CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

DATA ACQUISITION

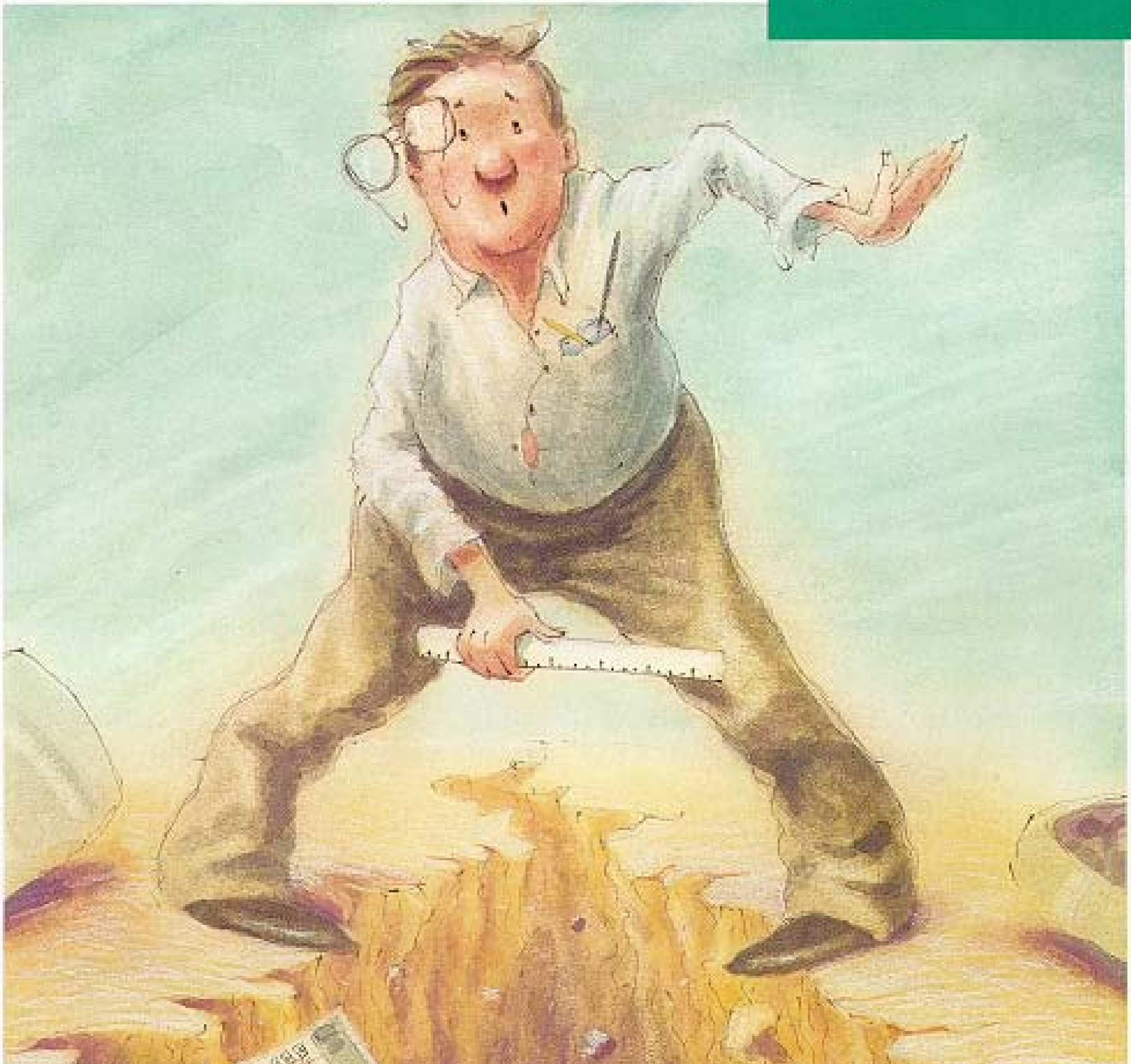
Replace a Strip-chart Recorder with a Dot-matrix Printer

The précis 100-kHz, 16-bit A/D Converter

Software Patent and Copyright Issues

80x86 Protected-mode Programming

August 1994 — Issue #49



$3.95 U.S.
$4.95 Canada

EDITOR'S INK

The Ultimate Data Collection Machine

Have you ever thought about the vast amounts of data the human brain collects and processes each day? The raw data coming from your eyes alone would overwhelm even the fastest computer available today with gigabytes of storage. Consider the most mundane daily tasks of recognizing objects on a table, listening to and understanding the spoken word (or even simple telephone call progress tones), using just the right amount of pressure to grasp a delicate object without crushing it, or picking out the subtle aroma of burning potpourri from a roomful of other smells.

