to get used to the idea that I can't optimize the library code. For example, I wouldn't need the generality of a `sprintf()` in a shipping version of the code I produced, and I would prefer to trim it to the functions needed by the application. Without the source code, optimizing is impossible.

Franklin 8051 C Compiler, Version 2.51—\$995
 includes compiler, assembler, linker, and utilities
 Compiler and Franklin Simulator Debugger—\$1495
 purchase includes 90 days (from invoice date) free tech support
 Franklin Software
 888 Saratoga Ave, #2
 San Jose, CA 95129
 (408) 296-8051

BSO/TASKING C
Principal evaluator—
Curtis Franklin, Jr.

I had a couple of advantages over the other evaluators in this process.

The first was that I could draw on the experience of the **CIRCUIT CELLAR INK** engineering staff. The second was that I got to see the other two evaluations before finishing my own.

I received BSO's Tasking C 8051 Compiler v. 1.0.3, 8051 cross-assembler v. 2.2, and utilities v. 4.3 for evaluation. The C compiler and assembler produce code compatible with BSO's XRAY51 symbolic debugger. As we specifically requested C compilers for this evaluation, we did not receive XRAY51.

The Tasking compiler has the "look and feel" of a high-quality, professional product.

IN GENERAL

The first thing I noticed about the Tasking C compiler was the copy protection. BSO includes a Sentinel Pro parallel dongle from Rainbow Technologies with every package. If the software doesn't find the dongle at compile (or assemble or even installation) time, it generates a Fatal Error and stops. Now, I have heard the arguments for copy protecting professional software, I understand the concerns of the software vendors, and I still think that copy protection is a bad idea. I feel so strongly about this issue that I will not buy copy protected business software. You may have different feelings about this subject.

Excuse me while I rant. I tried to install the BSO utilities on my Zenith Z-286. According to the documentation supplied with the software, the Z-286 has been tested and found to work with the hardware key. Obviously,

they didn't test it with my Z-286. For nearly half an hour I was treated to a cascading progression of messages telling me that the hardware key isn't talking to the installation program. I'm told that the error is number 2 (my, isn't that a helpful bit of information), and that the installation is aborting due to my pitifully obvious attempt at software perfidy. BAH!

I gave the package to Engineering Staff member Ed Nisley, so that he could install it on his True-Blue IBM PC/AT. No go. Now, it's true that Ed has used his system for a number of hardware and software projects, but any modifications he has made to his system are within the range that any working system designer is likely to have made.

We finally tried to install the compiler on a Micromint OEM-286 AT compatible made up of a number of odd Taiwanese, Canadian, and U.S. parts. Lo, and behold, it installed and worked. Unfortunately, real-world deadlines and a computer owner who had to get work done for his day job meant that we couldn't provide the same library size numbers we have for the other compilers. That's too bad, because it means that BSO's mistrust of its customers didn't let us tell you about a package that otherwise looks absolutely top-drawer.

It's a genuine shame that I'm so angry about the parallel dongle right now, because the Tasking compiler does several things well. One of the more important is the way that Tasking handles assembly language routines. Like both of the other compilers, Tasking allows assembly language functions to be called or linked. In addition, the Tasking compiler uses the ANSI standard to good effect.

BSO wisely says that the Tasking compiler is "based on" the ANSI standard. By this, they mean that they've read the standard and used it where it made sense: They haven't been strait-jacketed by the standard. One of the more interesting features of ANSI C is the pragma. Pragmas are loosely defined as sections of implementation-specific code. The Tasking compiler uses pragmas for several purposes,

among them the inclusion of in-line assembler. I like the feature, especially for including short assembler routines that speed specific functions.

THE LIBRARY

The all-important library is another area where BSO offers a product with a great deal of potential. Among the 80 different functions provided, there are enough memory and I/O options to ease a programmer's life considerably. In addition, when you buy the BSO/Tasking package, you get complete source code to all library functions. There are simply too many potential system variables and too many situations where timing is critical to let someone else make all of the library code decisions for you.

First, there are standard dynamic memory allocation functions like malloc(), calloc(), and free (). The standard library includes functions to deal with a number of different memory types (xdat, idat, RAM,

etc.). All told, the programmer can deal with almost any memory configuration the system designer has thrown at him, and use either static or dynamic memory models for various tasks. So far, so good.

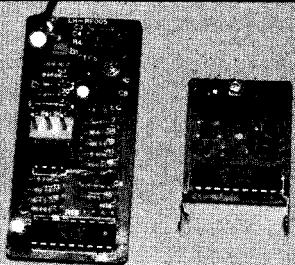
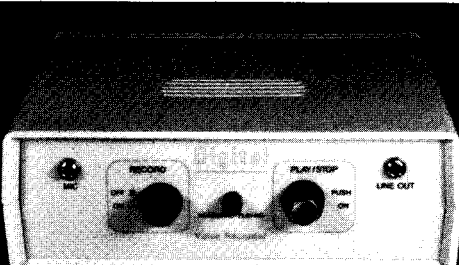
When you need to do standard, low-hassle data I/O, you'd like to be able to call on the basic functions C libraries provide: scanf() and printf(). Unfortunately, in many libraries these functions mean that a large chunk of someone else's code is added to your program. The Tasking library provides the functions, but adds a nice touch-a choice of size. Both functions come in three versions, where size varies according to flexibility of allowed I/O. In short, if you need the maximum data flexibility, they're betting that you'll be willing to live with a larger amount of "built-in overhead." It's a good bet, and one that will make both of these functions of considerable use to programmers. An important side effect is that you can use these standard functions in

more situations, increasing the portability of your code. I wish that I could tell you about the space savings available, but I've complained about the copy protection enough.. .

FEELINGS

The Tasking compiler has the "look and feel" of a high-quality, professional product. The documentation is reasonably good, with an index for each manual. The assembler has been around longer than the C compiler, and it shows in the refinement of the documentation. The index for the assembler is more comprehensive, and the assembler manual presents information a bit more comprehensively.

While I'm talking about the documentation, I have a minor bone to pick with most software houses: It seems to me that, if a customer is paying anywhere from several hundred to several thousand dollars for a software package, the software vendor

<p>REMOTE CONTROL MODULE</p> 	<p>ELECTRONICS 1 2 3 A DIVISION OF MING E&P. INC.</p> <p>1. Exclusive items at good prices. 2. Unique items at better prices. 3. Popular items at the best prices.</p> <p>I - 800- 669- 4406 (ORDER DESK ONLY)</p> <p>1. Prices are subject to change without notice. 2. VISA, MASTER CARD and COD(add \$3) accepted. 3. Freight charge adds \$5 UPS Ground, \$8 UPS Blue, \$15 UPS Red.</p> <p>977 S. Meridian Ave. Alhambra, CA 91803 TEL: (818) 281-4065 FAX: (818) 576-8748</p>	<p>DIGITAL VOICE RECORDER</p> 
<p>TX-88 \$9.95 * Tiny size, only 1" X 1.5", powered by 12VDC. * 4096 possible codes. 100-200' line-of-sight distance</p> <p>RE-01 \$19.95 * Small size, 1.25" X 2.75", powered by SVDC. * Single channel RF receiver.</p> <p>RE-99 \$19.95 * Same as RE-01, except it's a multi-channel RF receiver. * 8 bit address codes, 4 bit data codes.</p> <p>RC-01 Combination of TX-88 & RE-01 \$29.95 RC-99 Combination of TX-88 & RE-99 \$29.95 (Above items are board-level products only)</p>	<p>ROM/SRAM DISK CARD</p> <p>RDC-512 (0 KB) \$179.95 RDC-1024 (0 KB) \$199.95 NEW UPDATED VERSION</p>	<p>DVR-120M \$149.95 * 4 Mb DRAM, up to 2 minutes recording time. * 16 variable-length messages each w/ direct trigger terminal. * Very low standby current, only 8mA! * VOX on/off switch. REPEAT mode available. * Built-in back-up system to prevent memory loss. * "ENDLESS" feature for continuous recording. * Completed product w/ dynamic MIC & AC adapter</p> <p>TALKING ALARM TIMED MESSAGE DEVICE VERBAL NOTEPAD GUIDANCE SYSTEM HAM OPERATION TALKING COMPASS TELEPHONE MUSIC-ON-HOLD TALKING BIRD TRAINER VOICE WARNING SYSTEM</p>
<p>INSTANT EPROM ERASER Erase EPROM in only 5 seconds</p> <p>SALE IEE-9088 \$149.95 was \$249.99</p>	<p>* Now boot from SRAM (RDC.512 only). * The best way to prevent "Computer Virus". * Put DOS & app. files in EPROM or SRAM * Vibration, dust and humidity proof.</p>	<p>REMOTE CONTROL SYSTEM</p> <p>BEST SELLING ZEMCO SA-432 Only \$49.95</p>

could pay someone a few bucks an hour to put the manual together. I mean, really—we don't send uncut sheets, a razor blade, and two staples to our readers and call it a magazine. If you're paying big dollars for software, shouldn't you get a ready-to-read manual?

Aside from the minor points about the documentation, the only beef I have with this compiler is the copy protection. If you can live with that, then I have no trouble suggesting that you try this compiler.

BSO/Tasking 8051 C Compiler (PC), Version 1.1—\$1495
 includes compiler, assembler, linker, library source
 purchase includes one year free technical support via "800" number, and all upgrades and updates
BSO
 411 **Waverly** Oaks Road
Waltham, MA 02154-8414
(800) 458-8276

WHAT'S IT ALL ABOUT?

Even if we wanted to tell you which compiler to run out and buy (and we don't want to do that), there isn't really a standout here. Each compiler has strengths and foibles, and each has idiosyncrasies that will suit it or not to your work habits. The most significant differences involve libraries and copy protection—only Franklin doesn't give you library code, and only BSO is copy protected. The more important point is one of similarity.

None of these compilers are quite mature. Each is in a first or second version, and there are clear signs that the compiler designers are still searching for the right approach to the special embedded application market. Many of these issues, which all boil down to having a thorough knowledge of your customer, were sorted out several years ago in the desktop application market. In another three years, I expect to see dramatic improvements in all of these packages.

If you can't wait three years to buy a C compiler, what should you do? The most important piece of advice I can give is to ask *questions*. Ask colleagues or professional contacts whether they are using any of the available C compilers, and what they think about the one they use. Take advantage of BBSs or on-line information systems to ask the same questions of a wider range of professionals. When you contact a compiler vendor, ask specific questions about the features that are important to you.

There are compelling reasons for many programmers to switch to C. The reasons to make sure you choose the right C are equally compelling. ❖

Scott Martin is the president of Integrated Vessel Inc. Tim McDonough is the president of Cottage Resources Inc. Curtis Franklin, Jr. is Editor-in-Chief of Circuit Cellar INK.

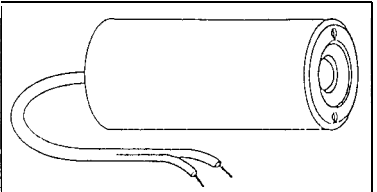
IRS	416 Very Useful
	417 Moderately Useful
	418 Not Useful

LASERS

VISIBLE DIODE MODULE

NEW!

Shown
Actual
Size



Cat. #
LDM-005

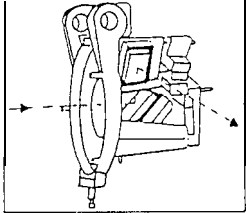
\$19500

New technology in Solid State Lasers makes this Laser Module compact, powerful, visible, and affordable.

Specifications:

Output: 4.5 mW (typical) 670 nm Red
 Spot Size: 1 inch diameter @ 50 feet
 Power Requirements: 5-10 VDC @ 55 mA
 will operate 5 hours on a 9 volt battery
 Dimensions: 1.125" length x .580" diameter
 Weight: .5 ounces (12 grams)


X-Y SCANNER UNIT



Cat. #
LSGAL-2

\$7500

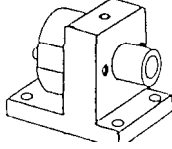
ideal for beam positioning and low speed scanning systems.



SHARP LASER DIODES
 Part # LTO22MC, 780nm, 5 mW output
 Cat. # LD-22 \$15.00

L. D. COLLMATOR

Which also serves as a Holder and Heat Sink. Works great with the above diode. Cat. # LDC-3 \$25.00



In a hurry? Call to ship C.O. D. Add \$15 for S & H. AZ residents add 6.5% tax. FREE CATALOG --- To receive one .. Call, Write or FAX today!

MEREDITH INSTRUMENTS "THE SOURCE FOR LASER SURPLUS"

P. O. Box 1724 • 6403 N. 59th Avenue • Glendale, AZ 85301 • Phone (602) 934-9387 • FAX (602) 939-3369

Ed Nisley

High-Level Languages for Microcontrollers

Don't Believe the Hype

The main justifications for High-Level Languages are that you can produce more lines of code, more bug-free lines, more maintainable lines than you can in assembler during a given number of days. Having used quite a few different HLLs and assemblers on a wide variety of processors, I entirely agree.

An HLL takes care of the grisly details: where are the variables, how to manipulate things bigger than ALU registers, how to manage subroutine parameters and local variables, and so forth. Because humans do those things so poorly (no matter how carefully they work), any decent HLL compiler frees up desperately needed brain-power.

An HLL doesn't help by concealing details: it hurts by suppressing information you need to do the best possible job.

You trade off control over the details to focus on the Big Picture. You assume the compiler is doing a good job on its part and devote your attention to the Rest Of The Story. When this synergism works well, your project will reap those HLL benefits.

But here is a simple HLL quiz: what is the difference between these two 8051 C statements:

```

The C code:

int Variable1;
int Variable2;
int Variable3;

near int Variable4;
near int Variable5;
near int Variable6;

main0 {
Variable1 = Variable2 + Variable3;
Variable4 = Variable5 + Variable6;
}

The generated code:

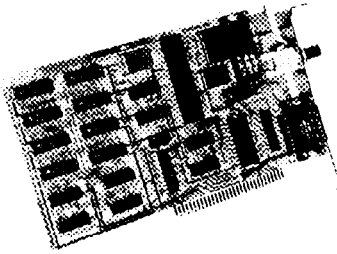
;demo.c51: 13: Variable1 = Variable2 + Variable3;
0003 90 0004 mov dptr,#_Variable3
0006 E0 movx a,@dptr
0007 FA mov r2,a
0008 A3 inc dptr
0009 E0 movx a,@dptr
000A FB mov r3,a
000B 90 0002 mov dptr,#_Variable2
000E E0 movx a,@dptr
000F FC mov r4,a
0010 A3 inc dptr
0011 E0 movx a,@dptr
0012 FD mov r5,a
0013 ED mov a,r5
0014 2B add a,r3
0015 FD mov r5,a
0016 EC mov a,r4
0017 3A addc a,r2
0018 FC mov r4,a
0019 90 0000 mov dptr,#_Variable1
001C EC mov a,r4
001D F0 movx @dptr,a
001E ED mov a,r5
001F A3 inc dptr
0020 F0 movx @dptr,a
;demo.c51: 14: Variable4 = Variable5 + Variable6;
0021 ES 03" mov a,_Variable5+1
0023 25 05" add a,_Variable6+1
0025 F5 01" mov _Variable4+1,a
0027 E5 02" mov a,_Variable5
0029 35 04" addc a,_Variable6
0028 F5 00" mov _Variable4,a
    
```

Variable1 = Variable2
 + Variable3;
 Variable4 = Variable5
 + Variable6;
 Hint: one statement occupies 29 bytes and takes 36 cycles, the other requires 12 bytes and 6 cycles. Which

is which?
 Microcontroller systems (notably the 8051 family and other tiny micros) impose severe constraints on program size, data allocation, and run-time performance. Often the program must fit into 32K and respond to "real-time

étude™

25 MHz 8-bit
ANALOG-TO-DIGITAL CONVERTER



Based on the TRW THC10681 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
- Factory calibrated
- 16 jumper selectable base addresses
- External clock and trigger inputs TTL compatible
- Software source code included

PRICE: \$495⁰⁰

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.

1390 CARLING DRIVE
SUITE 108

ST. PAUL, MN 55108

VOICE MESSAGE OR FAX
(612)645-8088

©1990-91 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 & specifications subject to change.

Reader Service # 192

events" measured in a few microseconds. You cannot afford to trade off code space for execution speed and you must pay attention to the location of every variable in that precious internal CPU RAM.

The difference between the those statements is that one uses External Data Space and other uses Internal Data Space. because C hides the details of variable manipulation from you, you cannot tell which statement is which. I contend that you really ought not lose sight of a two-times increase in code size and a six-times increase in execution time quite that easily!

Good microcontroller code requires detailed knowledge of the whole project: hardware, software, and firmware.

There are ways around the problems, which I have used on several projects. Microsoft's "Hungarian notation" helps to identify the variables:

```
nsVar1= sVar2 + *npsVar3
```

indicates which variable are "near" and "far" although the space and time implications may not be obvious at first glance. And most current C dialects support keywords that give you control over memory allocation; if you are particularly lucky, you may actually be able to use the standard library routines in a mixed-memory-model project.

I contend this wraps up the worst of both worlds in one ungainly package: the memory allocation headaches of assembler with none of its precision and a less readable, more fragile HLL program burdened with internal compiler details.

Even though combining HLL and assembler routines in one program (a.k.a., "tweaking the hot spots") sounds inviting, approach with caution! HLLs make many assumptions about how memory is used; you must contort the assembly code into that mold. You'll spend a lot of time jockeying the Hatfields and McCoys into the same EPROM.

If your project is blessed with ample CPU resources or it is so trivial that any CPU is adequate, there is no justification for assembly language. Pick a good HLL, implement the best algorithms you can find, and have at it. You'll get good results and never miss the details.

However, when you set out to develop nontrivial code with tight timing requirements for a microcontroller with cramped address spaces and an idiosyncratic instruction set, I strongly suggest you gnaw the assembly language bullet from the start. If you do it right, the overall design can be just as clean and the code just as readable and maintainable as an equivalent HLL program. And those tight timing requirements won't be such a big deal because you'll polish them off first rather than patching them later.

Writing good microcontroller code requires detailed knowledge of the whole project: hardware, software, firmware, design requirements, whatever. If you don't keep everything in mind all the time you will screw it up. An HLL doesn't help by concealing details: it hurts by suppressing information you need to do the best possible job. Assembler doesn't help by giving you full control: it **burdens** you with more details.

Pick the right tool for the job and don't believe the hype. ❖

IRS

- 440 Very Useful
- 441 Moderately Useful
- 442 Not Useful

How We Evaluate

It has taken **CIRCUIT CELLAR INK** three years to start doing product **evaluations**. We didn't want to look at products until our readers told us they wanted them. Now that we will evaluate products, we want to make sure that they are as useful as possible to you. Here's what we plan to do:

THE PRODUCTS

We are interested in the products that engineers, system designers, and programmers use to develop computer applications. That means no spreadsheets and no page-layout programs. We will, over time, split our attention between hardware and software tools, and between hardware and software development. We'll also keep our attention focused on the products that you **need to know about**—there may be some **great** CAD packages for IBM 3090s, but if you're in the market for one of them, you certainly don't need our help., .

THE PEOPLE

There now are people who earn a fair portion of their living reviewing products for magazines. They won't be writing any of the evaluations for **CIRCUIT CELLAR INK**. Our evaluators are working professionals who have agreed to look at a product they don't normally use so that they can help you understand its advantages and disadvantages. Their writing style may not be as polished as that of someone who spends 60 hours a week writing PC software reviews, but you'll get a point of view that is relevant to your working world.

THE EVALUATIONS

We want our evaluations to be more than just feature lists. We ask each evaluator to tell us how the product fit into their working routines. Then, the editors don't try to come up with a magic number or Editor's Favorite award. You read the evaluation, you make up your mind.

We won't generally use benchmarks to generate numbers. We may try to run a standard situation through each evaluated product, but we don't think that there's much point in pulling Dhrystone numbers out of 8051 packages,

THE FUTURE

As we said, we want these evalu-

ations to be useful to you. You can help us by doing at least one of three things:

1. **Tell us which products you want to know about.**

We have **certain** ideas based on letters, **conversations**, and BBS messages, but we want to know **which** products have you confused.

2. **Tell us how you want the products evaluated.**

Are short, **concise** evaluations what you need? Long-term reports of workplace compatibility? Lots of color pictures? No evaluations at all? Let us know how to

make these evaluations work for you.

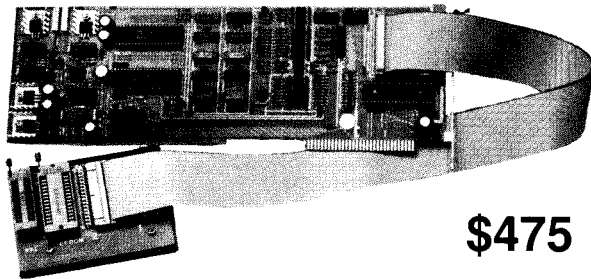
3. **Join the Evaluation Staff**

If you're a working professional and you'd like to write an evaluation for **CIRCUIT CELLAR INK**, send us a letter telling us what you do, which product(s) you'd like to evaluate, and why your views would be valuable to the rest of the readers. Send your letters to:

Circuit Cellar INK
Product Evaluation Staff
4 Park Street
Vernon, CT 06066

UNIVERSAL PROGRAMMER

optional
87C51...
87C751
87C752
87C48
87C49



\$475

PAL
GAL
EPROM
EEPROM
PROM

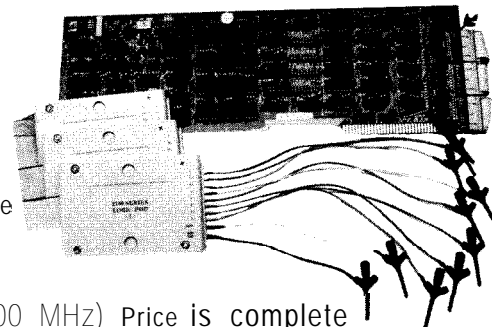
5ns PALs
4 Meg EPROM (8, 16 bit)
26V12 & 18V10 GALs
Parts added at your request.

16V8
22V10
26V12
27040
27220
27011
93xx
XC1 736A
and more

FREE software updates on BBS.
Powerful menu driven software.

200 MHz Logic Analyzer

- 16K samples/channel
- Variable threshold levels
- 3 External Clocks
- 12 Qualify lines
- 200 or 100MHz sampling rate
- 24 Channels, Timing and State
- 16 Levels of Triggering



\$ 799-LA12100 (100 MHz) Price is complete
\$1299-LA27100 (100 MHz) Pods and Software
\$1899-LA27200 (200 MHz) included



Call - (201) 994-6669

Link Computer Graphics, Inc.

4 Sparrow Dr., Livingston, NJ 07039 FAX:994-0730

Ken Davidson

Using High-Level Languages on Embedded Controllers

In Some Cases, It Ain't That Bad

I've recently had what's been called a "religious experience" while using a high-level language on a small micro

So what do you think? Are high-level languages suited for program development on embedded controllers? Before I say what I think, let me give you some background so you know where I'm coming from.

I've been programming for about 14 years now, with most of my experience being with assembly language. I've also done a good amount of work

in BASIC, Pascal, and Modula-2, and have done at least some programming in a lot of languages, on more processors than I can recall.

I also design hardware. My favorite type of programming is to write code that is, as Ed so aptly put it in his very first "Firmware Furnace" column in issue #1 Of *CIRCUIT CELLAR INK*, "lean and mean" and "built to run

right down on the bare metal."

By now you must be thinking, "He can't possibly favor using a high-level language to develop code for an embedded controller. What's the deal here?" Well, I've recently had what's been called a "religious experience" while using a high-level language on a small micro, and I think that, in certain cases, it is well justified.

HIS EYES ARE OPENED

I was recently given the task to develop a set of test procedures to check out the design of the RTC-V25 (see "From the Bench" in issue #17 of *CIRCUIT CELLAR INK*). Jeff had done some rudimentary assembly language routines to check out the hardware during design, but they needed some help to be able to work together as part of a unified test program.

I started doing the necessary revisions and upgrades when a box landed on my desk with a copy of LOCATE and TDREM from Paradigm Systems. (I was right: a steep learning curve), I couldn't believe how quickly the development went. One of the biggest advantages HLLs have over assembly language programming (unless you have a good library in place which, in this case, I didn't) is they make doing a usable user interface much easier. In a matter of hours, I had most of the test program written. For example, the code for dealing with the EEPROM took less than an afternoon to write and debug, while it would have taken several days doing it with assembly language (and since I've already dealt with the same EEPROM from assembly language with the RTC180, I know of what I speak).

Using Turbo C and Turbo Debug-

LEARN TO WRITE AND TEST PROGRAMS FOR THE 8051



The 8051 Microcontroller: Architecture, **Programming and Applications** • by Kenneth **Ayala**

This new text shows you how to program the 8051 microcontroller. Many examples and sample programs are included to help you master the unique instruction set of the 8051. Also included is a disk which contains an assembler and simulator that runs on IBM PCs and compatibles. This disk assembles and allows you to test your programs without having to purchase any additional 8051 hardware. The disk was developed by David Akey of PseudoCorp. You can purchase the text, with disk included, for only \$49.00 + local tax, shipping, and handling.



To order, call or write:
West Publishing Company • Attn: COP Department • P.O. 64833
3773 Hwy. 149 • Eagan, MN 55123
1-800-328-9352 • Visa and Mastercard accepted

Reader Service #204

ger in remote mode, I can write my C source, invoke a MAKE file to compile the source, link it to an executable, and convert it to an Intel hex file (with LOCATE). Then I run Debugger, the code is automatically transferred to the RTC-V25's memory, and I'm presented with a full-screen display of my source code and the assembly code generated by the compiler. I can elect to single step through the C source or through the assembly equivalent, and can even back up through steps that have already been executed. I can select any variable in the C code to watch or inspect, and can look at any memory location or processor register in the system. I can even tell the debugger to start the code running, and stop it any time with a simple press of Ctrl-Break.

WHAT DOES IT ALL MEAN?

OK. Enough preaching. Obviously, the same debugging scenario I outlined above could be used by someone using assembly language rather than a compiled language, so ease of debugging is just icing on the cake. Besides, you get much of the same functionality from an ICE, but at a higher cost.

We all know the traditional advantages of HLLs over assembly language, and I think I've touched on most already. What about the traditional disadvantages of using a HLL on an embedded controller (i.e., memory and speed)?

The RTC-V25 is a different animal from a small controller like the RTC31. The 8031 was designed for low-cost dedicated applications. It supports just 64K of code and 64K of data (assuming they haven't been combined into a single 64K space). When memory is important, you avoid a high-level language. The V25, on the other hand, supports up to one megabyte of memory (384K on the RTC-V25), so memory becomes much less of an issue.

What about speed? In many cases, such as my user interface example, we don't care much about speed, so the HLL becomes very attractive. Any

part of the code where speed is important can be recoded in assembly language and linked in after everything else has been compiled.

I'm not saying HLLs are for everyone and every embedded controller. Far from it. What I am saying, though, is that in cases such as the RTC-V25, they are a viable and desirable alternative. The V25 is well suited to a compiled language and there is a vast sea of development software on the market that is compatible with it. There are numerous languages from

which to choose and libraries such as multitasking executives that make code development easier abound. While smaller (and cheaper) processors have some high-level tools available, their numbers are far smaller, so assembly language often is the only good alternative. ❖

IRS

- 419 Very Useful
- 420 Moderately Useful
- 421 Not Useful

Cross-Assemblers from \$50.00

Simulators from \$100.00

Cross-Disassemblers from \$100.00

Developer Packages

from \$200.00(a \$50.00 Savings)

Make Programming Easy

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

Get It Debugged--FAST

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

Recover Lost Source!

Our line of disassemblers can help you re-create the original assembly language source.

Thousands Of Satisfied Customers Worldwide

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

Processors

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096,196kc
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog 280	NSC 800
Hitachi HD64180	Mot. 68k,8,10		

New

Zilog Z8 Zilog Super 8

. All products require an IBM PC or compatible.

For Information Or To Order Call:

PseudoCorp

716 Thimble Shoals Blvd, Suite E
Newport News, VA 23606

(804) 873-1947 FAX:(804)873-2154

DEPARTMENTS



page 74

Firmware Furnace



page 82

From the Bench



page 93

Silicon Update



page 100

Practical Algorithms



page 103

Domestic Automation



page 105

ConnectTime

It's Just You and The CPU

Intel 80x86 Instruction Timings

The firmware equivalent of a tree falling over in the woods with nobody around is a program with no output. Executive credenza warmers aside, all computers produce output signals of one sort or another. Firmware in embedded systems also has timing restrictions, so it must deliver the right answer at exactly the right time.

In the limiting case there may be only a few instructions between output events. Benchmarking this type of performance leaves your programming skill nowhere to hide, because the results don't depend on algorithms, compiler optimizations, or other trickery. It's just you and the CPU, alone together...

In several earlier columns I described ways to measure your code's performance and tune for best results. Many examples ran on 8031 systems, simply because 8031 CPUs appear in many embedded systems. However, more complex systems are using CPUs from the Intel 80x86 series; while the box may not be a PC clone, the CPU executes the same old PC instructions.

If you are accustomed to the 8051 architecture, you're in for a shock when you write embedded controller code for an 80x86 CPU. (The converse is also true, but that is the topic for another column.) Apart from the instruction set differences, you can no longer measure time intervals by counting instruction cycles.

The Microsoft MASM reference manual includes cycle times for all the

instructions, but sports the disclaimer "The clock counts are for best-case timings. Actual timings vary depending on wait states, alignment of the instruction, the status of the prefetch queue, and other factors." Does this sound like something you should know more about?

Thought so...

THE SETUP

You are familiar with standard benchmarking. Typically, you run a subroutine many times to get a meaningful time interval, then divide the total elapsed time by the number of iterations to get the average execution time. This works well for application programs, because the average time is a meaningful number.

FIRMWARE FURNACE

Ed Nisley

That technique doesn't work at all when you measure a very short instruction sequence, because the loop overhead is much larger than the test code. Worse, because instruction timings are influenced by the **surrounding code**, you may get dependable, repeatable, hard numbers that are also completely wrong. The only valid way to examine your code sequences is in the context of your own program.

Regardless of what your code actually does, output boils down to an **OUT instruction** that sends a byte or word to a device. The data may write a character on the screen, twitch a stepper motor, snap a relay, or start a counter, but as far as the CPU is concerned it's all the same opcode. There are actually three different OUT instructions, but nearly all programs use the OUT DX, AL form which I'll use in this column.

Because all desktop PC systems have a printer port, I'll define a "useful output" as toggling a printer port bit on and off. In your embedded hardware, you can use chip selects, write pulses, or whatever is critical to your application. Whatever the output, measuring instruction timing is thus a matter of connecting an oscilloscope to those signals to get a visible indication of what your code is up to.

Figure 1 shows the measurement setup. Although I used a Tektronix 2445 oscilloscope with on-screen timing readouts to get nice pictures for this column, you certainly don't need

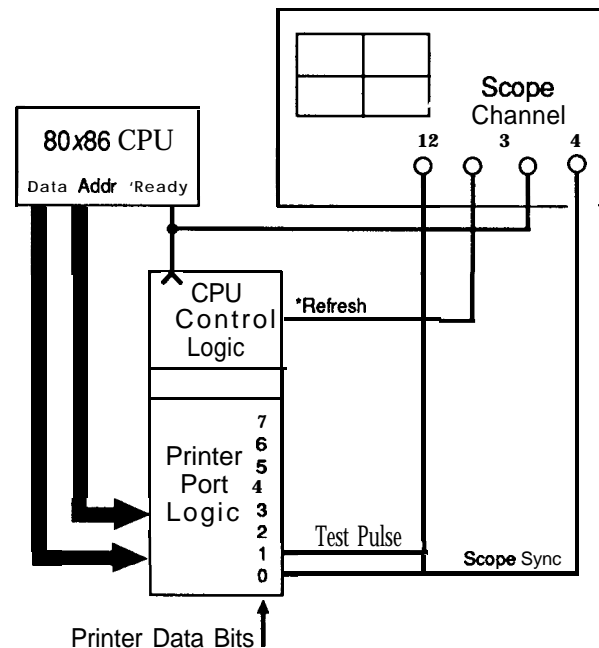


Figure 1 -The measurement setup includes a connection to the computer's 'REFRESH signal plus some control lines from the printer port.

that much firepower for your investigations. However, your scope must be able to distinguish time intervals to an accuracy of one clock cycle, which is about 200 ns for an old 4.77-MHz PC and 100 ns for ATs. I used a 10-MHz Micromint OEM286 because the clock cycles work out to a nice, round, 100 ns each.

The software uses Microsoft C 6.00 and MASM 5.10a, but should be portable to Turbo C and Turbo Assembler without too much effort if you favor those dialects. The code seeks out the highest-numbered printer port in your system, which seemed a reasonable default. The full source code won't fit here, but it is, as always, available on the Circuit Cellar BBS. [Editor's Note: Software for this article is available on the

Circuit Cellar BBS and on Software on Disk #19. See page 107 for download and ordering information. 1

THE BASICS

The `Pulse1` subroutine in Listing 1 forms the basis of all the other code for this column. A C driver program handles the user interface (such as it is) and a C function written in assembly language starts the appropriate test routine. Both are fairly obvious and aren't relevant to this discussion.

`Pulse1` uses three OUT instructions: The first triggers the scope by raising bit 0 (a.k.a. TRIGGERBIT), while the second and third OUTS generate the actual output pulse on bit 1 (a.k.a. PULSEBIT). If you look closely at your own code, you'll find a similar section that you can investigate the same way.

In most applications that require output pulses you must change only a single bit in the current output value. `Pulse1` loads a binary zero in register AL, turns on TRIGGERBIT using a logical OR instruction and writes it out, then turns on PULSEBIT and writes that value out, and then clears both bits with a logical AND instruction and writes the (now zero) value. I could use three constant values, but that would simplify the situation too much.

The 'Punt macro' appearing after each OUT instruction is shown in Listing 2 (along with a number of other useful macros). The IBM PC/AT docu-

```

Pulse1   PROC    NEAR
Restart: MOV     DX,PortAddr      ; set up output port
          MOV     AL,0             ; start with zero bits
          OR      AL,TRIGGERBIT    ; scope sync
          OUT     DX,AL
          Punt
          OR      AL,PULSEBIT      ; output high
          OUT     DX,AL
          Punt
          AND     AL,NOT (TRIGGERBIT OR PULSEBIT) ; outputs low
          OUT     DX,AL
          Punt

          WaitTimer LOOPDELAY      ; enforce delay
          TestKeyPress             ; cancel?
          JZ      Restart

          MOV     AX,0             ; set return code
          RET
Pulse1   ENDP

```

Listing 1—The fundamental timing loop. This routine creates two output pulses on a printer port. The oscilloscope trace triggers on the first pulse to provide a stable view of the second.

```

Punt     MACRO
LOCAL    NextInst
JMP      SHORT NextInst

NextInst: ENDM

WaitTimer MACRO    Delay
LOCAL            Retest,Restart
PUSH            AX
PUSH            BX
          CLI
Restart: GetTimer TIMER0
          MOV     BX,AX
          SUB     BX,Delay
          CMP     BX,AX
          JA      Restart
Retest:  GetTimer TIMER0
          CMP     BX,AX
          JB      Retest
          STI
          POP     BX
          POP     AX
          ENDM

GetTimer  MACRO    TimerAddr
MOV       AL,TimerAddr ; set up latch command
AND       AL,03h
OR        AL,TLATCH
ROR       AL,1
ROR       AL,1
          OUT     TimerAddr,AL ; latch count
Punt
IN        AL,TimerAddr ; get LSB of timer value
Punt
MOV       AH,AL
IN        AL,TimerAddr ; get MSB of timer value
Punt
XCHG     AH,AL ; MSB to AH, LSB to AL
          ENDM

SHIFTKEYS EQU    03h ; mask for shift key state

TestKeyPress MACRO
TEST     ES:KeyCtrl1,SHIFTKEYS
          ENDM

```

Listing 2—These useful macros are used in all the display routines. Punt flushes the instruction prefetch queue after each I/O operation. It is required on some AT-class machines, but may not be needed on others. Wait Timer pauses for a specified delay interval measured by Timer 0. Get Timer reads Timer 0 “on the fly” by latching the current count. Finally, TestKeyPress tests the BIOS keyboard shift state variable and returns the zero flag clear if either shift key is pressed. The ES register points to the BIOS data area.

mentation includes the warning that you must put a “null jump” between each pair of I/O instructions to allow the I/O devices time to recover. My experience is that many machines will work fine without this precaution, but others go nuts. If you are writing for a particular hardware configuration, spend some time testing this to see if the hardware really needs it.

The WaitTimer macro delays for about 1 millisecond (set by the LOOPDELAY constant) by reading Timer 0; this produces a reasonable scope refresh rate and introduces a slight randomization because the 8254 and the CPU are driven by different crystals. The GetTimer macro latches the current count and reads the two bytes into AX. Remember that Timer 0 runs in Mode 3, so it decrements by two counts every 840 ns.

TestKeyPress is a one-line macro that checks the BIOS keyboard data area to see if either shift key is pressed. This allows the loop to run until you press a shift key, but doesn’t involve the overhead of DOS or BIOS functions. In many cases you need a positive way out of a loop, but can’t afford lots of time for a full keyboard test. In this code, of course, that overhead wouldn’t make much difference, but the trick is handy to know.

With all that in mind, Photo 1 shows Pulse1 in action. The top trace is the pulse generated by the OUT instructions, and you can see at least three copies of the pulse: the brightest starts at 2.1 microseconds, but there are others at 2.9 and 3.1 μ s. All three are 1.9 μ s long, so what’s going on?

THE REFRESH THAT PAUSES

If you have any experience with this sort of thing, your instinct shouts “Gotcha! He didn’t shut off the interrupts!” This indicates that your instinct can get you into a lot of trouble. The starting times vary by only a few hundred nanoseconds, which is too short for an interrupt.

True, interrupts are enabled and occasionally the pulse is stretched by a timer tick. On the scale of Photo 1, the BIOS timer interrupt handler produces a pulse about five yards long. If

you do this experiment on your own system, you may be able to see a very dim trace scoot off to the right every now and again.

You are watching dynamic RAM refresh in action!

Everybody's desktop PC uses DRAM, except those old Dell 286 machines with static RAM to squeeze the last itsy from a no-wait-state design. Everyone knows that DRAM must be refreshed once in a while to keep it from forgetting its contents. Unless you're reminded occasionally, it's easy to forget that DRAM refresh occurs every 15.1 microseconds. Right?

The middle trace in Photo 1 is the *REFRESH signal from the I/O backplane. The pulses are not synchronized with the test loop, so the refresh cycles don't show up as single pulses. However, there are several refresh pulses corresponding to each pulse on the top trace.

The bottom trace shows the *READY signal marking the completion of each bus I/O cycle. It is rather fuzzy because the bus cycles are perturbed by the refresh cycles; obviously the instruction timings aren't constant.

If your embedded controller design calls for precise pulses, you cannot afford to forget refresh. In fact, one of the advantages of a static RAM design is that there aren't any refresh cycles around to confuse your timing. The Micromint RTC-V25 (and similar boards) use static RAM; this may be a compelling reason not to use a standard XT or AT system board in your project!

But if you're stuck with DRAM, what's one to do? Simple: turn off the refresh when you need a precise time interval.

Photo 2 displays the result. The output pulse on the top trace is very stable, and the refresh pulses are con-

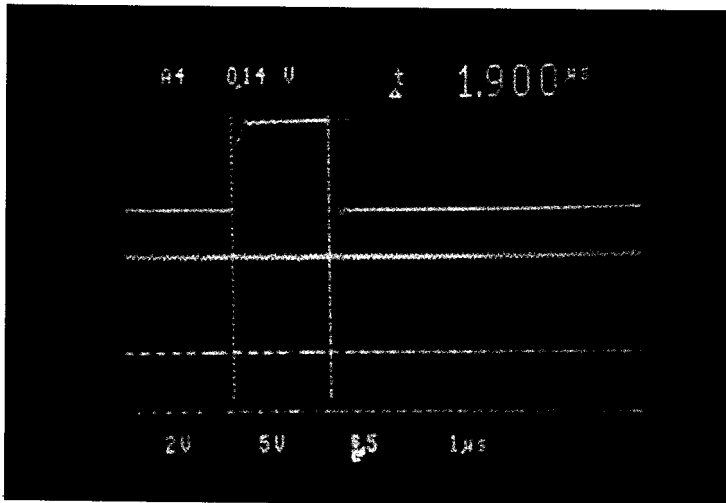


Photo 1—
Dynamic RAM refresh is responsible for the 'ghosting' effects in the top and bottom traces.

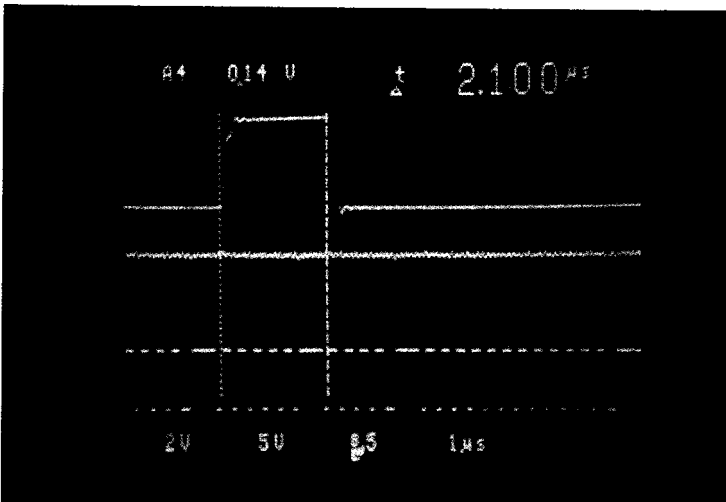


Photo 2—
Elimination of the refresh signal results in rock-solid traces in both cases.

A Message to Subscribers

Circuit Cellar INK, The Computer Applications Journal occasionally allows companies which have products or services of interest to our readers to conduct a mailing to our subscriber list.

If you do not wish to receive information from these companies, your name can be removed from the list which we supply to them.

To have your name removed from the outside mailing list, send a written request to:

**Circuit Cellar INK
Subscriber Service
P. O. Box 3050-C
Southeastern, PA
19398**

```

Pulse3   PROC    NEAR

          MOV     DX,PortAddr    ; set up output port

Restart:  CLL                     ; eliminate distractions
          RefreshOff              ; turn off refresh

          MOV     AL,0            ; start with zero bits
          OR      AL,TRIGGERBIT   ; scope sync
          OUT    DX,AL
          Punt
          OR      AL,PULSEBIT     ; output high
          OUT    DX,AL
          Punt
          AND    AL,NOT (TRIGGERBIT OR PULSEBIT) ; outputs low
          OUT    DX,AL
          Punt

          RefreshOnTFASTREF      ; use fast refreshing
          STI

          WaitTimer LOOPDELAY     ; enforce delay
          TestKeyPress           ; cancel?
          JZ     Restart

          RefreshOn TNORMREF     ; restore normal refreshing
          MOV    AX,0            ; set return code
          RET

Pulse3   ENDP

```

listing J-Disabling refresh. This routine is similar to Listing 1, but turns off DRAM refresh during the critical pulse interval.

```

TIMER0   EQU    40h            ; TOD interrupt timer
TIMER1   EQU    41h            ; RAM refresh timer
TIMERCTL EQU    43h            ; control port

TREFRESH EQU    01010100B     ; mode 2 LSB load only
TFASTREF EQU    10            ; high-speed refresh
TNORMREF EQU    18            ; normal refresh

;-----
; Disable RAM refresh

RefreshOff MACRO

          MOV     AL,TREFRESH   ; shut off refresh
          OUT    TIMERCTL,AL
          Punt

          ENDM

;-----
; Enable RAM refresh
; Write timing value to Timer 1

RefreshOn  MACRO  Interval

          MOV     AL,Interval   ; set new interval
          OUT    TIMER1,AL
          Punt

          ENDM

```

listing I-DRAM refresh control macros. These macros disable and enable DRAM refreshing. Because refreshing should not be shut off for very long, the functions cannot be implemented as subroutines.

spicuous by their absence on the middle trace. The *READY pulses are now synchronous with the output pulse because the instructions are not disturbed by refreshing.