Given the wonders of the human body, we have a remarkable amount of work left to do to even approximate a single human sensory processing system. Going back to my sight example, think about the work involved in trying to get a computer with an attached video camera to simply recognize that a human face (*any* face) is somewhere in its field of view. I contrast that with being able to stare at my newborn daughter's face and compare her features with those of her older sister. The lips are the same, but the nose isn't quite as upturned and the shape of her head is different. The person who can make a computer do that will be very rich indeed.

Returning to the world of the practical, our first data acquisition feature article this month considers the ubiquitous laboratory strip-chart recorder. While simple in concept, it can be expensive. By applying some much cheaper off-the-shelf hardware and some code, we can make a dot-matrix printer do much of the same work.

Next, for those who want to collect data so fast it taxes the capabilities of today's best desktop machine, we present précis: a 1 00-kHz,16-bit A/D converter board for the ISA bus. Along with covering the details about the board itself, we also show how the précis board was applied to calibrating some seismic sensors.

Finally, following up on an article we carried a few months ago on ownership of work, we look at the current debate raging over copyrights and patents for software. The legal world has a lot of catching up to do.

In our columns, Ed starts a series of articles exploring the somewhat scary world of protected mode programming; Jeff checks out the current state of low-cost voice recognition hardware (it's still not even close to human standards); Tom surveys the current crop of sensors and their slow migration into the digital realm; and John starts experimenting with the Dallas Semiconductor DS80C320 microprocessor that can speed up any 8031 system by simply replacing the processor.

Ken

CIRCUIT CELLAR INK®

THECOMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Janice Marinelli

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Hatv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated. 4 Park Street. Suite 20, Vernon, CT 06066 (203) 875-2751 Second class postage paid at Vernon, CT and additional offices One-year (12 issues) subscription rate U.S.A and possessions $21.95, Canada/Mexico $31.95, all other countnes $49.95. All subscription orders payable in US funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 7694, Riverton, NJ 06077 or call (609) 786-0409 POSTMASTER, Please send address changes to The Computer Applications Journal, Circulation Dept., P O Box 7694, Riverton, NJ 06077.

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST
Debra Andersen
(617) 769-8950
Fax: (617) 769-8982

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

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

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

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

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (203) 871-1988; 2400/9600 bps Courier HST, (203) 871-0549

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

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

14 The Two-channel Printer Recorder/Replace that Expensive Strip-chart Recorder with a Dot-matrix Printer
by Brian Millier

28 Get Precise, with the **précis A/D** Converter/Collect Lots of Precise Data with this 16-bit, 100-kHz ADC
by J. Conrad Hubert

38 Calibrating Seismic Velocity Transducers with **précis**
by Chris Peoples

44 Patents and Copyrights for Protecting Software/Recent Progress in Finding the Proper Balance
by Barry Rein, Esq.

52 ❑ Firmware Furnace
Journey to the Protected Land:
Segments All the Way Down
Ed Nisley

60 ❑ From the Bench
Ta(1) king Control
Jeff Bachiochi

68 ❑ Silicon Update
In the Realm of the Sensors
Tom Can trell

74 ❑ Embedded Techniques
Speed Demon in 803 l's Clothing/Exploring the DS80C320 Processor
John Dybowski

INSIDE ISSUE 49

2 **Editor's INK**
Ken Davidson
The Ultimate Data Collecton Machine

6 **Reader's INK**
Letters to the Editor

8 New Product News
edited by Harv Weiner

ConnecTime 82
Excerpts from the Circuit Cellar BBS conducted by Ken Davidson

Steve's Own INK 96
Steve Ciarcia
Time to Move On

Advertiser's Index 81

READER'S INK

So Let's Hear it for the Environment

I have been following ***Circuit Cellar INK*** since its appearance. I have never seen articles related to the environment. I would like to know, for example, how I can use the wind and the sun to provide electricity in my home? The design of a control system to store the energy and monitor power consumption would be interesting.