Listing 3 is the test loop that generated Photo 2, while Listing 4 shows

the macros that control DRAM refresh. The code steps up the refresh rate to about twice the normal rate to ensure that all DRAM locations are hit often enough to compensate for the missing pulses. You must disable interrupts while refresh is off, because

you cannot afford to allow anyone else to get control of the CPU and waste precious time on another function.

The typical DRAM refresh spec is “every location every four milliseconds” at the usual temperatures. Refresh cycles use only the low-order eight address bits, so 256 cycles every four milliseconds averages out to one refresh every 15 microseconds. However, as long as you refresh every location you can bunch the refreshes up any way you like.

Incidentally, tight loops like these will continue to work correctly even with refreshing completely disabled, as I discovered by a simple goof (pronounced “experiment”). A refresh cycle is basically just a DRAM read; the test loops are tight enough that all the instructions and data are read often enough to keep the DRAM locations valid. When the loop terminates and the CPU fetches an instruction from a location that hasn’t been refreshed for a minute or so.. .

If you use an IBM PS/2 machine (or, presumably, any other Micro Channel clone) this code will not work correctly because there is no Timer 1 controlling the refresh. After poring over the PS/2 technical reference manuals for a while, I can’t find any way to shut off DRAM refresh. While not a tragedy, it does point out one more little difference between the old AT systems and the newer PS/2 models.

UNCOUNTABLE CYCLES

With interrupts disabled and DRAM refreshing shut off, the output pulse is stable enough to measure instruction times. For an introduction to how tricky this is, compare the pulse widths in Photos 1 and 2. With refresh turned on the pulse is 1.9 microseconds long, with refresh off it is 2.1 microseconds. The CPU runs at 10 MHz, so the pulses are 19 and 21 cycles long, respectively.

Table 1 includes the cycle counts for some of the more useful instructions, but does not include wait states. According to the IBM AT Tech Ref, the 8-bit I/O instructions used in these

Instruction		8088/Word	80286	80386
OUT	DX,AL	12	3	11
OUT	DX,AX	12	3	11
IN	AL,DX	8/12	5	13
IN	AX,DX	8/12	5	13
MOV	DX,imm	4	2	2
MOV	AL,imm	4	2	2
OR	AL,imm	4	3	2
AND	AL,imm	4	3	2
ROL	reg,CL	8+4n	5+n	3
REP	MOVS	9+17n/25n	5+4n	8+4n
REP	STOS	9+10n/14n	4+3n	5+5n
NOP		3	3	3
LOOP	short	17	8+b	11+c
JMP	short	15	7+b	7+c
JMP	far	15	11+b	12+c
JMP	reg	11	7+b	7+c
CALL	near	23	7+b	7+c
RETN		20	11+b	10+c
CALL	far	36	13+b	17+c
RETF		34	15+b	18+c
INT	imm	71	23+b	37
PUSH	reg	15	3	2
PUSH	segreg	14	3	2

Notes:

When 8088 timings for byte and word operands differ, the word timing appears after the slash.

"n" indicates a count value

"b" is the number of bytes in the target instruction

"c" is the number of components in the target instruction

Table 1—

Intel 80x86 instruction cycle counts. These are adapted from the Intel manuals and the Microsoft Assembler Reference manual. They are "best case" timings for real-mode execution and do not include wait states, prefetching delays, or other effects. Code timings in the article assume a 10-MHz 80286 CPU.

test loops require six cycles. Fortunately, the five-cycle refresh operations specified in the manual don't affect our pulses.

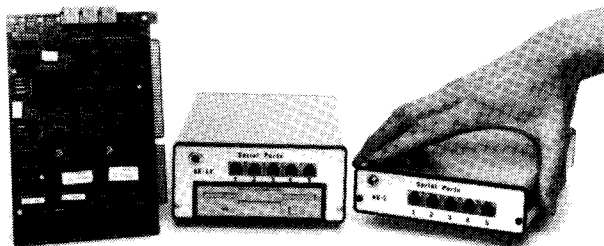
Adding up the cycle times for all three instructions, the loops for Pulse1 and Pulse2 should take 18 cycles. Obviously, something is missing...

Photo 3 adds to the mystery. The code is identical to Pulse2 with a single NOP added between the second two OUT instructions. According to the table a NOP is three cycles long, which should add 300 nanoseconds to the pulse. In reality, the pulse is 22 cycles long, only 100 ns longer.

Photo 4 includes nine more NOPs, so the pulse should be 48 cycles long. Wonder of wonders, it is actually spot on.

Give up? What you are seeing now is the CPU's instruction prefetch queue in operation. The amount of time an instruction takes depends whether it must be fetched from memory or is already in the CPU's queue. In the latter case, an instruction can be executed in a single clock

Get on the PC BUS



ROM or Disk based AT Systems Cards \$299, Systems \$399

It's easy to run your compatible applications on our single board computer! Develop code on a PC, and follow our ten easy steps to place your .exe files and DOS in ROM.

CPU Card: V50 CPU, 8086 Code Compatible
1 MB Ram, 256kB Rom, 4.5" x 7"
5 Serial Ports, , CMOS (2 watt)

Expansion: Backplanes for PC/AT cards
Piggyback card with: Floppy,
SCSI, Printer, and Keyboard ports

Software: BIOS, Utilities, Monitor, & Source code



303-444-7737 fax 303-786-9983
655 Hawthorn Ave., Boulder CO 80304

Reader Service #160

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

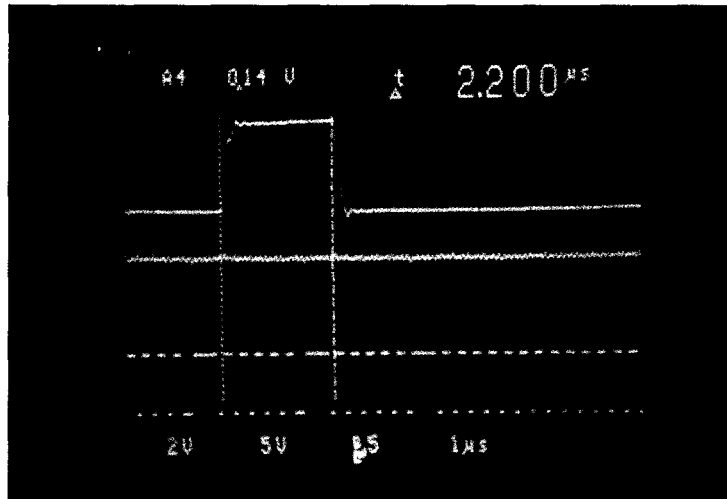


Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

Reader Service #161

Replacing that `NOP` with a `Punt` macro, which is a jump to the next sequential location, takes an additional 13 cycles rather than the 9 you'd deduce from the table. The jump (and call and return) instruction flushes the prefetch queue, so the CPU must re-fetch the next instruction. That's why IBM specifies a `JMP` instead of a bunch of `NOPS`: it can't be prefetched out of existence. If you can guarantee about a dozen cycles between I/O instructions, you don't need a `JMP`.

Photo 3— The addition of a `NOP` instruction should add more time to the sample program's execution than the 100ns shown here.



SNAKES AND LADDERS

What all this boils down to is that you cannot just count up the cycles and expect to get anything more than an estimate of the right answer. What I've found is that you must write the tightest code you can, then tweak it to get the result you want. The cycle counts are a good starting point, but this is truly programming as an experimental science.

One handy trick is to use the bit rotate instructions as a programmable

delay generator. Table 1 shows that they have a five-cycle setup time, plus one cycle per shift. The 8088/8086 CPUs will happily rotate hundreds of bits, but the more recent versions stop after 31, on the assumption that you really didn't know what you were doing.

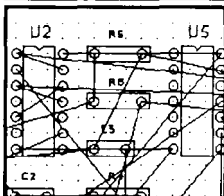
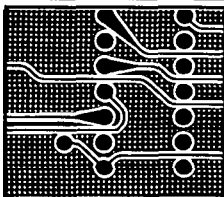
Inserting a single `ROL BX, CL` instruction into the test code produces a 24-cycle pulse when `CL` equals zero.

Each `CL` count adds a dependable 100 ns to the pulse width, with no worries about prefetching or other confusion.

When your code fetches a data value, you must ensure the data is in the right spot. For example, fetching a 16-bit value from an odd address requires three cycles more than fetching it from an even address. Many high-level language compilers force variables to the right alignment, but you

PROFESSIONAL CIRCUIT DESIGN

QICAD



Save time and money!

QICAD is a full-featured printed circuit layout package that gives you everything you need to design circuit boards quickly.

- ON-LINE HELP
- AUTOROUTER
- POWERFUL EDITING
- HPGL/DMPL PLOTS
- GERBER
- POSTSCRIPT
- EXCELLON (DRILL)
- EGA / VGA compatible

\$195.00 complete price

gav

345 W. Williams Avenue
Fallon, NV 89406
(702) 423-1653 (702) 423-1654 FAX

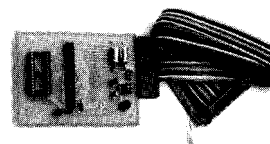
TIRED OF WAITING FOR THE PROMPT ?

Speed up with a ROM DRIVE! Boots DOS and programs instantly. Also used to replace mechanical drive completely in controllers or diskless workstations. The only perfect protection from viruses. Easy to install half-size card.

MVDISK1 64k.....\$95
MVDISK2 360k.....200
MVDISK3 1.44m...300

Quantity discounts!

DOS IN ROM!



\$95 EPROM PROGRAMMER

-PLUG IN ROMS FOR PC
-DO SBC MVDISK
-PROGRAM 2764/27010 EPROMS

WORLDS SMALLEST PC !!!
ROBOTS ALARMS RECORDERS DOS

THREE EASY STEPS:
1. Develop on PC
2. Download to SBC
3. Burn into EPROM

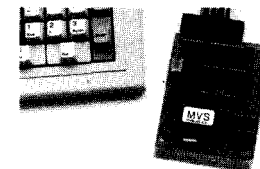
\$95
-2 PARALLEL -LCD INTERFACE
-3 SERIAL -KEYBOARD INPUT
-PC TYPE BUS -REAL TIME CLK
-BIOS OPTION -BATTERY OR SV

FREE SHIPPING IN U.S.

5 YEAR LIMITED WARRANTY

.MVS Box 994
Merrimack, NH
(508) 792 9507

8088 SINGLE BOARD COMPUTER



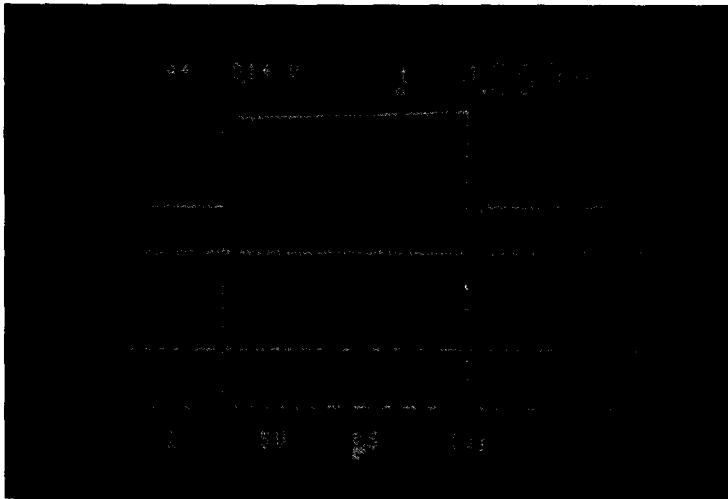


Photo 4— Adding nine more NOPs to the sample code results in the expected increase in execution time.

But if **your** hardware budget won't stand a timer, you might just be able to pull the fat out of the fire with a little tricky firmware. You never can tell. .

The downloadable code for this column should get you started in your own investigations and includes several test cases I didn't have room for here. I'm interested in hearing from anyone running the code on an Intel '486 CPU; the effect of the instruction cache might optimize those null Jumps right out of existence.

Dial up the BBS and tell us how your machine ticks along.

Happy counting! ✚

may have to manually tweak C structures and Pascal records if you are doing something obscure.

Whatever you do, you must actually measure the resulting instruction times to make sure that something unforeseen hasn't occurred. And, of course, if your code will run on several different CPUs, you will need to hand-code different sequences for each possibility.

THE LAST PULSE

As Jeff puts it, "If you need micro-second pulses, put in a real hardware timer and skip all that baloney!" He's right, of course. The best way to get a precision pulse is from a digital timer driven by a clock, perhaps triggered by an incoming event. The firmware need only set the pulse duration, stand back, and wait for the results.

Ed Nisley is a Registered Professional Engineer and a member of the Circuit Cellar INK engineering staff and enjoys making gizmos do strange and wondrous things.

IRS

- 422 Very Useful
- 423 Moderately Useful
- 424 Not Useful

IMAGE PROCESSING & VIDEO IMAGE EDITING

VIDEX

VIDEO EDITING

Combine & manipulate video images:

- add + sub (+/* e 1 n)
- overlay pictures
- zoom in and out
- rotate and slide
- smooth or sharpen
- color enhance, etc.



IMPACT

IMAGE PROCESSING

- math functions
- Boolean operations
- zoom, rotate, translate
- threshold, edge enhance
- editable mask filters
- histogram-equalization
- 2-D Fourier transforms
- convolutions... etc.



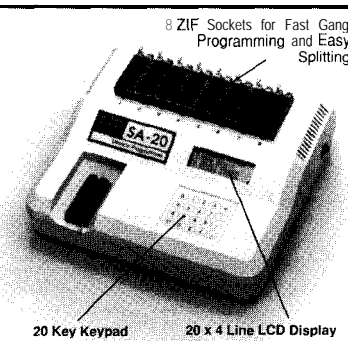
VIDEX and IMPACT redesigned to run on any PC/AT with 640K, DOS 2.2 or later and an EGA or VGA Monitor. Both operate under an integrated windows environment with pull down menus, hot keys, windows editing, full system file support and on line documentation. Get VIDEX for \$25.00 and IMPACT for \$99.95. Please add \$3.50 for stripping and handling.

TARDIS SYSTEMS
945 San Ildefonso, Suite 15
Los Alamos, NM 87544. (505) 662-9401

EPROM PROGRAMMERS

Stand-Alone Gang Programmer

\$750⁰⁰



20 Key Keypad 20 x 4 Line LCD Display

- Completely stand-alone or PC-driven
- Programs E(P)ROMs
- 1 Megabit of DRAM
- User upgradable to 32 Megabit
- 3/8" ZIF Sockets RS-232, Parallel In and Out
- 32K internal Flash EEPROM for easy firmware upgrades
- Quick Pulse Algorithm (27256 in 5 sec. 1 Megabit in 17 sec.1)
- 2 year warranty
- Made in the U S A
- Technical support by phone
- Complete manual and schematic
- Single Socket Programmer also available. \$550.00
- Split and Shuffle 16 & 32 bit
- 100 User Definable Macros, 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex. and Motorola S
- 2716 to 4 Megabit

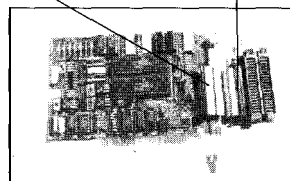
Internal Programmer for PC

\$139⁹⁵

New Intelligent Averaging Algorithm Programs 64A in 10 sec 256 in 1 min 1 Meg (27010, 011) in 2 min 45 sec 2 Meg (27C2001) in 5 min Internal card with external 40 pin ZIF

- Reads Verifies and programs 2716 32 32A 64, 64A, 128, 128A 256, 512, 513, 010, 011, 301 27C2001, MCM 68764, 2532, 4 Megabits
- Automatically sets programming voltage
- Load and save buffer to disk
- Binary, Intel Hex. and Motorola S formats
- No personality modules required
- 1 Year warranty
- 10 days money back guarantee
- Adapters available for 8748, 49, 51, 751.52, 55, TMS 7742, 27210, 57C1024, and memory cards
- Made in U S A

2 ft. Cable 40 pin ZIF



EMPDEMO.EXE available BBS (916) 972-8042

NEEDHAM'S ELECTRONICS

4539 Orange Grove Ave -Sacramento, CA 95841
(Monday-Friday 8 am-5 pm PST)

C.O.D.



Call for more information

(916) 924-8037

FAX (916) 972-9960

FROM THE BENCH

Jeff Bachiochi

Multidrop A/D and D/A Network

Using your PC's Printer Port and Four-Conductor Phone Cable

It seems that any time we talk about data acquisition or control around here, it involves some form of microcontroller at the collection or control site to keep track of things. This month, I'm going to take a slightly different tack and use my desktop PC to do the controlling. I'm also going to go one step further and not open the box.

Instead, we'll use the printer port's eight output bits and four of the input bits to act as a simple interface to remote ADC and DAC satellite modules. The interface, which plugs onto the PC's printer port, provides power to all modules and

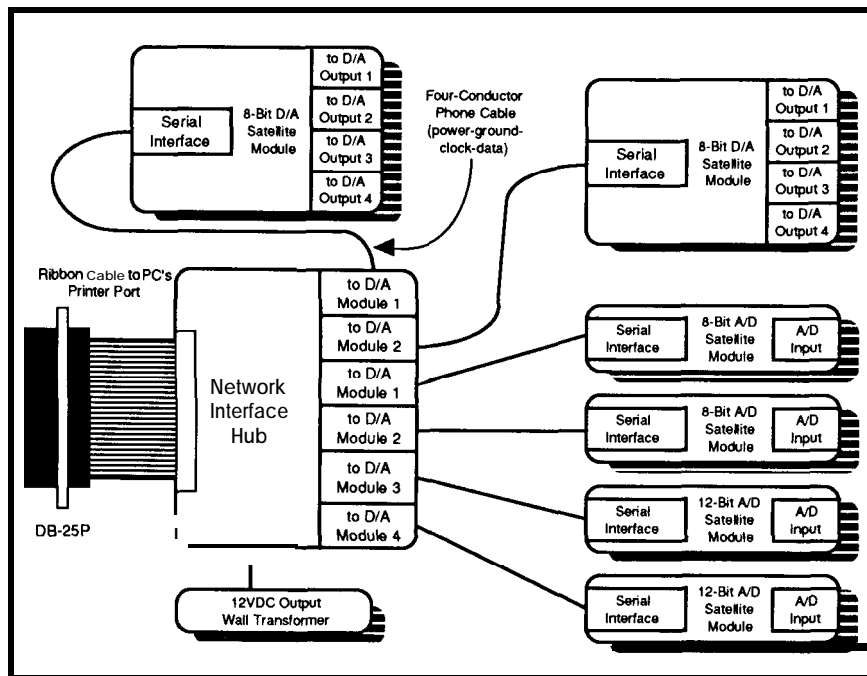
increases drive for all the clock and data lines. Umbilical cords made of standard four-wire telephone cable connect each module to the distribution interface. Each module will use two lines for communication and two lines for power. The ADC modules will use an "output clock" bit and an "input data" bit. The DAC modules require two output bits: clock and data. Individual modules contain all the power regulation, step-up, and inverters necessary for each device. Precision references are included.

Referring to Figure 1, the four most-significant bits of the printer port's data output port drive 7406 open-collector inverters. These bits will be used to clock the data out of each of the four ADC modules. A 1489 level shifter converts data read from each ADC module back into TTL signals connected to four bits of the parallel port's status input register. Pin 11 on the printer port's connector—Printer Busy—is inverted, unlike the other three status inputs I am using. To make all four status bits read the

same (something we could do in software), I used an extra 7406 gate to invert this line to eliminate confusion later.

The two DAC modules require two output bits each. A second 7406 is driven by the lower four bits of the parallel port's output register. Each module has four outputs.

An inexpensive wall transformer power supply (+12 VDC @ 500 mA) has plenty of "oomph" for the whole system. The satellite modules, which need 5 volts, will be powered from this 12 volts as well. This prevents a problem with loss through the cable, even at 1000 feet.



THE PSEUDO-SYNCHRONOUS SERIAL SERVER

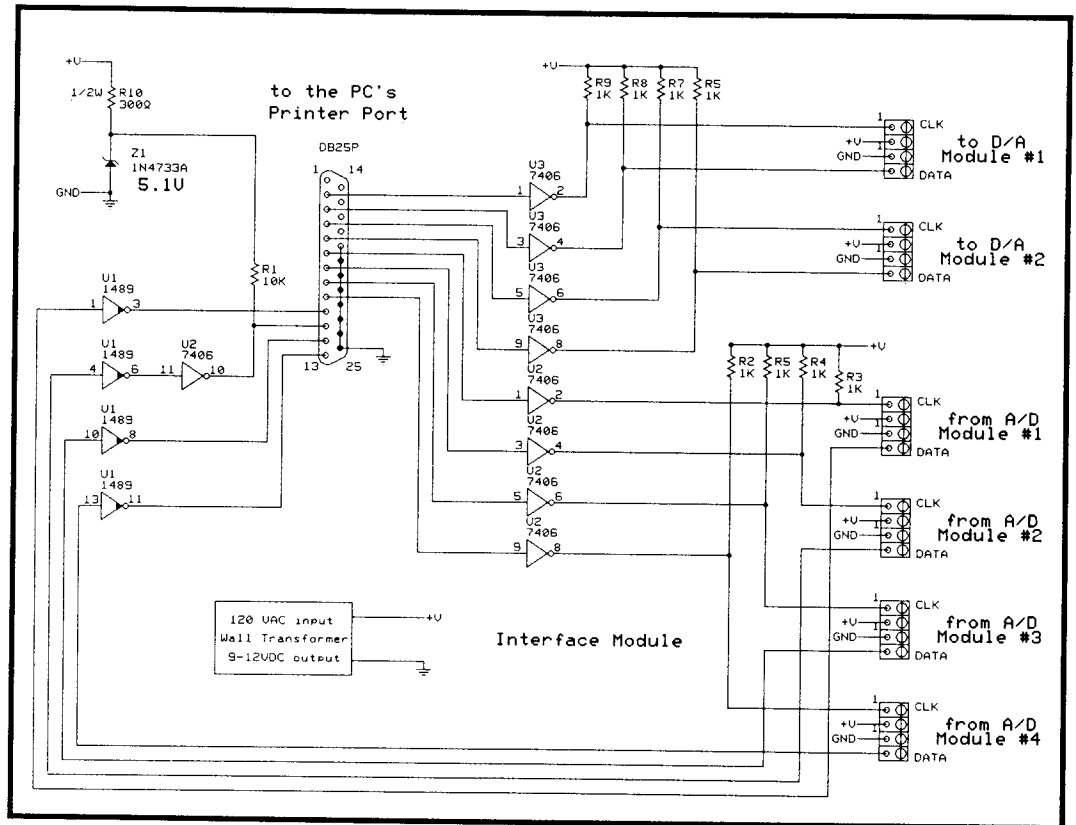
To minimize the hardware, a serial protocol was chosen for this project. The standard UART protocol is inflexible in word length. An alternate approach to the silicon UART is done with software. A software UART requires processing power but is very flexible. The serial format I chose is a synchronous format which

uses two lines. A clock line indicates legal data by the rising or falling of the clock signal, and a data line carries data between devices. With this method, there is no limit to data word length. See Figure 2 for the protocol of each of the devices used.

CONVERTING MEASUREMENT INTO DATA

One of the most widely accepted A/D converters on the market today is the ADC083x series. This family of 8-bit serial converters starts with an 8-pin DIP single-chan-

Figure 1—The PC's printer port makes an unlikely interface to multiple serial devices.



nel unit and ends with 20-pin DIP eight-channel device. Because they are second sourced by a number of manufacturers, product life is assured.

The ADC0831 is perfect for this application. Small in size, differential input, single 5-V supply, and a serial data path make it easy to use as well as functional. The big brothers of the ADC083x family require setup data which complicates the proposed four-wire connection. As it is, the ADC0831 requires a clock input, chip select input, serial data output, reference input, plus power and ground.

The ADC0831 is a serial 8-bit analog-to-digital converter. The ADC0831's clock line is tristated (but pulled high by an external pull-up) in the idle mode. As the idle mode high clock line falls for the first time, the one-shot's *Q output enables *CS, which must go low 350 μ s prior to the clock's rising edge. This is not a problem since the maximum clock rate is 400 kHz (2.5- μ s period). The one-shot is retrigged by each clock cycle for the duration of the transmission. The second falling clock edge initiates a forced "0" on the data line as a start bit, then the 8-bit conversion data bits follow, MSB to LSB, until all have been sent. Reading the data should be performed on the rising edge of each clock cycle to allow for settling of the data. When the clocking ceases, the one-shot finally times out. When *CS returns high, the device tristates the data line.

Since +12 volts (minus any drop in the cable) is provided, a good clean +6 volts can be produced using a LM317. Running the ADC on +6 volts will ensure V_{cc} remains above 4.950 volts, the absolute minimum for a +5-volt input to produce a "1111111" code (see Figure 3). Also, the 5.00-V voltage reference will be double regu-

lated. A trim pot adjusts the reference to 5.00 volts. A second trim pot allows this reference to be lowered as an alternative to the 0-5-volt input range. For instance, using a 5.00-volt reference, 5 volts + 255 steps = 19.6 millivolts per step. If we cranked the reference down to 2.55 volts, each step (between 0 and 2.55 volts) would be 10 mV.

In the case of LM34 (precision Fahrenheit temperature sensor), the output changes 10 mV/ $^{\circ}$ F over a -50 to +300 $^{\circ}$ F range. Disregarding temperatures below 0 $^{\circ}$ F, if the voltage reference is set at 5.00 volts, each step covers 1.96 $^{\circ}$ F for a total coverage of 0 to 300 $^{\circ}$ F (255 \times 1.96 $^{\circ}$ = 499 $^{\circ}$, exceeding the rated range). Change the reference to 2.55 volts and each step covers 1 $^{\circ}$ F for a total coverage of 0 to 255 $^{\circ}$ F.

For the LM35 (-55 to +150 $^{\circ}$ C), and the LM135 (218 to 423 K) precision temperature sensors, a 10-mV step is equal to one degree C or K. That's 1.8 $^{\circ}$ F (1 $^{\circ}$ C) for the LM35, and 1.8 $^{\circ}$ F (1 K) for the LM135. Disregarding temperatures below 0 $^{\circ}$ C, if the reference was adjusted to 1.50 volts, each step would equal 0.59 $^{\circ}$ C (150 + 255 = 0.59). A step size of 0.59 $^{\circ}$ C for an LM35 would convert to a 1.06 $^{\circ}$ F step with a coverage of 32 to 302 $^{\circ}$ F (32 $^{\circ}$ F + [255 \times 1.06 $^{\circ}$ F] = 302 $^{\circ}$ F). Adjusting the reference to 4.23 volts would equal 1.66 K (423 + 255 = 1.66). A step size of 1.66 K for an LM135 would convert to a 3 $^{\circ}$ F step with a coverage of -67 to 302 $^{\circ}$ F (-67 $^{\circ}$ F + [(255 \times (423 K - 218 K)) + 423 K] \times 3 $^{\circ}$ F = 302 $^{\circ}$ F).

The LM34 (Fahrenheit) and LM35 (Celsius) require a negative bias for outputs below 0 $^{\circ}$ F or 0 $^{\circ}$ C. Note that the LM135 does not require this.

You don't get something for nothing. As the reference voltage decreases, the maximum permissible input which produces the "1111111" code also decreases. Therefore

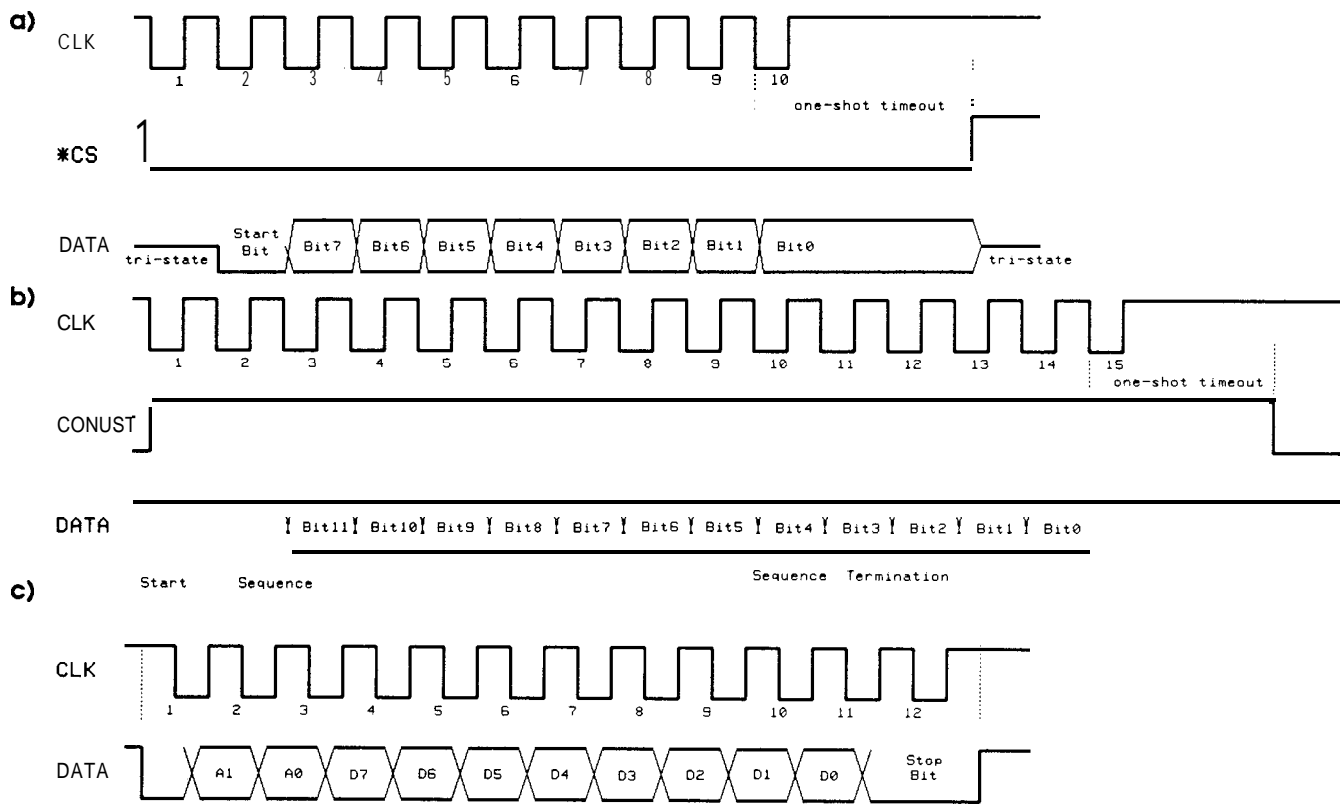


Figure 2—The ADC0831 (a), MAX170 (b), and MAX500 (c) all use a serial interface of one form or another.

68HC11 PROCESSOR BOARD

Offering exceptional value in a single-board embedded controller, Micromint's **RTC-HC11** combines all of the most-asked-for features into a small 3.5" x 4.5" package at a reasonable price. Featuring the popular Motorola **MC68HC118-bit** microcontroller, the **RTC-HC11** gives you up to 21 lines of TTL-compatible I/O; an **8-bit, 8-channel** analog-to-digital converter; two serial ports; a real-time clock/calendar with **battery backup**; **512 bytes** of nonvolatile EEPROM; and up to **64K** of on-board RAM or EPROM, 32K of which can be battery backed.

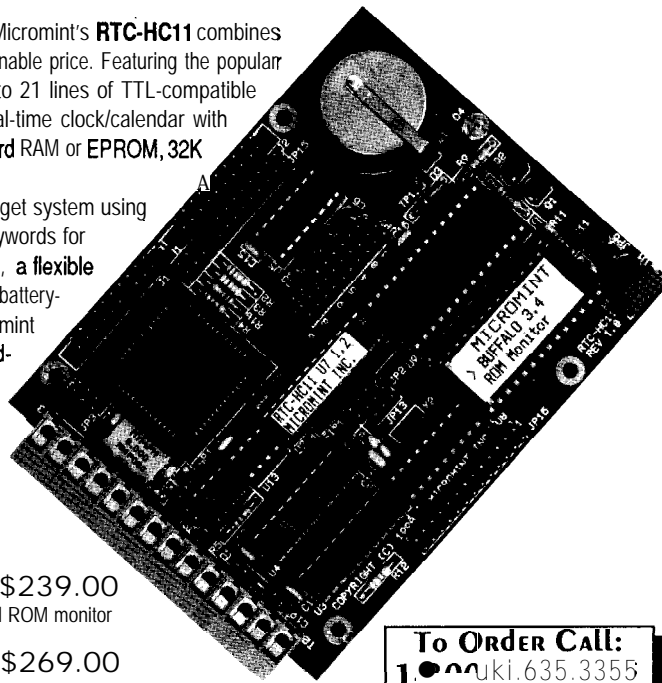
Software development can be done directly on the **RTC-HC11 1** target system using **BASIC-1 1**, an extremely fast integer BASIC interpreter with dedicated keywords for I/O port, A/D converter, timer, interrupt, and EEPROM support. In addition, a flexible configuration system allows a BASIC program to be saved in the on-board, battery-backed static RAM, and then automatically executed on power-up. Micromint also offers several hardware and software options for the **RTC-HC11** including the full line of RTC-series expansion boards as well as an assembler, ROM monitor, and C language cross-compiler.

Additional features include:

- Asynchronous serial port with full-duplex RS-232 and half-duplex RS-485 drivers
- 1-MHz synchronous serial port
- CPU watchdog security
- Low-power "sleep" mode
- 5-volt-only operation
- RTC stacking bus

RTC-HC11-2 \$239.00
 Board w/ 8-bit ADC, EEPROM, 8K RAM, and ROM monitor

RTC-HC11-3 \$269.00
 Board as above w/ battery-backed RAM, clock-calendar, and BASIC-11 in ROM



To Order Call:
 1. 800.635.3355



MICROMINT, INC. 4 Park Street • Vernon, CT 06066 • (203) 871-6170 • Fax: (203) 872-2264

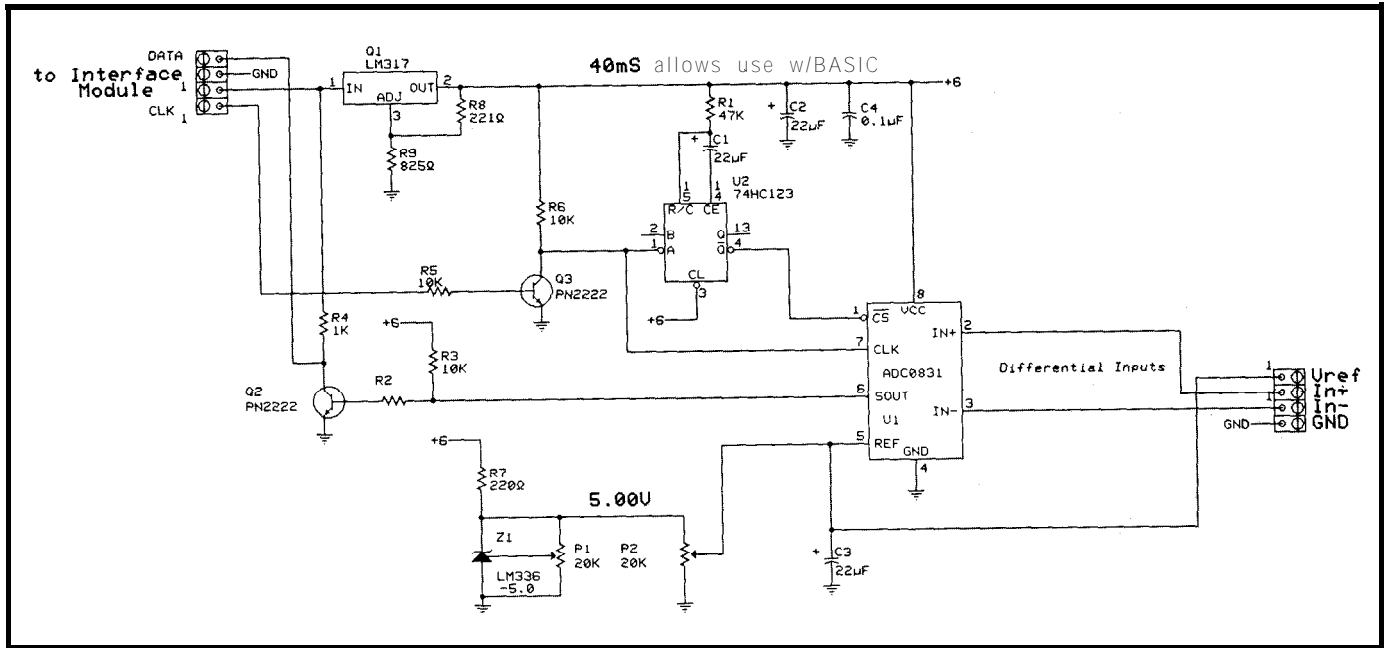


Figure 3—The serial 8-bit ADC module includes a one-shot to provide the ADC0831 with chip select.

the span (minimum to maximum range) of the input is traded for smaller incremental steps within the span. You might find that over a particular span you need finer incremental steps. If so, a more expensive ADC can be used. The MAX170 can be used if 12-bit resolution is necessary. It requires a CONVST (rising edge) signal to

initiate a conversion. A one-shot's Q output will provide this signal which must precede the next falling edge by 200 ns. The 12-bit data will be presented starting on the third falling edge, MSB to LSB. Note that unlike the ADC0831, the MAX170 does not output a "0" start bit. Maximum clock speed is greater than 1 MHz. Figure 4 illustrates a 12-

CIRCUIT CELLAR COLLECTIBLES

PROJECT COMPONENTS AT BLOWOUT PRICES

Item/Chip	CCC Project	Blowout Price
IS-32 Optic RAM	Micro D-Cam	\$25.00
SN76489	Sound Generator	\$5.00
SN76832AN IR REC	IR Remote Control	\$1.00
AY-3-1350 Synthesizer	Whimsi-Bell	\$3.50
TMS9918AN Sprite Gen.	Color Video Display	\$5.00
TMS9118AN Sprite Gen.	3-Chip Color Video Display	\$9.00
TMS99532AN Modem	Single chip 300bps Modem	\$5.00
TMS1121 Real Time Ctl. Chip	RTC-4 Timer/Controller	\$6.00
SP1000 Voice Recog. Chip	Lis'ner Voice I/O	\$10.00
HD6802P Processor	HCS - 6802 Processor	\$1.50
NCR5380 SCSI Controller	COMM180/SB180-SCSI	\$15.00
ADC0831 8-bit A/D	Lis'ner Voice I/O	\$2.00
Ceramic Mike	Acoustic Modem	\$1.00
Acoustic Modem Rubber Cups	Acoustic Modem	\$2.00 pair
4.032 MM Xtal	For TMS99532 Modem Chip	\$0.75
10.738 MHz Xtal	For Sprite Chips	\$0.75
7.16 MHz Resonator	For SP1000 Chip	\$0.75
BT450 Video DAC	GT180-Analog Video Driver	\$25.00
Z8613RS Z8 Protovac	Piggyback EPROM 28 Chip	\$15.00

Minimum order \$25.00.
Prices do not include shipping.

CIRCUIT CELLAR KITS

4 Park Street • Vernon, CT 06066
(203) 875-2751 • Fax: (203) 872-2204

ProControl!

Modular data acquisition and control for your IBM PC

At last, an industrial quality system that can start small and easily expand! Just insert the Host Adapter into your PC and then add only the I/O modules you need.

IBM PC Host Adapter \$129.95

Interfaces IBM PC's to ProControl bus I/O modules. Includes 24 digital I/O lines built-in and can connect directly to Opto-22 PB-24 I/O racks.

AIN-616 A/D Module \$729.95

Sixteen 8-bit analog-to-digital conversion channels, 50us conversion time (20us optional), on-board voltage reference and trim pots.

AOUT-84 D/A Module \$99.95

Four (expandable to six) B-bit digital-to-analog conversion channels.

Industrial Card Cage \$799.95

Safely house 8 I/O modules (expandable to 24 slots).

More I/O modules are available. Call for our FREE catalogue today!



(404) 352-47 88 1920 Moores Mill Road
Atlanta, GA 30318

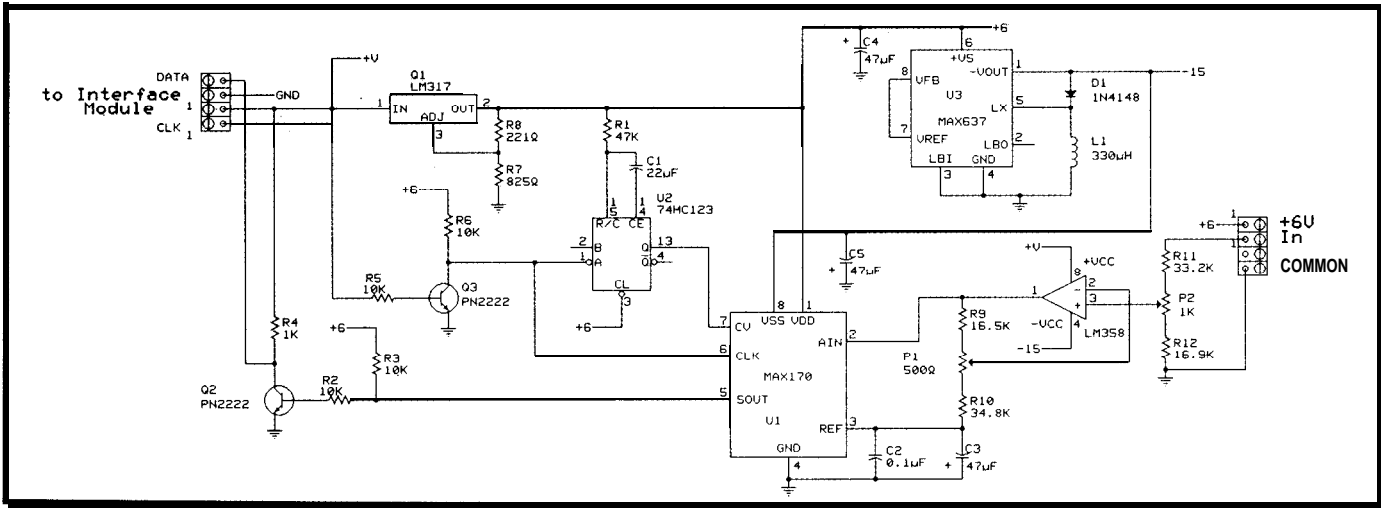


Figure 4—The serial 12-bit ADC module features a bipolar input with a range of ± 5 volts.

bit circuit. Notice the MAX637, from Maxim, which converts +5 volts to -15 volts for the MAX170 ADC.

Modular phone jacks (RJ-11) or quick disconnects can be substituted for the screw terminals. Extra output terminals can be mounted on your satellite modules. You may choose to bring out +12 V, +5 V, V_{ref} or other useful signals. Extra room on each satellite module will come in handy if you choose to add signal conditioning or enhance the circuit in some way.

CREATING CONTROL OUT OF DATA

Providing a controllable DAC voltage out can be handled much the same way as the ADCs previously discussed. The MAX500 is a quad serial 8-bit digital-to-analog converter. The protocol for converting digital to analog is much the same as one used in converting analog to digital. While the clock is still driven by the PC, the data

bits will be transmitted to the DAC instead of being read from the ADC. This protocol starts with clock and data high while "idle." A low output on the data line, while the clock is high, initiates the conversion. Following the falling edge of the clock, the appropriate data level is applied to the data line. Changes in the data level must occur only while the clock is low or the transmission will be reinitiated. The data bits are sampled within the device during the clock's rising edge. Ten bits total are necessary: two to indicate which DAC channel (1 of 4) will be updated, and eight for the 8-bit DAC value. After the clock's eleventh (final) rising edge, an internal *LDAC (Load DAC register) is implemented, updating the appropriate DAC channel. Maximum clock speed is greater than 1 MHz.