Everybody can benefit from using conventional forms of energy. Some areas are particularly suitable for using the wind and/or solar energy. In the Mediterranean area, for instance, there is a huge installation of solar panels used to provide hot water. Maybe these panels could be used in an additional way. There are ways (special heat pumps) to use underground water to heat a house in the winter or air condition it in the summer.

There is air and water pollution in almost every city, and I believe it would be interesting to learn how one can measure the mass of the various gases (CO, CO^2, NO, etc.). I'd also like to be able to measure the pollution in the water resources.

Please include articles that will help me do my part in saving the environment.

Yannis Roussias
Athens, Greece

LAN/WAN Chip Set: One Chip Only Please

I have been a subscriber since issue #23. In that issue was an article discussing the ISDN BRI chip set that AT&T Microelectronics had developed. Since then, I have seen many articles on I^2C, CEBus, X-10 and now, in issue #47, BIONet. I have enjoyed reading them and like the detail you provided. I have an interest that you could satisfy with articles similar to the AT&T article.

I have developed a great deal of appreciation for the complexity of the IS0 OSI network model. ISDN, X-25, LAN, and WAN protocols are all defined as the lower part of it with the 3/4 interface of each being identical. I believe that, with single-chip implementations of ISDN's layers, ***l-3*** should now be possible or at least more complete than the 1991 AT&T chip set, which didn't have B channel multiplexing, security, accounting, or control management. I believe Rockwell, Seimens, Northern Telecom BNR, and AT&T are all working on this. Could you check with them to see if one of them would consider writing an article for your magazine on their work?

Along these lines, I also would like to see X-25 over serial/parallel cabling discussed, preferably with single-chip implementation. LAN and WAN single chips are also possible, but most that I have seen have truncated their protocol support with the Microsoft-defined NDIS drive interface, instead of the ANSI/ISO-defined interface. A single-chip implementation of LAN and WAN protocols to the ISO-defined level 3/4 interface (sans routing) would be an interesting series of articles.

I say this as the IS0 has finally finished the management functions definitions in 1992. Thus, the chips now coming out will or should have complete support for all lower-level operations.

William L. **Hartzell**
Garland, TX

New Address

The Fomebords Co. was listed in the source section at the end of the "Prototyping-Beyond Electronics and Software" article in the June 1994 issue. They have since changed their name and address. Contact them at

Superior Fomebords Corp.
1040 N. Halsted St.
Chicago, IL 60622
(312) 278-9200
(800) 362-6267

Contacting Circuit Cellar

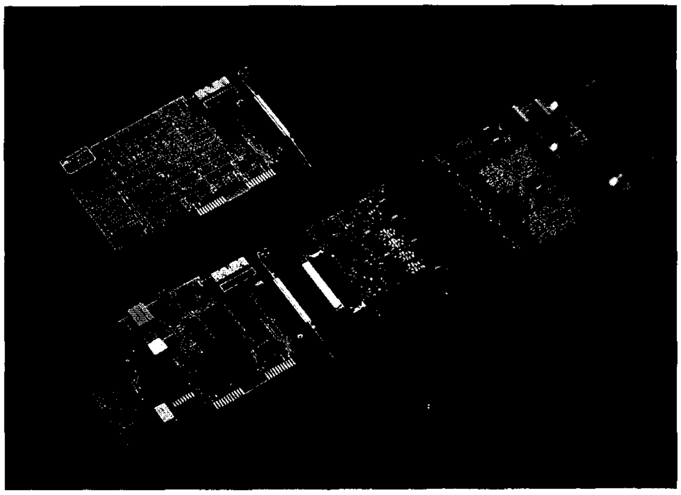
We at the *Computer Applications Journal* encourage communication between our readers and our staff, so have made every effort to make contacting us easy. We prefer electronic communications, but feel free to use any of the following:

Mail: Letters to the Editor may be sent to: Editor, The Computer Applications Journal, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (609) 786-0409. Contact our editorial offices at (203) 8752199.
Fax: All faxes may be sent to (203) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (203) 871-l 988 with your modem (300-l 4.4k bps, 8N1).
Internet: Electronic mail may also be sent to our editors and regular authors via the Internet. To determine a particular person's Internet address, use their name as it appears in the masthead or by-line, insert a period between their first and last names, and append "@circellar.com" to the end. For example, to send Internet E mail to Jeff Bachiochi, address it to jeff.bachiochi@circellar.com. For more information, send E mail to info@circellar.com.