The MAX500 has four output channels in one 16-pin DIP package. The first two channels share a reference voltage input, but the other two channels have independent reference inputs. Supplies of +15 and -5 volts are

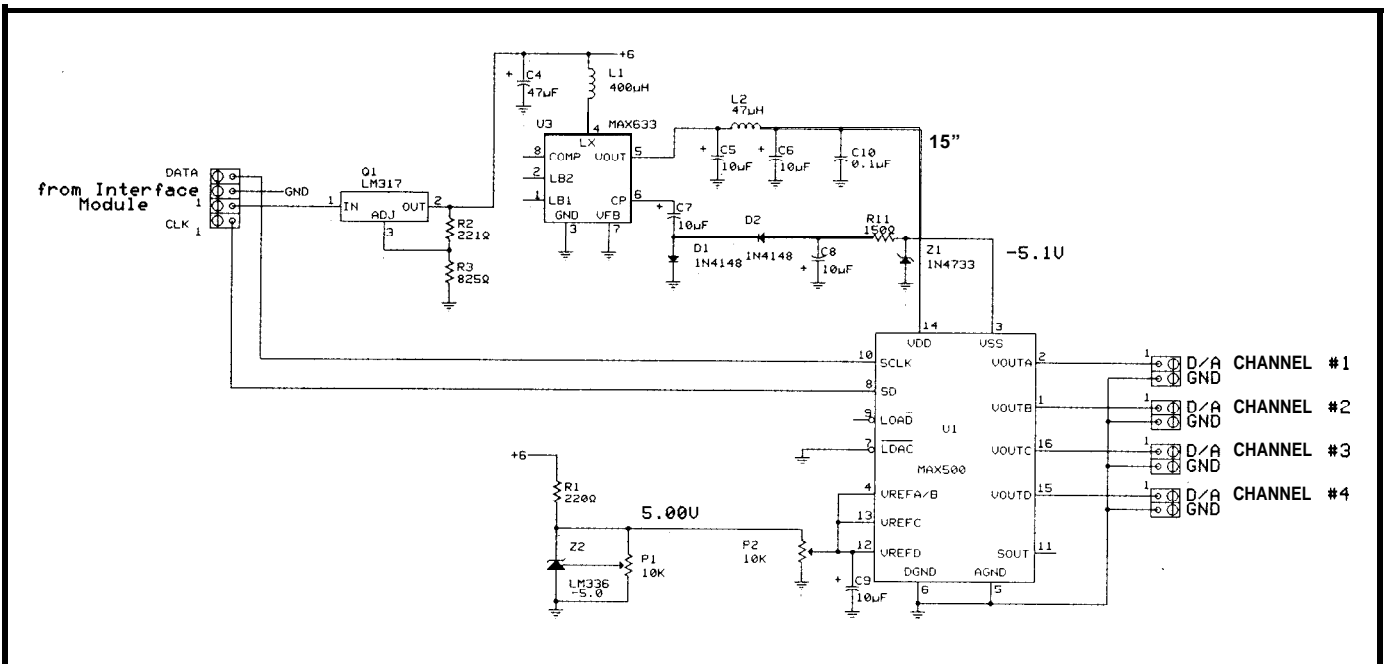


Figure 5—The serial quad 8-bit DAC module uses a MAX633 to generate the necessary +15V and -5.1 V needed by the MAX500.

NEW!! 68000, COP800, PIC16Cxx versions!

µASM™ Cross Assemblers
for the Macintosh™

•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

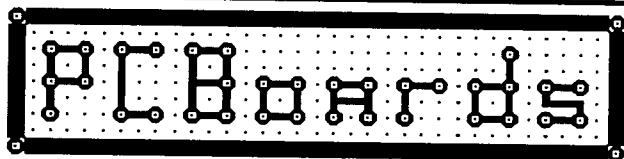
us \$149.95
EACH
PLUS S/H*

AVAILABLE FOR MOST 8-BIT MICROPROCES-
SORS AND 68000/010. CALL OR WRITE FOR
TECHNICAL BULLETIN. **30 DAY MONEY BACK
GUARANTEE.** MC/V/AE.

* PER SHIPMENT:
\$5 CONTIGUOUS USA
\$10 CANADA AK, HI
\$20 INTERNATIONAL

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

Reader Service #172



P-C-B ARTWORK MADE EASY ! Create and Revise PCB's in a Flash

- * HERCULES, CGA, EGA, VGA
- * HELP SCREENS
- * ADVANCED FEATURES
- * EXTREMELY USER FRIENDLY
- * AUTO GROUND PLANES
- * DOT- MATRIX, LASER and PLOTTER ART
- * CREATE YOUR OWN FILMS with 1X ART
- * 1 YR. FREE UPGRADES
- * LIBRARIES

REQUIREMENTS: IBM PC or Compatible, 384 K RAM
DOS 3.0 or later. IBM compatible printers.

PCBoards - layout program **99.00**
(PCBoards HP or HI PEN PLOTTER DRIVER 49.00)
PCRoute - auto-router **99.00**
SuperCAD - schematic pgm. **99.00**
Demo Pkg. - (includes all 3 programs) **10.00**

Call or write for more information
PCBoards

2110 14th Ave. South, Birmingham, AL 35205
1-800-733-PCBS / (205)933-1122

Reader Service #186

```

10 REM OUPUT 8-BIT VALUES TO CHANNELS 1-4 OF
   DAC MODULES 1-2
20 REM INPUT 8-BIT VALUES FROM THE ADC
   MODULES 1-4
30 CLS
40 PPWRITE=&H378 : REM PRINTER PORT OUTPUT
   REGISTER (COULD USE &H278 OR &H3BC)
50 PPREAD=&H378 : REM PRINTER PORT INPUT
   REGISTER (COULD USE &H278 OR &H3BC)
60 PSREAD=&H379 : REM PRINTER PORT STATUS
   REGISTER (COULD USE &H279 OR &H3BD)
70 A1=0 : A2=0 : A3=0 : A4=0 : REM ZERO ALL
   INPUT VARIABLES
80 LOCATE 1,1 : REM MOVE CURSOR TO TOP OF
   SCREEN
90 PRINT "Hit 1 or 2 to output to a DAC module"
100 I$=INKEY$
110 IF I$="1" OR I$="2" THEN GOTO 420 : REM DO
   DAC ROUTINE IF ASKED
120 OUT PPWRITE,&HOF : REM START ADC CLOCKING
130 OUT PPWRITE,&HFF
140 OUT PPWRITE,&HOF : REM SECOND CLOCK
150 OUT PPWRITE,&HFF
160 S=INP(PSREAD) : REM SAVE THE STATE OF THE
   ADC DATA LINES (FOR START BIT)
170 FOR X=7 TO 0 STEP -1 : REM LOOP COUNTER FOR
   8 BITS MSB-LSB
180 OUT PPWRITE,&HOF : REM CLOCK FOR EACH BIT
190 OUT PPWRITE,&HFF
200 V=INP(PSREAD)
210 REM CHECK EACH ADC'S DATA BIT AND ADD ITS
   WEIGHTED VALUE TO ITS VARIABLE
220 IF (V AND &H10)=&H10 THEN A1=A1+(2^X)
230 IF (V AND &H20)=&H20 THEN A2=A2+(2^X)
240 IF (V AND &H40)=&H40 THEN A3=A3+(2^X)
250 IF (V AND &H80)=&H80 THEN A4=A4+(2^X)
260 NEXT X
270 REM PIN 13
280 REM NOW CHECK FOR LEGAL START BIT AND PRINT
   THE ADC VALUES IF LEGAL
290 PRINT "CHANNEL 1 = ";
300 IF (S AND &H10)<>0 THEN PRINT "BUSY" ELSE
   PRINT A1;" "
310 REM PIN 12
320 PRINT "CHANNEL 2 = ";
330 IF (S AND &H20)<>0 THEN PRINT "BUSY" ELSE
   PRINT A2;" "
340 REM PIN 10
350 PRINT "CHANNEL 3 = ";
360 IF (S AND &H40)<>0 THEN PRINT "BUSY" ELSE
   PRINT A3;" "
370 REM PIN 11
380 PRINT "CHANNEL 4 = ";
390 IF (S AND &H80)<>0 THEN PRINT "BUSY" ELSE
   PRINT A4;" "
400 FOR Q=0 TO 1000 : NEXT Q : REM WAIT TO
   ENSURE ONE-SHOT TIMEOUT
410 GOTO 70 : REM DO IT ALL AGAIN
420 REM THIS IS THE DAC ROUTINE
430 LOCATE 1,1
440 M=ASC(I$)-&H30 : REM CONVERT MODULE # FROM
   ASC
450 PRINT "Hit 1, 2, 3 or 4 to select the
   channel of module ";M
460 I$=INKEY$
470 IF I$="1" OR I$="2" OR I$="3" OR I$="4"
   THEN 490 : REM CONTINUE IF LEGAL
480 GOTO 460
490 LOCATE 1,1
500 N=ASC(I$)-&H30 : REM CONVERT CHANNEL NUMBER
   FROM ASC
510 PRINT "Enter a value between 0 and 255 to
   output to module ";M;" channel ";N;
520 INPUT V : REM THIS IS OUR OUTPUT VALUE
530 REM SETUP THE 'AND' AND 'OR' MASKS FOR THE
   CLOCK AND DATA
540 IF M=1 THEN AC=&HFE ELSE AC=&HFB : REM AC
   IS THE 'AND' CLOCK MASK
550 IF M=1 THEN OC=&H01 ELSE OC=&H04 : REM OC
   IS THE 'OR' CLOCK MASK
560 IF M=1 THEN AD=&HFD ELSE AD=&HF7 : REM AD
   IS THE 'AND' DATA MASK

```

(continued)

Listing 1—Test program written in PC BASIC.

```

570 IF M=1 THEN OD=&H02 ELSE OD=&H08 : REM OD
    IS THE 'OR' DATA MASK
580 OUT PPWRITE, (INP (PPREAD) AND AD) : REM
    BRING LOW THE DATA LINE
590 FOR X=2 TO 1 STEP -1 : REM LOOP FOR 2
    ADDRESS BITS MSB TO LSB
600 IF ((N-1) AND X)=0 THEN Q=0 ELSE Q=1 : REM
    SET UP 'Q' AS DATA FOR SUBROUTINE
610 GOSUB 730
620 NEXT X
630 FOR X=7 TO 0 STEP -1 : REM LOOP FOR 8 DATA
    BITS MSB TO LSB
640 IF (V AND (2^X))=0 THEN Q=0 ELSE Q=1
650 GOSUB 730
660 NEXT X
670 OUT PPWRITE, (INP (PPREAD) AND AC) : REM STOP
    BIT FOR SEQUENCE TERMINATION
680 OUT PPWRITE, (INP (PPREAD) AND AD)
690 OUT PPWRITE, (INP (PPREAD) OR OC)
700 OUT PPWRITE, (INP (PPREAD) OR OD)
710 CLS
720 GOTO 60 : REM DO IT ALL AGAIN
730 REM OUTPUT DATA BIT ROUTINE (Q HAS BIT
    VALUE)
740 OUT PPWRITE, (INP (PPREAD) AND AC)
750 IF Q=0 THEN OUT PPWRITE, (INP (PPREAD) AND
    AD)
760 IF Q=1 THEN OUT PPWRITE, (INP (PPREAD) OR OD)
770 OUT PPWRITE, (INP (PPREAD) OR OC)
780 RETURN

RUN

Hit 1 or 2 to output to a DAC module
CHANNEL 1 = BUSY
CHANNEL 2 = 137
CHANNEL 3 = BUSY
CHANNEL 4 = BUSY

```

listing 1 -continued

required for this device. Here I chose to use a MAX633 which will develop both voltages needed for the MAX500.

Each DAC output will drive 2k ohms and 100 pF (5 mA source and sink) at a slew rate of 3 V/μs. The maximum settling time to half an LSB is less than 4 μs for a 10-volt maximum output swing. Vss can be at ground, but the "zero code error" can be 30 mV off due to the buffer's pull-down circuitry. Using a negative bias on Vss eliminates the error. See Figure 5 for this circuit.

CONVERTING MOVING DATA

In order for "successive approximation" A/D converters to provide adequate results, the input must not move more than one-half of the unadjusted error during the conversion process. This means that the clock speed and the number of clock cycles required for a conversion is directly proportional to the maximum change in analog input. Slowly changing temperatures wouldn't be a problem, however sampling speech would be. A rule of thumb for maximum analog signal rate of change is:

$$F_{\max} = \frac{F_{\text{clock}}}{C \times 2^n \times 2}$$

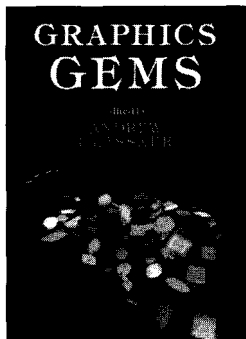
where:

F_{clock} = ADC clock frequency

C = number of cycles per conversion

n = number of bits of resolution

GET GRAPHIC

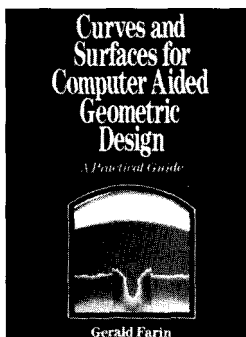


Graphics Gems

edited by
Andrew S. Glassner

This handbook provides practical solutions to graphics problems, and every graphics programmer will find it an essential tool in saving time and energy in their daily programming activities.

August 1990, 833 pp., \$49.95
ISBN: 0-12-286165-5



Curves and Surfaces for Computer Aided Geometric Design

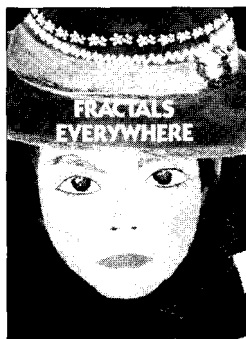
A Practical Guide
SECOND EDITION

Gerald Farin

From a Review of the First Edition:
"An interesting and informative text."

-MATHEMATICALREVIEWS

1990, 444 pp., \$39.95/ISBN: 0-12-249051-7



The Desktop Fractal Design System

Michael F. Bamsley

"This is a simple package to use...instructive and enjoyable!"

-PERSONAL COMPUTER WORLD

Includes **The Desktop Fractal Design Handbook** and one floppy disk.

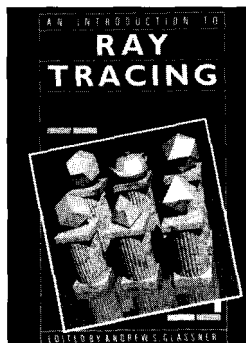
IBM Version:

1989, \$39.95/ISBN: 0-12-079063-7

Macintosh Version:

August 1990, \$39.95

ISBN: 0-12-079064-5



Fractals Everywhere

Michael F. Bamsley

"I think that *Fractals Everywhere* will commonly be referred to as the 'classic' in its field."

-THE AMERICAN
MATHEMATICALMONTHLY

1988, 394 pp., \$44.50/ISBN: 0-12-079062-9

An Introduction to Ray Tracing

edited by
Andrew S. Glassner

1989, 327 pp., \$49.95/ISBN: 0-12-286160-4

Write for a FREE brochure!

Order from your local bookseller or directly from

ACADEMIC PRESS

Harcourt Brace Jovanovich, Publishers
Book Marketing Department #07021
1250 Sixth Avenue, San Diego, CA 92101

CALL TOLL FREE

1-800-321-5068

Quote this reference number for free postage and handling on your prepaid order - 07021

Prices subject to change without notice ©1991 by Academic Press, Inc. All Rights Reserved. TC/SS-07021

Order Service #101

For the ADC0831 with a 400-kHz clock at 10 cycles/ conversion and 8-bit resolution, that's

$$F_{\max} = \frac{400000}{10 \times 2^8 \times 2} = 78 \text{ Hz (Triangle Wave)}$$

Raising the clock frequency is not the answer for vast improvements. The most effective solution is a **sample-and-hold** between the analog signal and ADC input. This will take an instantaneous snapshot of the analog value and store it for as long as the conversion takes. The maximum rate of change is then limited by the conversion time and not the rate of change during a conversion. Most processes have small rates of change (temperature, atmospheric pressure, or the level of a fluid in a tank), so complicating the hardware may not be necessary.

The slew rate for DAC outputs is the limiting factor on maximum output frequency (assuming a maximum excursion). At 3 V/ μ s with a 10-volt excursion, the maximum frequency equals:

$$F_{\max} = \frac{3 \text{ V}}{10 \text{ V} \times 1 \mu\text{s} \times 2} = 150 \text{ kHz}$$

The PC BASIC program in Listing 1 shows how to use four ADC0831s to provide four 8-bit ADCs and two MAX500s to provide eight 8-bit DACs. The one-shot's timeout should be shortened if you want to use a lower-

level routine for faster conversion. Simply adjust the RC time constant to exceed one clock cycle.

DOLLARS AND SENSE

Although industrial acquisition and control is usually associated with "big dollars," it usually is bundled with flashy software and an embedded processor. Here, making use of an existing PC greatly reduces cost.

Approximate parts cost for this project, excluding enclosures, when purchased through catalog distributors is as follows: PC interface-less than \$25 including power supply; ADC0831 module-less than \$15; MAX170 module-less than \$40; MAX500 module-less than \$30.

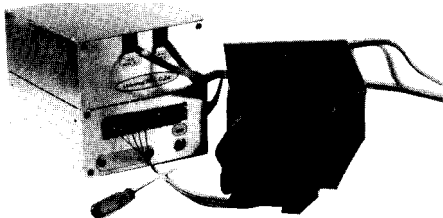
Additions to this project might include **sample-and-hold** inputs, signal conditioning for specialized inputs, optoisolation of each of the satellite modules, or PC-bus-mapped I/O ports. Once you start adding on, you'll find it hard to stop. ❖

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on the Circuit Cellar INK engineering staff. His background includes product design and manufacturing.

IRS

- 425 Very Useful
- 426 Moderately Useful
- 427 Not Useful

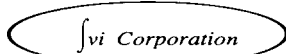
Universal RiTC Cube & RiTC UPS Case



Universal RiTC UPS Cabinet (RiTC-UPS8/4): A custom designed 96 HOUR Rechargeable Battery Power Supply, ON-LINE DC UPS enclosure for Micromint RTC-based applications. Supports up to three (3) board RTC stack of the Micromint RTC family. Provides up to 96 hour (HEAVY DUTY 8Ah Dual Lead Acid Gel-Cell) holdup time at +5VDC, 80mA (48 hour / 4 Ah mod 9 available). Instantaneously switches to Battery in loss of line power and returns to line power when restored. Built-in battery charger fully charges battery when AC power is present. Regulated line power supply provides +5VDC @ 500mA from a 120 VAC input / +9VDC output UL approved wall unit. Includes battery discharge (blinking red) & battery charge (solid green) LED indicators and battery & line power switches. Packaged in elegant silver and black RiTC-CUBE enclosure described below.

Enclosure - (RiTC-CUBE): Constructed from 16 gage brushed clear coated aluminum. 4-40 machine screws and PEM nuts for assembly. Full height case supports up to 7 RTC boards. Half height box supports up to 3 RTC boards. Includes: 3M bumper rubber feet & standoffs. Custom connector cutouts and silkscreening available. call for pricing.

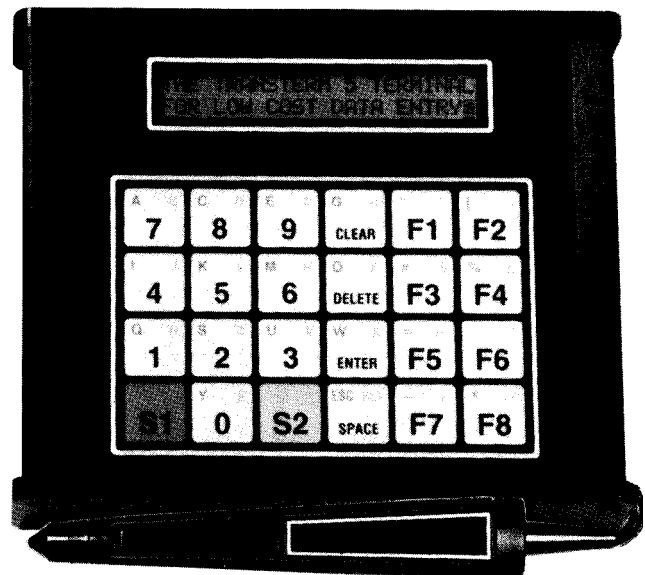
ITEM	DESCRIPTION	PRICE
RiTC-UPS4	4 Ah UPS "Full Height" RiTC CUBE case w/500mA UL Power Supply	\$169.95
RiTC-UPS8	8 Ah UPS "Full Height" RiTC CUBE case w/500mA UL power Supply	\$299.95
RiTC-CUBE Bare Bones	"Full Height" RiTC CUBE case sprrts 7 RTC stk, incl. H.W.	\$99.95
RiTC-HCUBE Bare Bones	"Half Height" RiTC CUBE case sprrts 3 RTC stk, incl. H.W.	\$69.95
RiTC-HKIT	RiTC Headlight Kit-7 LED's + driver, power & reset switches all on a board. Mounts in RiTC CUBE faceplate.	\$29.95
RiTC-CKIT	Communication Kit A pair of RJ 12 sockets configurable for MC-NET or dual serial ports on a board. DTE/DCE Selectable. Mounts in faceplate.	\$29.95
RiTC-1243Y	Battery Protected 8KB SRAM Real-Time-Clock, plugs into RTC memory sockets RTC-31/52/180/V25/HC11. Includes clock source code	\$34.95
RiTC-1244Y	Battery Protected 32KB SRAM RTC, same as above 4X storage.	\$ 49.95



Integrated Vessel Information Corporation
671 Via Alondra, Unit 805
Camarillo, California 93012
805-389-6870

Reader Service #156

\$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-422I/O, 20 Ma current loop I/O,

COMPUTERWISE, INC.

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

Reader Service #11

Hot Chips In The Summertime

SILICON UPDATE

Tom Cantrell

A Repot? From "Hot Chips II"

It's a truism of high-tech reporting-call it "Cantrell's Law Of Editorial Irrelevance"-that the amount of breathless coverage of a technology is inversely related to real-world applicability.

So, succumbing to the immutable law myself, let me tell you about the recent "Hot Chips II" symposium sponsored by the IEEE here in Silicon Valley. You'll hear all about the latest chips that only a starving reporter or university professor can love, chips that you won't see on your desktop any time soon. I promise to write about some 8-bit chips again soon, but for now, the "blue sky" is the limit.

In case you haven't heard, RISC (Reduced Instruction Set Computer) is all the rage. As I've said before, the RISC furor is really more about marketing (i.e., how to dethrone Intel and Motorola) than technology. Thus, Hot Chips II had nary a word about 80x86s or 680xxs. These chips fail the relevance test since too many are sold to merit the interest of gurus. Of course, the fact that the organizing committee is packed with those promoting RISC (Silicon Graphics [MIPS], Sun [SPARC], and their U.C. Berkeley and Stanford patrons) might have something to do with it.