NEW PRODUCT NEWS

Edited by Harv Weiner

ADC BOARD FOR THERMOCOUPLE DATA ACQUISITION



The **Direct Connect 5508SCi** from ADAC Corp. is the first PC plug-in board designed for industrial thermocouple data acquisition. The 5508SCi is a high-performance A/D converter board that includes a detachable screw-terminal panel which provides 1500-V isolation. The board contains integral signal conditioning to accept thermocouple inputs directly. In addition, the unit allows thermocouple type to be user selected on a channel-by-channel basis.

The 5508SCi features a unique Universal Software Interface (USI) to allow true cross-compatibility between software and hardware from different vendors. USI allows the 5508SCi to match the Control and Status Registcrs of other manufacturer's plug-in boards. USI also provides compatibility with virtually every DOS- and Windows-based data acquisition and control software package available.

The 5508SCi incorporates isolation located at the screw terminations outside the PC to prevent potentially devastating voltages from entering the host computer. For applications that have the potential for signal alias problems, the 5508SCi can be ordered with integral 8-pole antialias filtering. Available on a per-channel basis as either Bessel, Butterworth, or Elliptical filtering; the option eliminates the need for costly front-end filter systems. Prices start at $695.

ADAC Corp.
70 Tower Off ice Park
Woburn, MA 01801
(617) 935-6668
Fax: (617) 938-6553

#500

FAST CHARGE IC

The **bq2004 Fast Charge IC** from Benchmarq incorporates Peak Voltage Dctect (PVD) fast charge termination for NiMH batteries. PVD is the recommended voltage termination schemc for NiMH from some battery manufacturers. The bq2004 is ideal for in-system charging with a simple-to-use power-down mode and small 16-pin SOIC package.

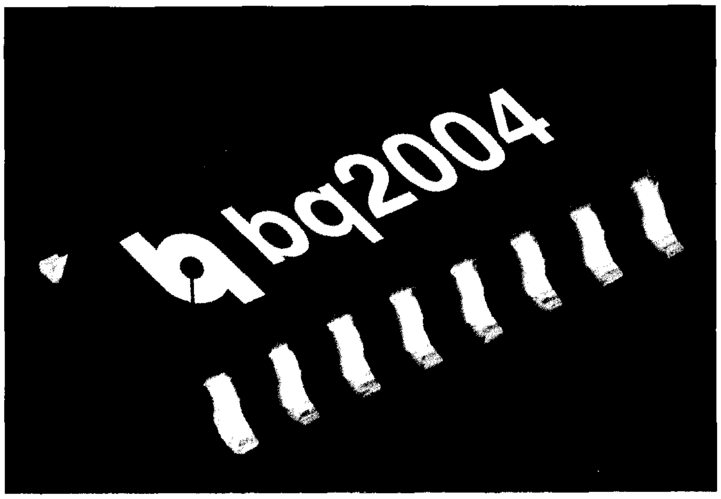
The bq2004 terminates charge based on negative delta voltage detection and on temperature-slope sensing. This involves calculating the slope of the battery temperature curve using the rapid temperature increase associated with fully charged batteries. This technique provides charge termination quickly when the rate of temperature increase is outside predetermined limits. Fail-safe terminations include maximum temperature, charge time, and battery voltage.

The bq2004 also offers modulated output to 300 kHz for switched-mode current regulation, LED outputs to display battery and charge status, low-power standby mode, battery temperature and voltage qualification before fast charge, and pulse trickle and "top-off" charge control.

The bq2004 sells for $5.44 (16-pin narrow DIP) and $5.83 (16-pin narrow SOIC) in I k quantities. Development systems are also available.

Benchmarq Microelectronics, Inc.
2611 Westgrove Dr., Ste. 109 • Carrollton, TX 75006
(214) 407-0011 • Fax: (214) 407-9845

#501



NEW PRODUCT NEWS

LOW-COST FORTH COMPUTER

The TDS9092, a CMOS single-board Forth computer/controller, is available from the Saelig Co. The 8-bit computer features the Hitachi HD63A03Y microprocessor, a 16K Forth language kernel, and built-in symbolic assembler. Generated code can be stored in either nonvolatile RAM or EPROM on-board. Typical uses for the unit include machine-tool control, instrumentation, data logging, flow control, and measurement

The TDS9092 features 35 I/O lines, two RS-232 serial ports, a watchdog timer, a driver for I^{2}C serial peripherals, and two timers. Memory included on the chip includes 16 KB of RAM for data collection applications arrays and variables, 29 KB user application space, and 256 bytes nonvolatile EEPROM. The 4" x 3" board runs from very low power (3-15 mA at 6-16 V) and features a battery-low output, touch switch turn-on, and an on-board regulator.