Anyway, here we go. Hang on to your logic probes. . .

IT'S A GAAS

First up is a chip (actually a multichip module, single chips having become passé) that literally sizzles!

Take a SPARC architecture, throw in an esoteric process (GaAs, or Gallium Arsenide) and a healthy dose of NASA funding and voilà: a 200-MHz code-burner from System and Processes Engineering Corporation (a.k.a., SPEC) that is guaranteed to turn heads.

Packing in 12 separate chips (integer, float, two cache controller/MMUs, and eight 4K x 8 cache RAMs) isn't easy, but there is really no choice; it's the only way to deal with the ugly realities of a 5-ns clock cycle. Even integration doesn't solve all the system designer's problems: Don't expect to hang 100-ns DRAMs on this

puppy. Instead, ECL (Emitter Coupled Logic, the next most esoteric after GaAs) main memory is called for.

Of course, since the goal is performance at any cost, you won't quibble about the module's \$5-\$6k cost. Of more immediate concern is the fact this little guy dissipates somewhere between 50 and 75 watts.

So, I hereby file first claim to the concept of a liquid-cooled PC which doubles as an aquarium.

LIGHTNING STRIKES

Simply boosting clock rate isn't fair. Conceptually, it's too easy and furthermore, it takes control from the computer gurus and hands it to the chemists and physicists.

So, the same "architects" that brought you RISC are struggling to wring more performance from the design of the machine, not its implementation. Unfortunately, though it isn't polite to bring up in mixed company, architecture has pretty much reached a dead end. Even the most clever innovation contributes only fractionally to performance (versus the order-of-magnitude impact of process/clock rate). Nevertheless they're severe.

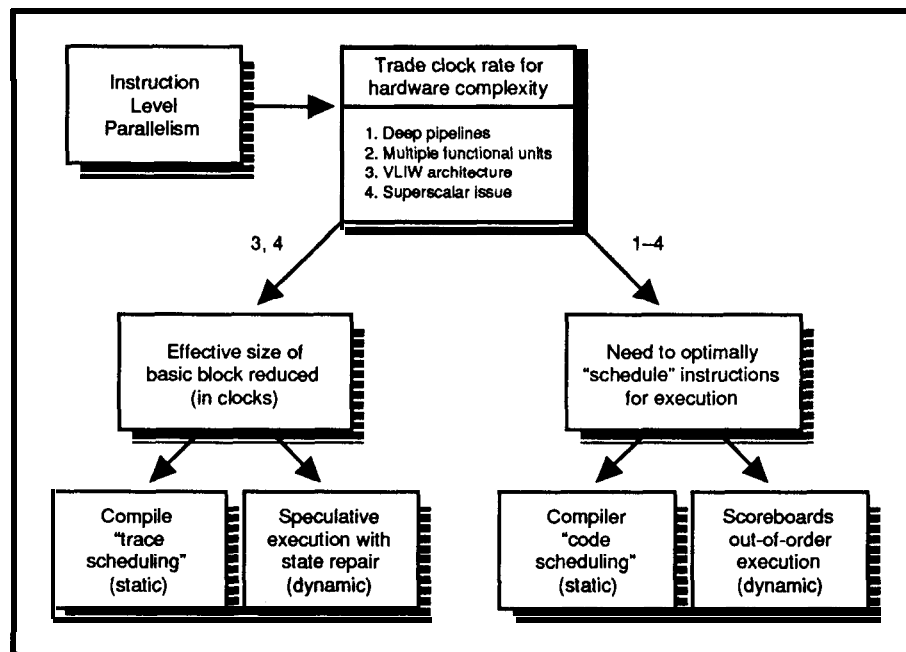


Figure 1 — Adding parallelism to a processor doesn't come without complexity cost.

For some time, effort has been focused on “superscalar” techniques. The idea is to try to increase execution “parallelism” without confusing the poor programmers who have enough trouble thinking sequentially. Where the original goal of RISC was to execute one instruction per clock, the new goal is multiple instructions per clock.

Making the process transparent to the programmer calls for both hardware and compiler techniques. The hardware inherently requires “multiple functional units” to execute multiple functions at the same time. Unfortunately, that’s not all there is to it. Though multiple units may be offered, it’s not easy to keep them all busy. In the attempt to do so, additional techniques, such as a scoreboard (hardware that allows out-of-order instruction execution), and compiler software which attempts to “schedule” execution to avoid “hazards” and “stalls” is required (Figure 1). This is heavy stuff, and the reality is that chips which can theoretically execute “x” instructions per clock are lucky to achieve half that running real applications.

The bane of all computer architects is the conditional, or even worse calculated, branch. Deep pipelines, which can happily chug through sequential instructions, are forced to throw up their hands and ask, ‘Where to next?’

From Metaflow Technologies Inc. comes yet another SPARC variant—the Lightning—which attempts to deal with the problem by throwing hardware at it in the form of a “Dataflow Content-Addressable FIFO” (DCAF). The idea is simple enough: If you’re not sure where to go, don’t just stand there. Instead, go somewhere and hope for the best. If you’re right, things are ducky. If you make a wrong turn, try to backpedal as fast as you can.

The idea is called “speculative execution.” In the face of a forthcoming conditional branch, go ahead and execute down a path even before the condition is evaluated. If you goof, back everything up to the way it was; a process known as “state repair.”

Consensus is that the Metaflow chip is pretty neat, with the theoretical four instructions per clock machine actually able to achieve about two instructions per clock average throughput—somewhat better than previous superscalars.

On the downside, the scheme can’t deal with “calculated” branches—ones in which the target is based on data, not a condition. A conditional branch is going to go one of two ways, but a calculated branch could end up anywhere.

In any case, all these schemes are clearly a case of more hardware and compiler tricks asymptotically chasing fewer MIPS. Oh well.. .

BENCHMARK REALITY

How in the world do we know just how “hot” a chip is short of using a thermometer? Actually, given the overriding influence of process and clock rate, the thermometer approach might not be a bad idea.

Until that idea takes off, the world relies on benchmarks, and according to Rafael Saavedra-Barrera from U.C. Berkeley, this is a cause for concern.

His presentation started by describing two popular sets of benchmarks: SPEC (System Performance and Evaluation Cooperative) and the so-called Perfect Club. Both sets are somewhat an improvement over older “synthetic” benchmarks like Dhrystone and Sieve. The newer packages are composed of “real” applications which notably exercise caches in a realistic manner.

The optimistically named ‘Perfect Club’ is a set of 13 FORTRAN programs which are purely scientific and especially suitable for evaluating vector and array processors. The set is, so far, not prone to excessive marketing hype since no one pretends that “transonic inviscid flow past an airfoil” is relevant to Joe Commercial User.

SPEC, on the other hand, is promoted as a “real-world” mix. The reality is SPEC is somewhat biased towards floating-point performance. Reflecting our need to keep things simple, a single “SPEC” number is computed. The results can be misleading. Conventional wisdom is that RISCs are “far superior” to CISCs. For example, the “SPECmark” for an IBM RS6x20 RISC is 22.3 while that for an AT&T ‘486 machine is only 11.6. Hidden beneath the surface is the fact that, if the floating-point benchmarks are removed, the ‘486 “Integer SPECmark” of 17.2 surpasses the IBM’s 15.8. The paranoid might point out that the originators of SPEC happen to be those offering machines with superior floating-point performance. Coincidence?

I’ve always argued that none of these benchmarks make any attempt to measure I/O—arguably something of real interest to many. Furthermore, they all ignore the

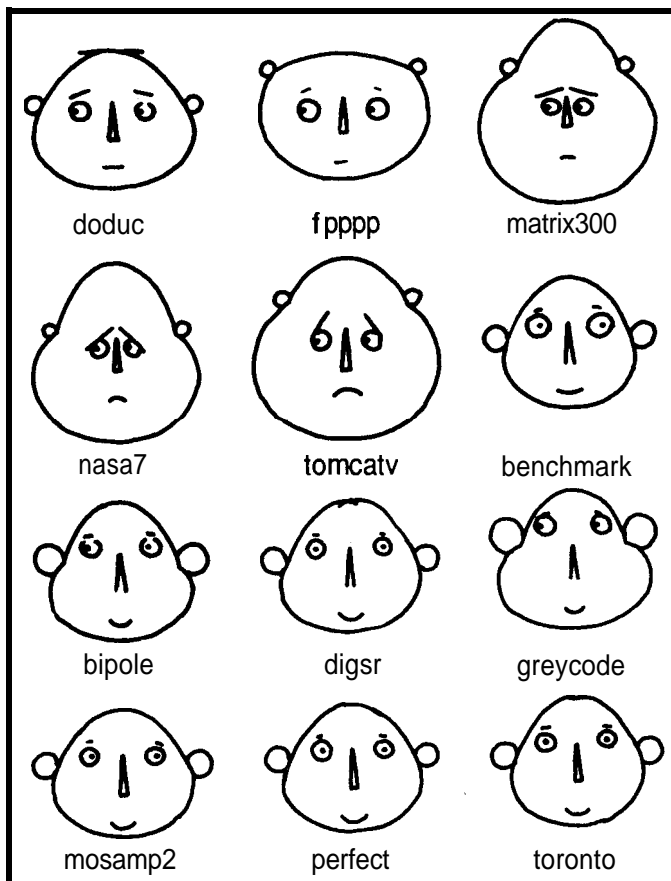


Figure 2—Chernoff Faces are used to characterize benchmarks.

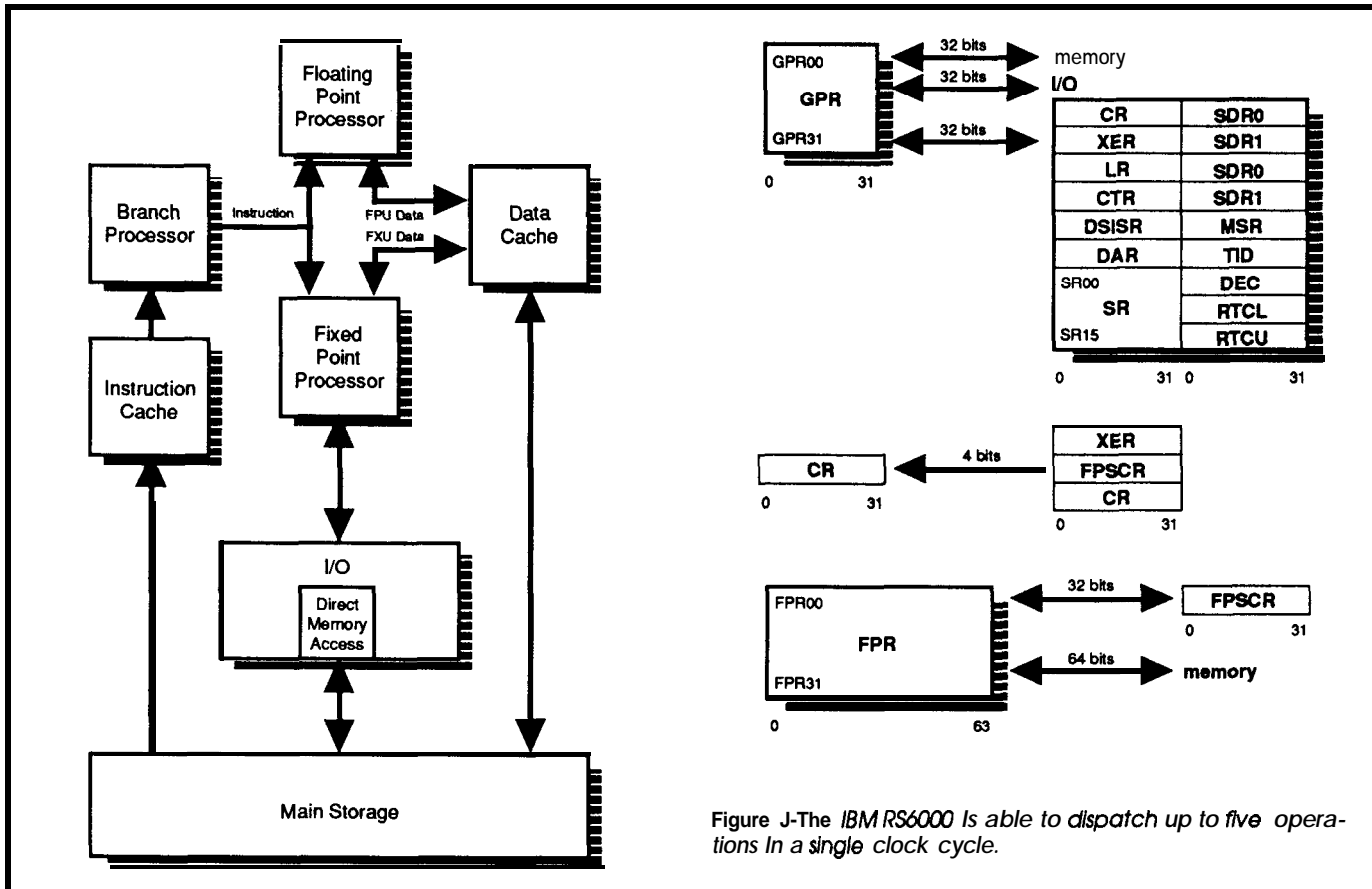


Figure J-The IBM RS6000 is able to dispatch up to five operations in a single clock cycle.

issue of compiler quality variation-something that can have far more impact than CPU architecture. Indeed, it seems that various contenders are taking compiler "tweaking" to the extreme, releasing new versions every few months to boost their SPECmark a point or two. It's not clear that the new compilers make applications run faster.

Putting these issues aside, Saavedra-Barrera makes the point that the benchmark fixation is dangerous since the "numbers" by themselves give evaluators little or no ability to predict the performance of their own applications. For instance, the IBM's SPECmark of 22.3 is largely driven by superlative performance on a few of the benchmarks. Looking further, it turns out these particular benchmarks are heavily dependent on tight "multiply/add" loops. And guess what? The IBM machine includes a specialized "Multiply & Add" instruction. Thus, the "truth" is that the high SPECmark will translate into high performance for **your** application if it is **dependent** on **multiply** and **add**. Otherwise you might be better off with a '486.

To get a handle on which benchmarks measure what, Saavedra-Barrera characterized their key **properties-data** type, data structure, and basic blocks--and mapped them to Chernoff Faces (the way a person "looks" depends on a few key parameters: head shape, eye spacing, mouth size, etc.) as shown in Figure 2. You should rely on the benchmarks that most resemble your application.

Ah, a blessed breeze of reason. However, I fear it's for nought. The marketing guys have their teeth into simple numbers and nobody has time for "messy" explanations.

68000

Z8 & Z80 Too!

- EMBEDDED DEBUGGERS** from \$250
 - ASSEMBLER, DISASSEMBLER
 - BREAKPOINT / TRACE
 - OEM OBJECT LICENCES AVAILABLE
- ROM EMULATORS** from \$150
 - PARALLEL PORT INTERFACE
 - 2764, 27128 OR 27256
 - MAY BE GANGED FOR 8, 16 OR 32 BITS
 - UP TO 1 MEG. TOTAL MEMORY CAPACITY
- UnderRom SERIAL INTERFACE** from \$395
 - TEMPORARY SERIAL PORT FOR DEBUGGING
 - USE UNDER 2764, 27128, 27256 OR 27512
- CROSS ASSEMBLERS** from \$50
 - USE MANUFACTURERS OPCODES
 - INTEL / MOTOROLA HEX OUTPUT
 - LOCAL SYMBOLS

Control Resources

P.O. Box 8694
Rowland Hts, CA 91748
(818) 912-5722 x3100

IBM-A STRANGER IN OUR MIDST

Don't get me wrong, the IBM RS6000 (Figure 3) is really a pretty neat machine; as superscalar as the best of them since it is able to dispatch up to five operations (branch, control reg, load, and two of the previously mentioned multiply&adds) in a single clock. However, I find it ironic that the machine, based on the "original" IBM 801 RISC (did you know IBM has "RISC" patents that outfits like Sun are paying to license), is promoted as having "high-function ops" including string instructions, bit field handling, and other decidedly "CISCy" features.

More interesting than the bits/bytes is the fact that IBM has a competitive machine--something that is traditionally not supposed to happen. The reaction of the "lean and mean" newcomers, largely university types who disparage all manners of "big business," seems to range from curiosity to denial to trepidation. IBM has said they will double performance shortly (mainly by, you guessed it, increasing the clock rate). It remains to be seen if IBM can shed their stodgy image or whether the gurus will ever wear big blue suits.

Nevertheless, they still do some goofy things from time to time. An astute member of the audience noted that buried in their floating-point unit design (which indeed, delivers superior performance) was some odd-looking hex conversion logic. Yes, the IBM unit manages IEEE binary floating point by converting to and calculating in hex and then reconverting the result to binary. What the heck, it works.

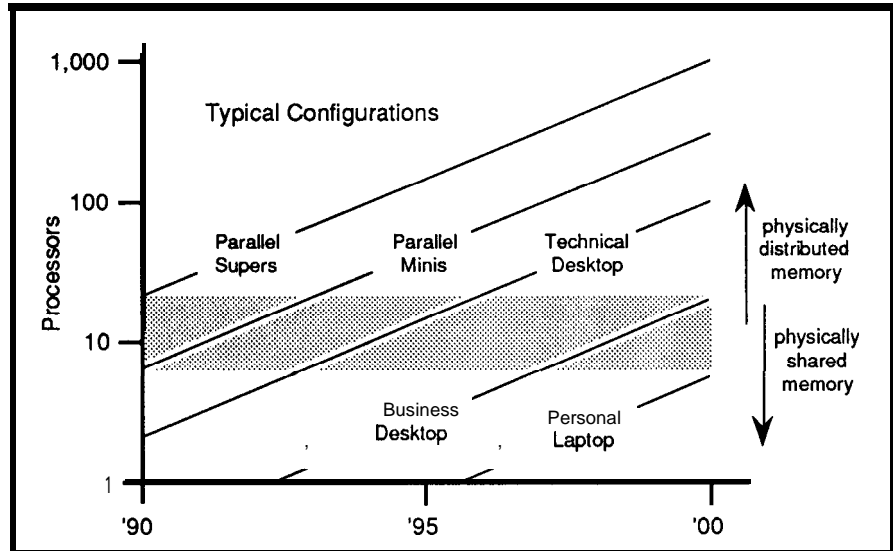
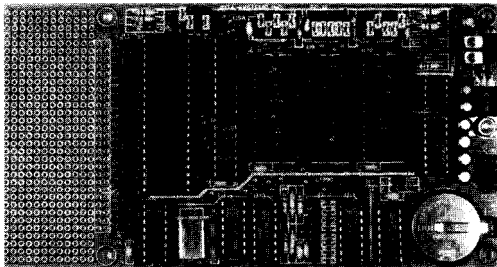


Figure 4-Numbers of processing elements in even small computers are expected to increase linearly over the next decade.

EMBEDDED CONTROLLER for Quick Development



The EC-32™ is a versatile 80C32 microcontroller board. It is ideal for quickly developing products, prototypes or test fixtures.

- 80C32 microcontroller (8051 compatible)
- 35 mA operating current, 100 uA standby
- Program in C, BASIC or assembly language
- 8 - 92K RAM, EPROM, EEPROM
- Breadboard area and expansion bus
- RS-232 port and 12 digital I/O lines
- \$100

Iota Systems, Inc.
POB 8987

Incline Village, Nevada 89450
(702) 831-6302

HAJAR ASSOCIATES

NATIONAL ADVERTISING
SALES REPRESENTATIVES

NORTHEAST

Lisa D'Ambrosia
49 Walpole Street
Norwood, MA 02062
(617) 769-8950
Fax: (617) 769-8982

MIDWEST

Nanette Traetow
242 East Ogden
Avenue, Suite A
Hinsdale, IL 60521
(708) 789-3080
Fax: (708) 789-3082

MID-ATLANTIC

Barbara Best
569 River Road
Fair Haven, NJ 07704
(201) 741-7744
Fax: (201) 741-6823

WEST COAST

**Barbara Jones &
Shelley Rainey**
3303 Harbor Blvd.,
Suite G-1 1
Costa Mesa, CA 92626
(714) 540-3554
Fax: (714) 540-7103

SOUTHEAST

Christa Collins
7640 Farragut Street
Hollywood, FL 33024
(305) 966-3939
Fax: (305) 985-8457

PARALLEL PARADE

Parallelism at the chip level is fine, but the true Holy Grail is massively parallel systems which combine dozens, hundreds, or even thousands of CPUs. The idea is tantalizingly simple: If one CPU can achieve X MIPS, then try to come up with a way to hook up Y CPUs to achieve X x Y MIPS.

Unfortunately, as in the case of chip-level parallelism, taking full advantage of the "potential" bandwidth is easier said than done.

Intel has a Scientific Computer group that has been fiddling with this for some time. Hey, as long as DARPA (Defense Advanced Research Projects Agency) will pay, why not? They talked about two machines: the Touchstone Delta and iWARP. The former is supposedly targeted at scientific and engineering computing while the latter is promoted as "parallel building blocks for signal and image processing."

Other than different nodes or processing elements (the Delta uses i860s while the iWARP uses a new "LIW" chip [Long Instruction Word, more in a moment]), the two systems share a lot in common (besides DARPA funding). Notably, yesterday's hypercube organization has apparently lost favor compared to a simple "mesh" 2-D design.

On the one hand, both machines are targeted to meet so-called Grand Challenges including vision, superconductor modeling, global change, and similar lofty goals. Indeed, to the degree these problems are parallelizable, the Intel machines can do a good job. For instance, in the superconductor modeling application, Intel projects that a

128-node machine can surpass the performance of an 8-CPU CRAY Y/MI'.

However, Intel seems to be trying to make a pitch that these designs are also usable in the commercial world. They **project** (hope?) that even our lowly PCs will consume dozens of 80x86s in the next few years (Figure 4). I'm skeptical since I don't think massively parallel architectures have much to offer in meeting the "Grand Challenges" of the commercial world such as making programming possible by mortals or eliminating user manuals.

BEYOND RISC-LIW

The quest for parallelism that permeates computing these days has led to a new scheme called "Long Instruction Word," "Very Long Instruction Word," presumably later "Darn Long Instruction Word," and so on.