The interactive Forth specially written for the board gives easy access to all its features and allows software to be quickly written. The Forth includes LCD and keyboard drivers together with many other utilities. Programs are written in a high-level language and can be mixed with assembler if required.

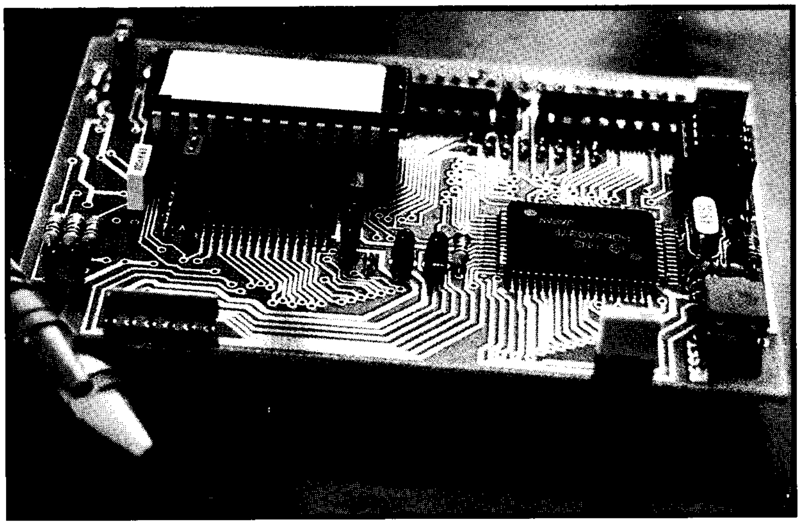
With Forth on-board, there is no need for an in-circuit emulator. The computer is programmed with an ordinary PC, building up program segments as needed. A defined assembler routine can be tested immediately without any downloading step. The interactive debugging of assembler is very powerful.

A StarterPack has everything needed for fast instrumentation control. It includes an improved, comprehensive, ready-made software library from which to mix-and-match subroutines.

The TDS9092 sells for **$105** in quantities of 25; the StarterPack sells for $225.

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

#502



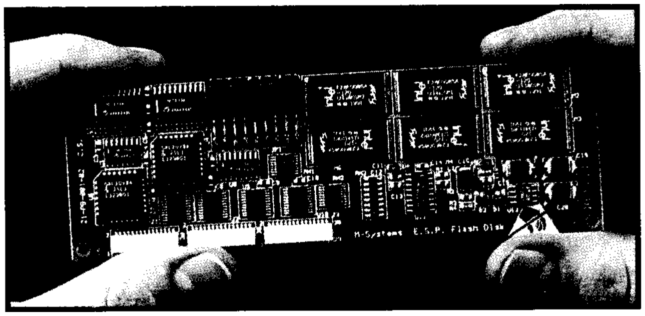
FLASH DISK

A line of Flash disk products in the E.S.P. form factor has been announced by Dovatron. The use of solid-state disk technology allows system integrators to greatly reduce power and space while eliminating the need for rotating media.

The E.S.P. Flash disk is available in 2–16-MB capacities, and up to four modules can be added to a single E.S.P. system. Flash technology allows almost instant read/write access and requires no power to retain data when idle. There is an optional boot ROM feature allowing the system to boot directly from the flash card. All E.S.P. Flash modules are shipped with M-Systems's TrueFFS Flash file utilities software.

E.S.P. (Extremely Small Package) is a 100% PC-compatible product in a small, low-power form factor. The product line offers processors ('286, '386, '486), power supplies, and a variety of I/O functions. All E.S.P. modules are 1.7" x 5.2" and conform to the ISA electrical standard.

Dovatron International
1198 Boston Ave. • Longmont, CO 80501
(303) 772-5933 • Fax: (303) **651-0916** **#503**



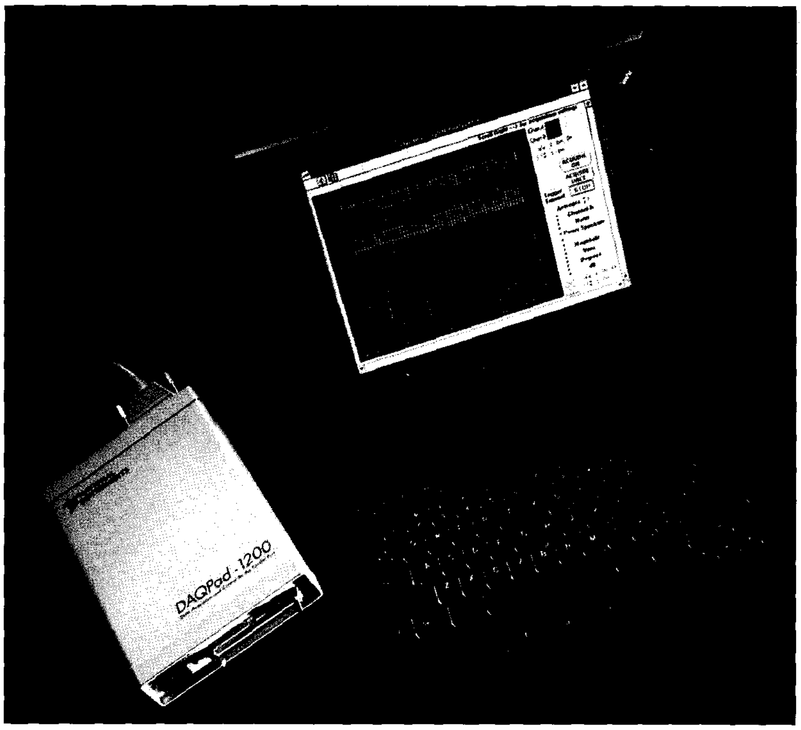
NEW PRODUCT NEWS

DATA ACQUISITION BOX

National Instruments has announced a compact, high-performance, external data acquisition box. Compatible with any PC that has a parallel printer port, the **DAQPad-1200** is ideal for PC-based data acquisition involving laptop and notebook PCs, or PCs with the available slots filled.

The DAQPad- 1200 has a 12-bit A/D converter that can digitize from eight single-ended or four differential inputs at rates up to 100 kS/s. It features programmable gains of 1, 2, 5, 10, 20, 50, or 100; a 2-kilosample first-in/first-out ADC buffer; two 12-bit D/A converters with voltage outputs; 24 lines of TTL-compatible digital I/O; and three user-available 16-bit counter/timer channels.

The DAQPad- 1200 is fully software calibrated and software configurable, with no jumpers or trimpots. It can sample in a variety of modes, including externally timed acquisition, external trigger with pre- and posttrigger mode, and interval scanning. With interval scanning, the DAQPad- 1200 scans all selected input channels at one rate, then delays a programmed interval before repeating the scan.



The DAQPad- 1200 is compatible with the IEEE 1284 Enhanced Parallel Port (EPP) standard and works with two types of ports: standard Centronics (unidirectional) and the fast EPP ports. In EPP mode, data is transferred at rates up to 100 kS/s. The DAQPad- 1200 features a second parallel port connector so users can simultaneously connect the unit to a PC and printer.

Software for the unit includes NI-DAQ, the company's library of DAQ functions for DOS, Windows, and Windows NT applications. The DAQPad- 1200 includes an AC adapter to supply power from a 120- or 240-VAC wall outlet. An optional battery pack with charger will power the unit for 9-12 hours.

The DAQPad- 1200 and AC adapter sell for $995. The battery pack sells for $295.

National Instruments
6504 Bridge Point Pkwy.
Austin, TX 78730-5039
(512) 794-0100
Fax: (512) 794-8411

#504

DEVELOPMENT TOOLS HANDBOOK

Intel Corporation has announced the availability of the Second Edition of the MW Media ***Development Tools Handbook,*** listing support solutions for the Intel MCS51, MCS96, and 80C186 architectures.

Created to assist embedded system designers, the ***Development Tools Handbook*** features design support products and services from 62 companies. It contains over 145 pages of information on microcontrollers and peripheral components, compilers and assemblers, debuggers, emulators, simulators, analyzers, programmers, design services, training, and accessories. Each page is a complete data sheet detailing each product with descriptions, features, specifications, contacts, and ordering information. This edition also contains a tutorial on "Understanding the Development Cycle" by Steven McIntyre, an Intel applications manager.