This is conceptually similar to superscalar in that the goal is to do many things at once. The difference is that superscalar attempts to preserve the image of a regular processor in that programmers "see" single-function instructions which the hardware and compiler attempt to munch together transparently.

LIW takes a simpler tack. Each "long" instruction contains a number of fields, each of which explicitly drives an associated functional unit.

For instance, each Intel iWARP 96-bit instruction controls five functions: single-precision float, integer/address, memory access, input, and output.

Upping the ante, a processor core called LIFE from Philips (Figure 5) boasts a 200-bit instruction feeding five

functional units: branch, register, ALU1, ALU2, and memory access. Once again, the challenge is keeping everything busy. LIFE benchmarks show that about 50% utilization is about the best that can be expected. Thus, the speed-up over a single functional unit design is roughly 2.5X instead of the theoretical 5X.

The Achilles heels for LIW are branch delays and code density. The branch delay problem is straightforward: where a single cycle delay would sacrifice one operation on a regular CPU, five are sacrificed on these LIWs. As for codedensity, 50% utilization of functional units implies 50% unutilization of instruction bits, that is, code density is half that of

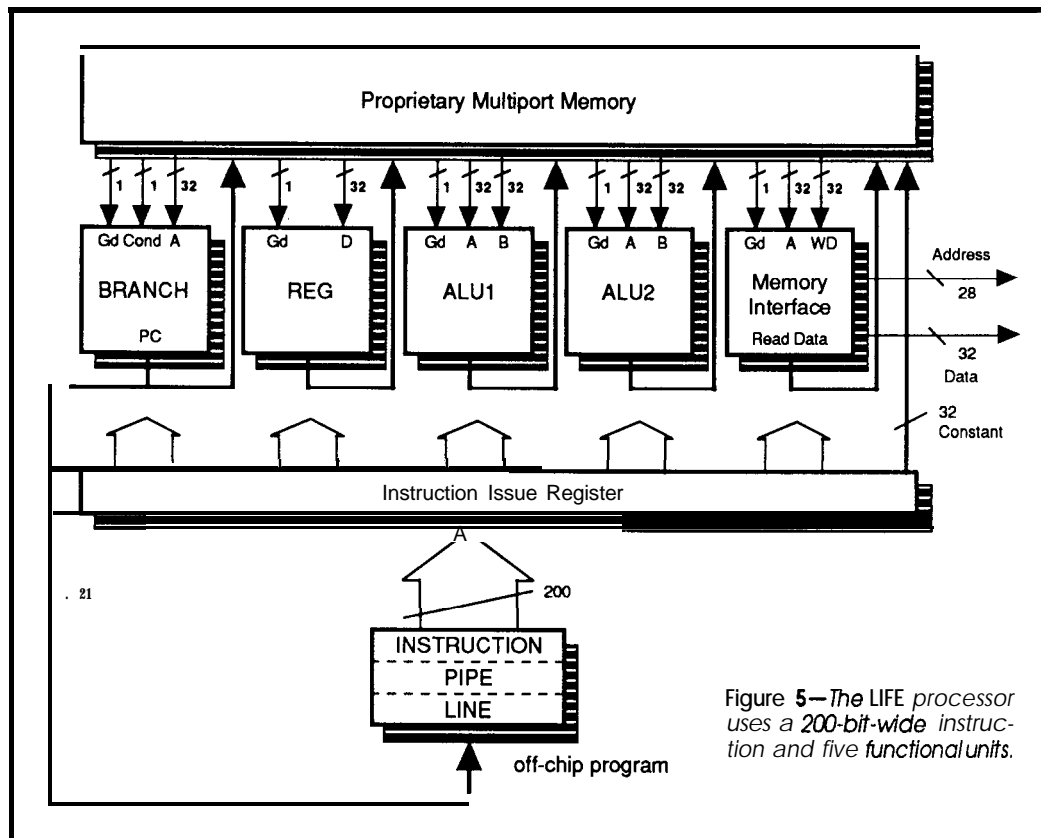


Figure 5—The LIFE processor uses a 200-bit-wide instruction and five functional units.

a regular CPU or, in the words of the Philips speaker, "lousy." Though memory is cheap, code density is a problem for LIFE since the ultimate goal is a single-chip device with memory on-chip. System designers would rather not deal with a 200-bit external memory bus.

For the LIFE of me (ha ha), I can't really see the difference between LIW and the classic "microcode." Though the "operations" controlled may be different, the idea of a long instruction with different fields working at the same time is not new. The more things change, the more they stay the same.

CHIP SHOT

Some would say that the "Hot Chips" are really pretty lukewarm. It seems clear that, having integrated all of computer history onto a single chip, future progress will not proceed at a blinding pace.

Nevertheless, advances are being made that will ultimately trickle down to your average system designer. Particularly, the speeding of process/clock rates continues—everything from 50-MHz CMOS to 200-MHz GaAs—and will yield brute-force performance gains.

And whatever your favorite architecture—RISC, CISC, LIW, massively parallel, etc.—they all keep getting cheaper. Remember Gelbach's Law (named after an Intel sales manager): "Every chip will be \$5—or less!" Of course, this law was stated before the era of sole-source monopolies, but the underlying idea still holds.

Now all we need is a "Cheap Chips" conference... ❖

Contact

System Processes and
Engineering Corp. (SPEC)
Gary McMillan
1406 Smith Rd.
Austin, TX 78721

Intel Scientific Computers
Justin Rattner
Cornet Oaks 1
15201 N.W. Greenbrier Pkwy
Beaverton, OR 97006-5771
M.S. CO1-07

Metaflow
Bruce D. Ughtner
6725 Mesa Ridge Rd. #100
San Diego, CA 92121

Philips Research
Sunnyvale Department
Gerri A. Slavenbufg
440 Wolfe Ave.
Sunnyvale, CA 94088

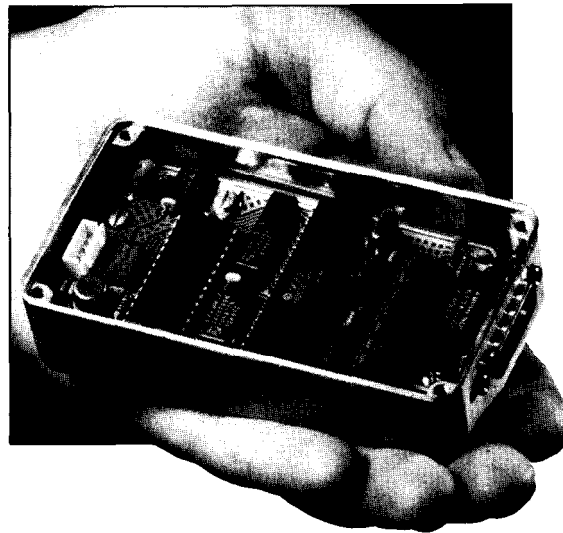
Rafael Saavedra-Bafrera
Computer Science Division
U.C. Berkeley
Berkeley, CA 94720

IBM
Richard Cehlef
Martin Hopkins
T.J. Watson Research Cntr
Hwy. 134E
P.O. Box 218
Yorktown Heights, NY 10598

Tom Cantrell holds a B.S. in economics and an M.B.A. from UCLA. He owns and operates Microfuture, Inc., and has been in Silicon Valley for ten years involved in chip, board, and system design and marketing.

IRS

428 Very Useful
429 Moderately Useful
430 Not Useful



Intelligent I/O

at your fi'ngertips!

TETHERLINE

allows you to control and monitor nearly any device or mix of equipment by putting **full** computing power at each node.

One low voltage **TETHERLINE** supports up to 127 **TETHERBOXes**, each capable of 7 I/O Yts. The central controller plugs into a standard PC expansion slot and handles timing, self-diagnostics, and communications — leaving the PC free for other uses.

Features:

- ✓ **RS-422/485** serial communications
- ✓ **TETHERBOX** program stored in EEPROM, easily changed and Upgraded over the **TETHERLINE**
- ✓ **TETHERBOX** supplied in die-cast aluminum cases with gold plated connectors **for** harsh environments
- ✓ Internal **opto-isolation** ensures clean signals in electrically noisy environments
- ✓ **Watchdog circuitry**, on-board regulation and power tail detection in every TETHWBOX
- ✓ **X-10*** two-way **communications** supported
*TM of X-10 USA, Inc.
- ✓ Compact, **affordable** devices

For details, call or write:

INFINICO inc.

P.O. BOX 1280 • Vestal, NY 13851-1280
(607) 798-9700 • Fax (607) 729-1364

PRACTICAL ALGORITHMS

Scoff Robert Ladd

Making Hash

A Table Built for Speed

A program can use a tree structure that stores information in **some** orderly fashion. Trees are commonly used for organizing databases of records that are accessed by key fields. The key information consists of the actual key data and a number that pinpoints the location of the data associated with that key. For example, a key value might be a person's name, and the number associated with that key would tell which record in a data file contains that person's data. A key is stored in the tree by comparing it to the values of other keys stored in the tree. To locate the record associated with a key, one merely needs to look up the key and extract the associated record number.

Trees are complex data structures, and there are limits on their efficiency. Trees also use lots of memory to store the keys and record numbers. The advantage of a tree indexing system is that records can be accessed in a sequential order based on the organization of the keys in the tree. In the earlier example, the tree would be set up to alphabetically store keys containing the name of a person. The tree could then be used to retrieve the records of people based on the alphabetical order of their names.

Often, a program doesn't need to retrieve keys in a specific order. For example, a dictionary program merely needs to look up a word in a database; if the word is found, its definition can be retrieved from a data file and displayed. For this kind of program, tree indexes can use too much memory and be too slow. A better approach is to use a hash table.

DEFINING A HASH TABLE

The term "hash table" may bring up visions of a cutting block covered with chopped corned beef and potatoes. As a computer term, hash table refers to an array of data that is indexed via a hash function. A hash function converts a key value into a numeric index into the hash table. Each entry in the hash table is a linked list of keys (and associated information) that were converted by the hash function into the same index. If the hash function distributes the keys widely enough in the table, there will be only a few items in each linked list to search. For example, if we have a 200-entry hash table in which 1000 keys are stored, there should be an average of five key values in each linked list. Calling the hash function will

return the index value for a given key, and then the appropriate linked list can be searched for the specific key.

To convert a key into a hash table index, the hash function must be able to treat the key as a numeric value. If the key is already a number, the problem is solved. If the key is a character string (and it usually is), the hash function will have to produce a numeric value from the characters. The simplest method of accomplishing this is to multiply the numeric values of the characters to produce

```
MODULE Hash;
FROM ASCII
  IMPORT nul, ht;
FROM InOut
  IMPORT Done, EOL, WriteString, WriteLn,
  WriteCard, Write, Read;
FROM Storage
  IMPORT ALLOCATE, DEALLOCATE, Available;
FROM Strings
  IMPORT Assign, CompareStr;
FROM SYSTEM
  IMPORT TSIZE;
CONST
  HashPrime = 383; (* bins in hash table *)
  MaxLength = 31; (* max string len - 1 *)
TYPE
  ListNodePtr = POINTER TO ListNode;
  WordBuffer = ARRAY [0..MaxLength] OF CHAR;
  ListNode = RECORD
    LinkPrev : ListNodePtr;
    LinkNext : ListNodePtr;
    Text      : WordBuffer;
    Count     : CARDINAL;
  END;
  HashRec = RECORD
    NodeHead : ListNodePtr;
    NodeTail : ListNodePtr;
  END;
VAR
  HashTable : ARRAY [0..HashPrime-1] OF HashRec;
  InputBuf  : WordBuffer;
PROCEDURE ClearTable;
VAR
  I : CARDINAL;
BEGIN
  FOR I := 0 TO HashPrime - 1 DO
    HashTable[I].NodeHead := NIL;
  (continued)
```

listing 1 -Sample code shows the techniques involved in using hash tables.

a number. Using an unsigned 16-bit integer and ignoring overflow (i.e., letting the value wrap around when it exceeds 65,535), a number can be calculated for any string.

A hash table will not generally contain as many entries as there are possible numbers. For example, a hash table index with 65,536 entries would use copious amounts of memory and would have many empty entries. So, a hash table has a fixed size, and the actual index of a given key is the remainder of dividing the numeric key value by the size of the table. For example, a hash function for a table with 200 entries would calculate a table index of 156 for a key with a numeric value of 4356.

For various arcane and mathematical reasons-to be explained in a future column-the best size for a hash table is a prime number. A prime number that is not close to a power of two works even better. So, for example, a table size of 383 (midway between 256 and 512) would work very well for a moderately sized table.

AN EXAMPLE

The best way to see how a hash table works is to examine a working program. Listing 1 shows the `Module 2` program `HASH`. The program reads text from the standard input device, breaks it into words, and stores the words in a hash table using the words themselves as the keys. With each word is a `CARDINAL` (unsigned 16-bit) value that contains the number of occurrences of the word in the text. When the program is done, it displays a formatted version of the hash table. The prime number 383 was chosen for the size of the table; the constant `HashPrime` holds this number. The table will hold string of up to `MaxLength` (in this case, 31 characters) in size.

`ListNode` is a type which defines the nodes in a doubly linked list. Each entry in the `HashTable` array is a `HashRec` containing pointers to the first and last nodes in the linked list for that hash index value. `WordBuffer` is a character string type that will hold words as they are read.

The `ClearTable` function sets the pointers in `HashTable` to `NIL` to indicate that there are no entries in any of the linked lists. `GetWord` reads words from the input file, and stores them in the `WordBuffer` supplied as a parameter. When end-of-file is reached, `GetWord` returns `FALSE`; otherwise, it returns `TRUE`.

`HashFunction` accepts a `WordBuffer` parameter and returns a hash table index. A preliminary value is calculated by multiplying the values of the characters in the string together. It then returns the remainder of dividing the preliminary value by `HashPrime`.

`InsertKey` has a `WordBuffer` parameter for the key to be stored in the hash table. It first calls `HashFunction` to calculate an index value for the key parameter. Then, the list is searched to find an existing entry for the key. If the key is found, the `Count` value for that node is incremented and `InsertKey` returns. Otherwise, `InsertKey` creates a new linked list node. If the list for the key's index is empty, the new node becomes the head of the list. Otherwise, it is placed at the end of the existing linked list.

```

    HashTable[Index].NodeTail := NIL;
END
END ClearTable;

PROCEDURE HashFunction(Key:WordBuffer):CARDINAL;
VAR
    HashIndex, I : CARDINAL;
BEGIN
    HashIndex := 1;
    I := 0;
    WHILE Key[I] # nul DO
        (*NOCHECK:O*)
        HashIndex := HashIndex * ORD(Key[I]);
        INC (I)
    END;
    RETURN HashIndex MOD HashPrime
END HashFunction;

PROCEDURE GetWord(VAR Key:WordBuffer):BOOLEAN;
VAR
    I : CARDINAL;
    Ch : CHAR;
BEGIN
    I := 0;
    LOOP
        Read (Ch);
        IF NOT Done THEN
            RETURN FALSE
        END;
        IF (I = MaxLength) OR (Ch = EOL)
            OR (Ch = '') THEN
            Key[I] := nul;
            RETURN TRUE;
        ELSIF (CAP(Ch) >= 'A')
            AND (CAP(Ch) <= 'Z') THEN
            Key[I] := Ch;
            INC (I)
        END
    END
END GetWord;

PROCEDURE InsertKey(Key: WordBuffer);
VAR
    WorkNode : ListNodePtr;
    Index : CARDINAL;
BEGIN
    Index := HashFunction(Key);
    WorkNode := HashTable[Index].NodeHead;
    WHILE WorkNode # NIL DO
        IF 0 = CompareStr(WorkNode^.Text, Key) THEN
            INC(WorkNode^.Count);
            RETURN
        END;
        WorkNode := WorkNode^.LinkNext
    END;
    ALLOCATE (WorkNode, TSIZE (ListNode));
    Assign (Key, WorkNode^.Text);
    WorkNode^.LinkNext := NIL;
    WorkNode^.Count := 1;
    IF HashTable[Index].NodeTail = NPL THEN
        WorkNode^.LinkPrev := NIL;
        HashTable[Index].NodeHead := WorkNode;
        HashTable[Index].NodeTail := WorkNode;
    ELSE
        HashTable[Index].NodeTail^.LinkNext :=
            WorkNode;
        WorkNode^.LinkPrev :=
            HashTable[Index].NodeTail;
        HashTable[Index].NodeTail := WorkNode
    END
END InsertKey;

PROCEDURE DisplayTable;
VAR
    WorkNode : ListNodePtr;
    Index : CARDINAL;
BEGIN
    FOR Index := 0 TO HashPrime - 1 DO
        WriteString('Index ');
        WriteCard(Index, 0);
        WriteLn;
    END
END

```

(continued)

Listing 1 — continued

SUPERVISION/8 VIDEO IMAGE GRABBER



\$269.95 Std. Res. 256 x 244
\$369.95 High Res. 512 x 488

Supervision/8 is an adapter with software, the latest in economically capturing high quality real world images with your computer.

- Half Size Cards
- Capture Time 1/60 second
- PC/XT/AT Compatible
- MC/Visa, AM, Express
- 256 Grey Levels — 8 Bit
- TIFF and PCX Format
- Binary Image File (BIF)
- RS-170 Video Source

IDEC, INC.

119s Doylestown Pike
Quakertown, PA 18951

TEL: 215-538-2600

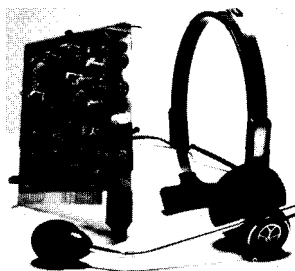
FAX: 215-538-2665

Reader Service #149

TALK TO YOUR COMPUTER

WITH VOICE MASTER KEY® FOR PCs/COMPATIBLES
VOICE RECOGNITION WITH SPEECH RESPONSE

GIVE A NEW DIMENSION TO PERSONAL COMPUTING The amazing Voice Master Key System adds voice recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, DTP, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. A real productivity enhancer!



SPEECH RECORDING SOFTWARE
Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files, voice memos, more. Send voice mail through LANs or modem. A superior speech/sound development tool.

INTERACTIVE SPEECH INPUT/OUTPUT
Tag your own digitized speech files to voice recognition macros. Provides speech response to your spoken commands -- all from within your application software! Ideal for business, presentation, education, or entertainment programs you currently use.

Augment the system for wireless uses in robotics, factory process controls, home automation, new products, etc. Voice Master Key System does it all!

EVERYTHING INCLUDED Voice Master Key System consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. High quality throughout, easy and fun to use.

ONLY \$149.95 COMPLETE

ORDER HOTLINE: (503) 342-1271 Monday-Friday 8 AM to 5 PM Pacific Time. VISA/MasterCard phone or FAX orders accepted. No CODs. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3 1/2" or 5 1/4") when ordering. Add \$5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes.

30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED.

CALL OR WRITE FOR FREE PRODUCT CATALOG.



COVOX INC.
675 CONGER ST.
EUGENE, OR 97402

TEL: (503) 342-1271
FAX: (503) 342-1283

Reader Service #131

```

WorkNode := HashTable[Index].NodeHead;
WHILE WorkNode # NIL DO
  Write(ht);
  WriteString(WorkNode^.Text);
  WriteString(' : ');
  WriteCard(WorkNode^.Count, 0);
  WriteLn;
  WorkNode := WorkNode^.LinkNext;
END;
WriteLn
END
END DisplayTable;

PROCEDURE DeleteTable;
VAR
  Next : ListNodePtr;
  WorkNode : ListNodePtr;
  Index : CARDINAL;
BEGIN
  FOR Index := 0 TO HashPrime - 1 DO
    WorkNode := HashTable[Index].NodeHead;
    WHILE WorkNode # NIL DO
      Next := WorkNode^.LinkNext;
      DEALLOCATE(WorkNode, TSIZE(ListNode));
      WorkNode := Next;
    END
  END
END DeleteTable;

BEGIN
  ClearTable;
  WHILE GetWord(InputBuf) DO
    InsertKey(InputBuf)
  END;
  DisplayTable;
  DeleteTable;
END Hash.

```

Listing 1 -continued

The main program calls `ClearTable`, and then begins a loop. Each pass through the loop, `GetWord` is called; if it was successful, `InsertKey` is called with the newly read word. When `GetWord` returns `FALSE` (to indicate the end of the input file), the loop ends.

I experimented with this program and a short science fiction story I was writing. At the time, the story contained 2500 words; the average `HashTable` entry contained a linked list with six words in it. Some `HashTable` entries were empty; others had as many as a dozen words. However, most nodes contained between four and eight words. The longest search for a word would require stepping through 12 entries in a linked list.

WRAPPING UP

I appreciate the patience shown by the folks at `CIRCUIT CELLAR INK` during my recent hiatus. As always, I look forward to hearing from the readers of this column. In the next issue, I'll round out the discussion of hash tables by improving efficiency and the distribution of keys. Until next time.. ❖

Scott Ladd is a writer specializing in computer software. Correspondence concerning "Practical Algorithms" may be sent to him at: Scott Robert Ladd, 705 W. Virginia, Gunnison, CO 81230, (303) 641-6438.

IRS

431 Very Useful
432 Moderately Useful
433 Not Useful

CEBus Gets Physical

The Standard Takes Two More Steps to Maturity

Domestic Automation

Ken Davidson

Well, I'm finally on the right EIA mailing list and within the last few weeks received copies of the proposed specifications for the CEBus SR and TP physical layers. As you'll recall from my past articles detailing CEBus (issues #10 and #15 of *CIRCUIT CELLAR INK*), SRbus is the infrared physical layer and TPbus is twisted pair. I'll be covering these proposals in more detail in a future issue, but I wanted to touch on the high points here.

The IR spec is written primarily for one- and two-way control, presumably using the ever-popular hand-held remote. While each CEBus physical layer defines a separate control channel and data channel, this first draft only addresses the control aspect (like the power line). Signals are encoded using the same four symbols as defined for the power line (i.e., 1, 0, End Of Frame, and End Of Packet), and the unit symbol time is defined as 100 microseconds, so the maximum data rate for IR is 10,000 one-bits per second. The superior state is defined as the presence of a 100-kHz subcarrier (as opposed to the 40-kHz often used in today's remotes) modulated on an IR wavelength in the range of 850 to 1000 nm. The inferior state is simply the lack of a subcarrier. (Note the use of 100 kHz as opposed to 40 kHz was likely due to the desire to push data rates up to 10,000 bps to be consistent with other CEBus physical layers. Use of 40 kHz would have resulted in only four cycles of carrier denoting a "1" bit instead of ten cycles with 100 kHz.)