To obtain a free copy of the book, call the Intel Literature Center at (800) 548-4725.

#505

NEW PRODUCT NEWS

CRYPTOGRAPHIC INSTALLER

HPI has released new versions of **Instalit,** a cryptographic installer incorporating public and private key and RC4 encryption. Available in 14 national languages, Instalit/Crypto is designed for developers wanting to lock products on diskettes or CD-ROM.

Developers and data distributors can encrypt product or data files as they are compressed for distribution using keys of any length. Distributions can be built so that, even on identical CD-ROMs or diskettes, a unique key is needed for each installation process performed. After payment arrangements, software vendors can provide an appropriate key allowing product installation. With these techniques, customers can purchase new product increments at a later date or after trying a demo version.

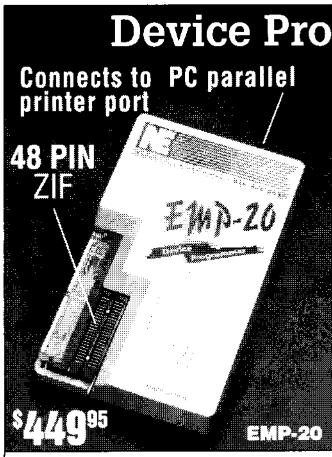
Instalit/Crypto is a complete product release system. The package incorporates data compression, patching, programmable automatic release production, scriptable compressed library builds, and diskette duplication technology. Versions are available for DOS, OS/2, Microsoft Windows, or Windows NT.

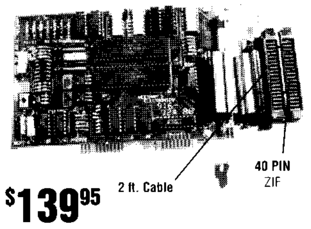
A communicating remote installer version, called **hpiOmni,** can install or intelligently update a product from a local computer to a remote computer via ZMODEM exactly as though a diskette set had been sent to the remote site. Key exchange can be handled in a completely automatic fashion. The software can operate in unattended mode and can also be used for directory synchronization, hardware and software inventory, backup, and general scripted utility functions at remote sites. Besides remote installation, the product can be used for routine customer check-in for potential update purchase with immediate direct installation.

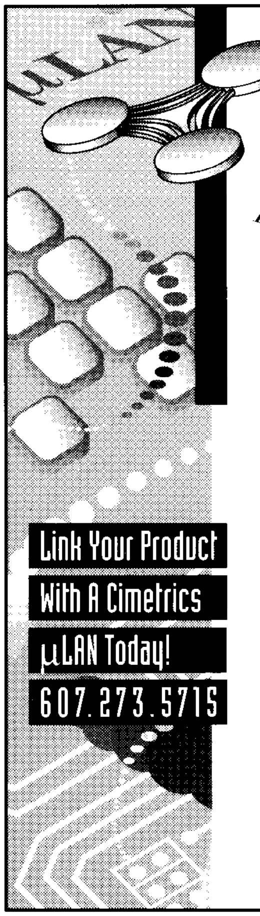
Prices start at $299 for Instalit and $349 for hpiOmni. No royalties apply. Demos are available on HPI's BBS.

HPI
917C Willowbrook Dr.
Huntsville, AL 35802
(205) 880-8782
Fax: (205) 880-8705
BBS: (205) **880-8785**

#506







NEW PRODUCT NEWS

X-10 POWER STRIP

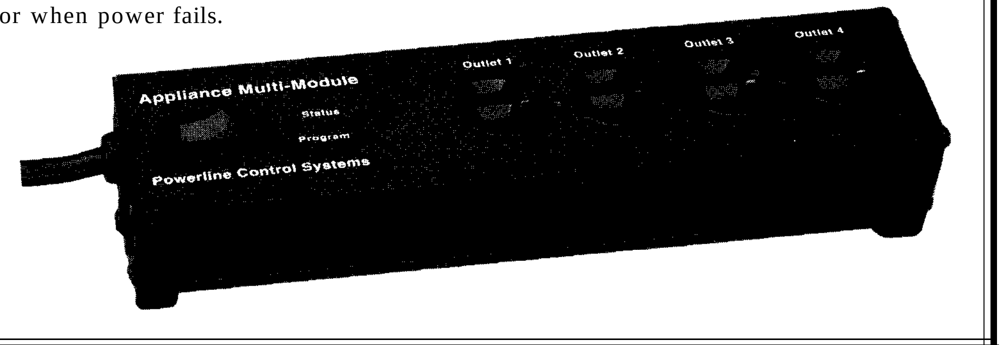
PCS has provided a solution for the need to plug more than one X-lo-type module into a standard wall outlet. The **Multimodule** is the equivalent of four X-IO-compatible modules packaged in a power-strip enclosure.