One of the problems inherent in the one-way control section is the lack of adherence to the CSMA protocol upon which all of CEBus is based. There is no way for the device to "listen" to the medium to find out if someone else is transmitting and no way to know if the transmission was properly received at the destination. However, this mode will likely be used in inexpensive hand-held remotes for same-room control where it will be obvious when the command wasn't received (e.g., the TV didn't really come on, so we'd better press the button again).

The proposed TP spec is much more interesting (how much can you say about an IR transmission?). The spec sets forth four twisted pairs in a full-blown CEBus implementation. The first pair, TP0, carries a control channel, 14 data channels, and power for connected devices. The second, third, and fourth pairs (TP1, TP2, and TP3) each carry 16 data channels. Optionally, TP2 and TP3 may carry

conventional telephone signals so CEBus and non-CEBus devices may be mixed in the same system.

Each twisted pair potentially carries signals in a frequency band from 0 to 512 kHz, which is broken into distinct channels modulated into the spectrum. Each channel on a twisted pair consists of a 10-kHz piece of bandwidth separated from adjacent channels by 32 kHz (i.e., 10 kHz of bandwidth with an 11-kHz guard band on each side). Since channel 0 starts at DC, it has the luxury of 21 kHz of bandwidth.

The method of control channel signaling is confusing at first, so I hope my explanation is clear enough. Like most of the other physical layers, the unit symbol time is 100 μ s, giving a data rate of 10,000 one-bits per second. The signal swings around the average DC supply voltage (V_{PS}) present on the pair by ± 250 mV and can take on one of three values: $V_{PS}+250$ mV, V_{PS} , and $V_{PS}-250$ mV. A superior state is designated by a step from one of the above values to the next at least every 100 μ s. An inferior state is denoted by the lack of such a transition.

Let's do an example. To denote a "0" bit (which lasts two unit symbol times, or 200 μ s) with a superior state, and we happen to already be at $V_{PS}+250$ mV, the signal would step to V_{PS} at the start of the symbol, then 100 μ s later would step to $V_{PS}-250$ mV, and finally the signal would step back to V_{PS} at the end of the symbol 100 μ s after that. The signal would stay at V_{PS} for the duration of the next symbol (which depends on what the symbol is) since we always alternate between the superior and inferior states. Remember that it is always the duration of the superior or inferior state, and not which state is used, that determines which symbol is being sent.

If you're still hopelessly confused, I'll be explaining things in more detail in an upcoming full-length article.

Finally, I want to mention connectors. At present, the spec only covers connectors for telco-only and mixed telco/CEBus setups. The CEBus-only connector has yet to be determined. The telco-only connector is, obviously, standard 6-pin modular connector (often containing just two or four wires). The mixed telco/CEBus connector is the similar 8-pin modular connector, which has one telephone pair on the center conductors, the next telephone pair on the pair surrounding those, and finally TP0 and TP1 on either side of those (though not straddling them).

Such a setup would prevent CEBus-compatible devices from being plugged into telco-only jacks (since the jack would be too narrow for the 8-pin plug), but would allow regular telephones with 6-pin plugs to be plugged into the 8-pin jack and operate normally.

So if you happen to be building a house right now and haven't covered up those walls yet, you might want to run some four-pair (24-gauge) wire to each room so you're ready for CEBus devices once they hit the market.

By the time you read this, the comment periods for both proposed specs will be over, but you still might want to contact EIA Literature Sales to get your own copies and perhaps send along comments to EIA. Even though the official comment periods are over, the committees will still be doing fine-tuning work.

REMOTE POSSIBILITIES

Not long after my article describing how to control X-10 devices using a hand-held trainable IR remote (CIRCUIT CELLAR INK, issue #9), I discovered the IR remote that prompted the design of the X-10 IR gateway controller in the first place. Available from Universal Electronics, the One For All is different from most universal hand-held remotes. Rather than training this remote using the original units that come with each piece of electronics, it has all the codes built into it already.

Universal has amassed a huge database of virtually every remote control unit ever produced and includes codes for the most popular devices in the unit when you purchase it. Find your piece of equipment in the book included with the One For All, punch in the code number for that equipment, and you can immediately start using the remote. Has the original hand-held remote for some piece of equipment been lost or broken? You don't need it since the codes are already built in. Does your cable company charge some ridiculous rental fee each month for the privilege of having a hand-held remote for their box (like ours does)? Chances are very good that the IR codes for that cable box are already in the One For All, so you can legally avoid that extra fee without getting out of your easy chair.

What happens when new devices are introduced? Don't throw the One For All away. Simply take the cover off the battery compartment, plug in a three-pin serial cable, plug the other end into your IBM PC compatible, and load up the remote with a new set of devices. Universal maintains a bulletin board system with the latest copy of their database where computer users can call to download updates.

This last feature is what I find to be most exciting. For Joe Average Consumer who finds it difficult to adjust the volume on his TV, it's not likely to be of much use, and a network of dealers with PCs and the update software is being assembled so that remotes owned by such consumers can be updated. But for those with at least some technical competence and an interest in home control, it leads to some interesting possibilities.

Part of the command set used to update the remote's memory allows the computer to remotely "press" the buttons on the controller. With minimal programming it's possible to control virtually every device in the house which has IR remote capabilities from a single computer. Now, not only can you turn a light on in the room when someone walks in, you can also automatically turn on the TV and select a particular channel depending on what day and what time it is without any extra fancy electronics.

The updatable memory also allows the One For All to be ready to work with CEBus. In talking with the people at Universal Electronics, the current One For All units should work just fine with the proposed IR standard I described above. Once finalized, owners will be able to call the BBS, download a database that includes the CEBus codes, and instantly update their hardware to work with the new standard. With the addition of a gateway to usher the commands from the IR domain to some other medium (probably power line at first), we can control virtually all CEBus-compatible devices by barely lifting a finger.

I've seen the One For All (and its little brother, the One For All II) available from numerous sources including local discount stores as well as the usual mail order companies oriented toward home automation, Universal Electronics also sells the X-10 IR gateway mentioned above. It allows you to use the One For All to control X-10 devices in the same way as with any X-10 controller, but without the wires. It's great fun to be able to select a television channel, adjust the volume, and dim the lights all with the same hand-held unit while sitting on the living room floor.

On a similar note, Memorex has just picked up the One For All technology to replace their line of trainable remotes, and since Memorex supplies Radio Shack with their universal remotes, we'll start to see these showing up in Radio Shacks across the country before long. (Note that the One For All III and the four-device remote available now from Radio Shack use similar technology but different guts and aren't reprogrammable, so be careful when buying one of these if updatability is important to you. When in doubt, lift the battery compartment cover and check for three holes in the printed circuit board just above the batteries. If they're not there, the unit has all its codes in ROM and can't be updated.)

Electronic Industries Assoc.
2001 Pennsylvania Ave.
Washington, DC 20006
(202) 4574975

Universal Electronics, Inc.
16308 South Sunkist
Anaheim, CA 92806
(714) 939-7823

Ken Davidson is the managing editor and a member of the Circuit Cellar INK engineering staff. He holds a B.S. in computer engineering and an MS. in computer science from Rensselaer Polytechnic Institute.

IRS

434 Very Useful
435 Moderately Useful
436 Not Useful

Conducted by
Ken Davidson

Excerpts from the Circuit Cellar BBS

The Circuit Cellar BBS
300/1200/2400 bps
24 ho&s/7 days a week
(203) 871-1988
Four Incoming Lines
Vernon, Connecticut

The Circuit Cellar BBS joined the '90s recently with the addition of a 9600-bps modem fo one of its incoming phone lines. The number (203) 871-0549 originally had a 1200-bps modem connected to alleviate some of the traffic on the modems connected to the main phone number. We just added a USRobotics Courier HST modem to that number, so users with HST modems can call in at 9600 bps. Note that neither Hayes nor V.32/V.42bis modems will work on this line and all normal time and download limits remain in place for the time being.

We'll start off this installment with a discussion involving building a solid-state answering machine (something I've always wanted to do) and will follow with a brief foray into soldering to steel.

Msg#:32581

From: AL DORMAN To: ALL USERS

How do they do it? I hear that there are answering machines out there that have no tapes in them and are operated with "speech recorder" chips that store everything in DRAM or SRAM. I have looked at the OK1 MSM6388 but at a 4-kHz sample rate it only gives four minutes of recording. And 4 kHz seems a little slow.

The TI TMS3477 samples at 16-64 kHz but will only address up to four megabytes of DRAM which gives about 32 seconds of message. TI says they are working on a TMS3478 that will work off of SRAM or EPROM but I cannot get any info about how long it "messages." Is there another company with a chip set out there that will access more memory for longer messages? I would appreciate any help in locating these chips. I need about 10 minutes of recording time.

Msg#:32655

From: STEVE CIARCIA To: AL DORMAN

It's called ADPCM. I did an article on it a while back including a project on how to build one. It's in volume 4 of the Circuit Cellar books.

Msg#:32661

From: AL DORMAN To: STEVE CIARCIA

Sure ADPCM... most of the chip manufacturers I have contacted use this method while a few do a straight A-to-D then D-to-A (PCM). I am looking for a chip that will address enough DRAM or SRAM to provide 3-10 minutes of speech at a 32-kHz sample rate. TI can do it with an add-on processor but a blurb on my answering machine today from NEC says they have a stand-alone chip that will handle the whole job. Fax to follow Monday. There is one commercial answering machine on the market but it only handles five minutes of messages. This is turning into an interesting project.

Msg#:32678

From: DONALD YUNISKIS To: AL DORMAN

Unless your "speech" is something like music, the 32-kHz sampling rate is overkill. Cut it back to 8-10k and triple/quadruple your playing time!

Msg#:32723

From: AL DORMAN To: DONALD YUNISKIS

Actually it is a music background with speech superimposed on top of it but it is being played over the phone lines. I was hoping for 32k because the customer (a musician) will find fault with the music quality before he complains about the speech.

Msg#:32729

From: BOB PADDOCK To: AL DORMAN

Try looking a chips that use CVSD instead of ADPCM. It is simpler and you can use a lower bit rate. The Harris 55564 works well with an SPI interface, or the MX*COM 690. The MX*COM 790 works well on an 8-bit bus.

MX*COM, Inc.
4800 Bethania Station Rd.
Winston-Salem, NC 27105-1201
(919) 7446050 or (800) 638-5577
Fax: (919) 7445054

From the front of the data book: ASICs for Continuous Tone Signaling, Sequential Tone Signaling, Secure Speech, Voice Stor-

age (the 790 chip I was talking about), Data Modems, Switched Capacitor Filters.. ..

They make the parts for paging companies (at the transmit end, not the end you carry around with you).

Msg#:32819

From: DONALD YUNISKIS To: BOB PADDOCK

A word of caution though: speech digitizing tends to leave you with lots of white noise—especially the CVSDs-if you don't properly use the dynamic range of the coder.. .

Msg#:32739

From: BURT BROWN To: AL DORMAN

I've built a couple of digital audio record/play cards for the IBM PC/AT and you can get pretty decent sounding music with a 20-25-kHz sample rate. An FM broadcast radio is only about 15 kHz and your phone line tops out at 35004000 Hz. As was previously mentioned, anything over 8-10k is just overkill. If the background (music) portion of the message doesn't change too often, why not put that part on tape and use ADPCM for the voice?

Msg#:32761

From: AL DORMAN To: BURT BROWN

That would be a simple solution but "they" want a single black box solution to the whole problem.

Msg#:33273

From: MATTHEW TAYLOR To: AL DORMAN

I'm just curious: What are you sampling? If it is voice, you will save yourself a lot of money by going to a lower sample rate (6-7 kHz).

Msg#:33310

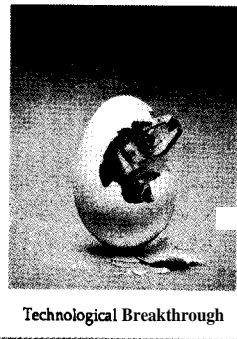
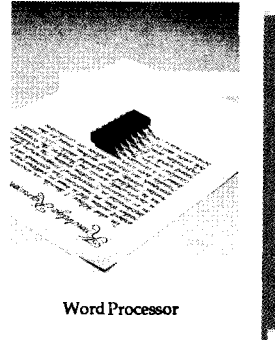
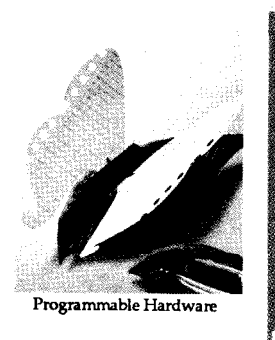
From: AL DORMAN To: MATTHEW TAYLOR

I need to "record" background music with a voice-a commercial. I actually found a chip (NEC 77501) that will record up to 15 minutes of speech at 32k. No special memory bank selecting necessary, the chip does it all to a row of DRAMs through an SRAM. It takes 16 MB of audio quality DRAM (\$7.50 ea) at 1 MB per two 4-bit chips. At a sample rate of 8k to 16k this chip can record up to 25 minutes of speech.

Most of us are lucky that we never have to leave the realm of gold and copper and rosin-core tin solder when making connections. The world of acid-core solder and dissimilar metals can be nasty, as the next few messages show.

Limited Editions

#G \$60 #H \$60



#K \$60 #C \$80

Circuit Cellar Ink cover artist Robert Tinney proudly offers these distinctive 16" x 20" Limited Edition Prints. Each is an exquisite reproduction from the pages of Byte Magazine, and is part of an edition of only 1000 prints. The museum quality stock is acid free, ensuring brilliance and durability for decades to come. The artist personally inspects, signs and numbers each print, which is accompanied by its own Certificate of Authenticity.

Order your prints beautifully triple-matted and Framed! The frames are of the silver metal variety, and mats are chosen to complement the colors of the print(s) you order. Plexiglass only.

The price of each print is shown at left. Order two or more and deduct 15%! Frames are only \$39.50 each. For shipping, add \$5 per order for unframed prints (\$25 overseas); for framed prints, add \$6.50 for one print and \$4.50 for each extra print (ground).

No frames shipped overseas. Full refund if not satisfied. For VISA, MasterCard or AMEX orders call 1-318-826-3003.

ORDER FORM				cci
Qty.	#	Title	Amount	
			\$	
			\$	
			\$	
			\$	
If you order TWO or more deduct 15%			\$	
Frames (\$39.50 each)			\$	
Shipping charges: See above.			\$	
Total			\$	
<input type="checkbox"/> I have enclosed a check or money order to Robert Tinney Graphics. <small>(Must be drawn on a U.S. bank; no foreign collection, please.)</small>				
Bill my <input type="checkbox"/> VISA <input type="checkbox"/> MasterCard <input type="checkbox"/> American Express				
Card #:			Expires: _____	
Name: _____				
Address: _____				
City: _____			State: _____ zip: _____	
Country: _____				
<input type="checkbox"/> Send a brochure showing your other prints.				
ROBERT TINNEY GRAPHICS P.O. Box 778 Washington, LA 7058C				

Msg#:32529

From: FRANK HENRIQUEZ To: ALL USERS

I have an ultra-thin coax cable that I have to solder to two connectors. The center conductor and shield are both made out of steel. (Why? Because steel has a good electrical conductivity compared to a low thermal conductivity-this cable is going inside a Dewar). How do I solder this stuff? I don't want to crimp the wires, and I really don't want to weld it to the connectors. I'm also worried about acid-based fluxes and solder pastes. Any ideas?

Msg#:32629

From: PETE KOZIAR To: FRANK HENRIQUEZ

I don't know how, but a few years ago, my father demonstrated soldering steel with ordinary solder and special flux. I don't think it was an acid flux, but I'm not sure.

Check out a well-stocked hardware (yes, hardware) store. Some of them have a rather amazing array of solders and solder-like "stuff" for various temperatures and materials.

Msg#:32637

From: FRANK HENRIQUEZ To: PETE KOZIAR

I ended up using an acid flux. It was a horrible, HCL-based flux, but it worked like a charm, and cleanup was fairly easy. Hopefully I won't have to do it again for a long time. Thanks

Msg#:32697

From: PELLERVO KASKINEN To: FRANK HENRIQUEZ

By now, you have done the job, but just for the benefit of the "general audience," here is some more about the topic.

The normal ways in industry can be found on dual fronts: the manufacturing of solder-plated sheet metal and the piece part fabrication. I do not know about the first one, but suspect a highly acidic flux. The second area relies on two processes. The first one

is to use ultrasonic energy in the solder bath to promote the adhesion. The second one uses an electroplate process. Once you have a thin layer of either copper or tin over the steel, then the rest can be easily done with ordinary hot dipping.

Now, about the electroplating. For hobby use, the handiest way is to get a "pen" from Hunter Tools. These pens are like ordinary (hard) tip felt-tip pens, with about a 0.5-inch diameter and a metal contact at the opposite end. You apply 5 to 8 volts DC there and the other pole you hook to the part to be plated. Then you just rub the pen to your part and it starts building a layer of the plating. I have a few of these pens (copper, nickel, gold, and rhodium to be precise) and they work fine, but are expensive for more than very occasional use.

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

437 Very Useful
438 Moderately Useful
439 Not Useful

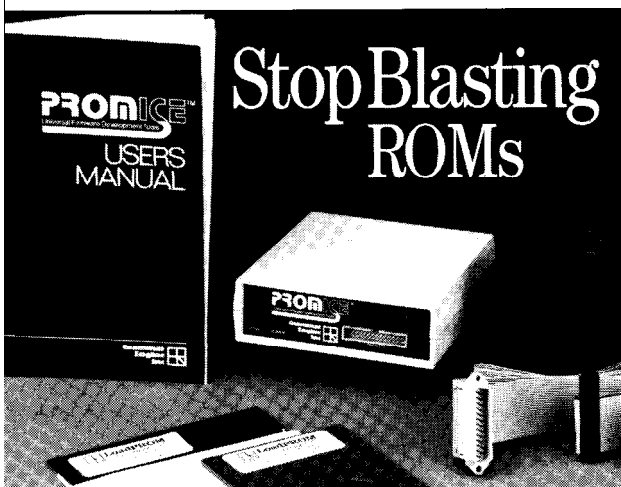
SOFTWARE and BBS AVAILABLE on DISK

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 two 360K, 5.25" IBM PC-format disks for only \$15.

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 1991) includes all public messages posted during November and December, 1990.

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 (233) 875-2199. Be sure to specify the issue number of each disk you order.



Stop Blasting ROMs

PROMICE emulates 8 bit ROMs from 2716-27080, or 16 bit ROMs 27C1024 or 27C2048. (Inquire about emulating other ROMs. Non-JEDEC ROMs require custom cable.) ■ Sophisticated LoadICE™ Host Software downloads, uploads and edits ROM contents, supports MS-DOS, UNIX, MAC & VMS. Software sources are included.

- Bi-directional Serial link, autobaud to 57.6KB—loads 1 Mbit in 25 secs.
- Bi-directional Parallel port (option)—loads 1 Mbit in 4 sec.
- Emulate up to 2 ROMs per unit, daisy-chain up to 256 ROMs from one port!
- New! Analysis Interface™ (option) implements a ROM-based UART for sophisticated debugging.

Grammar Engine Inc

1161 Cherry Street
San Carlos
California 94070
415/595-2252

Order Service #144

STEVE'S OWN INK

Steve Clarcia

The Sophomore Slump

The telephone calls seem to all follow the **same script**: "I'm an engineer with the Incredibly Large Corp., and I've got an idea for a wunderwidget that I think will sell pretty well. I've been working on the design in my spare time. I don't want to quit my day job right now, but with the economy the way it is, I'd like to have the option of going with my own company in a couple of years.

"Can you tell me how to get started in the manufacturing end of the business?"

I can tell them about working with board and assembly houses, but that's information that they should have gotten from reading the trade magazines. Furthermore, if all they want is a computer-related hobby that (sort of) pays for itself, then they don't need any more advice from me. If, on the other hand, they really want to start a company, then they should know that they've done the easy stuff. The hardest part—the second product—is yet to come.

When you design any product, your costs can be divided into two large categories: design and production. For someone who calls me following the script above, the design part is, in their mind, free. They generally have worked on the design "on their own time" and treated it like a hobby. When they think about actually selling the product, the only costs they think about are out-of-pocket expenses. Now, there's nothing wrong with this approach for a first product, and it's the only allocation of costs that allows many small businesses to get off the ground. Since nothing is (or very few things are) truly free, this approach does have a cost attached, and the cost is inflexible pricing.

Look at it this way: When you set the price of something, you look at what it costs you to make the product, then **you** add something for a profit. If you have valued the development time at zero, then you see a rather low cost of product. Since you want to have your name known, you don't add a lot of profit, you advertise and start selling the product. Everything runs fine unless a Large Multinational Corporation comes along and asks for 20,000 units at an appropriate discount. At that moment, you realize that the only way to give them a large discount is to lose money. That's when you vow not to make the same mistake on your second product.

Some companies price every product like the first product I described. They don't last. Other companies

realize that they should shift some of their costs to development, looking at options that increase their development costs in order to reduce the production costs. You can see extreme examples of this in many consumer electronic products. If you take a small camcorder apart (not that I'm really recommending that you do) you'll find a lot of "expensive" technology. Most of the chips are custom or semicustom devices. The components are surface-mounted to the PC board. The board itself may well be flexible, forming itself to the outline of the case. In each of these examples, the engineering and setup costs were maximized so that the production cost can be minimized. Initially, the price is set rather high, but the price drops with time and sales volume increases. Why is that so important?

Let's look back at the first example. If you figure in the development costs, your cost of product, and therefore your price, will be higher, but with every sale you "pay back" some of that development cost. At a certain point, your development costs have been repaid and you have two options: accept increased profits (what a burden) or lower your price and increase your market share. Furthermore, if the Large Multinational Corporation asks for 10,000 units, you've built much more flexibility into your pricing.

Starting a company isn't easy, and there are many more things to consider than what I've talked about here. It's important, though, to know that there is more than one way to allocate your costs and set your prices. It all depends on whether you want to sell 100 or 1,000,000 of your product, and whether you can afford to pay the costs at the beginning of the product cycle. Whatever decisions you make, I hope that 1991 is a successful year for you.