Multimodule is available in three versions: a four-outlet lamp module, a four-outlet appliance module, and a combination lamp/appliance module. The lamp version allows the user to safely control any combination of lights up to 1200 W from any combination of the four outlets. The appliance Multimodule can control any combination of appliance loads up to 15 amps. Every Multimodule is fully protected by a resettable circuit breaker. Each outlet can handle the full load, so every individual outlet is completely protected.

The Multimodule features four consecutive addresses and is compatible with all X-10 commands. The 10.5" x 1.7" x 2.6" unit includes standard 3-prong receptacles and a 6-foot power cord. Advanced features are available to allow custom user configuration of various options, such as enable/disable of dimming, remote On activation by load power switch, lamp flashing, and noncontiguous outlet addressing. The lamp module allows outputs to brighten from off without coming to full On first; preset dim and All Lights Off commands; and retention of the current dim level when outlet is turned off or when power fails.

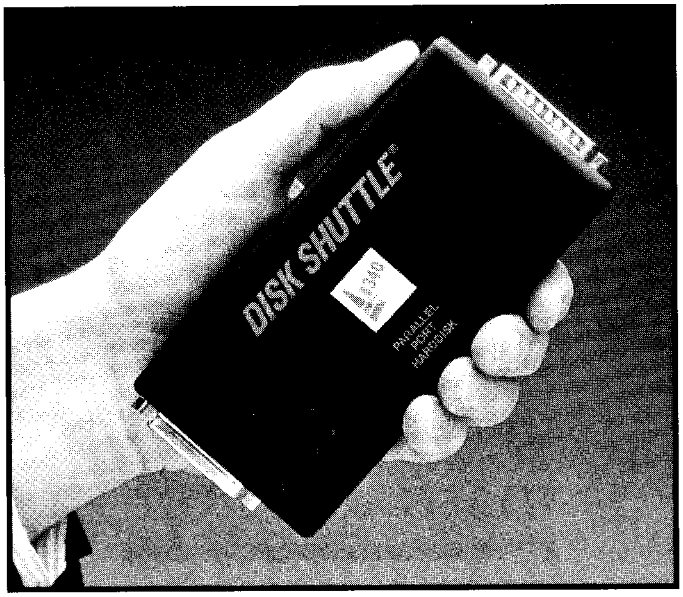
Powerline Control Systems
9031 Rathburn Ave.
Northridge, CA 91325
(818) 701-9831
Fax: (818) 701-I 506

#507



NEW PRODUCT NEWS

PARALLEL PORT HARD DISK



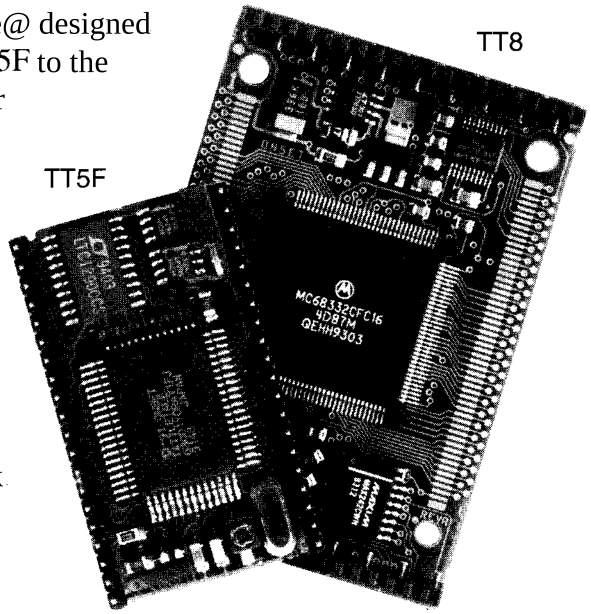
Disk Shuttle is a fast, reliable way to increase the hard disk drive storage capacity of a PC. The palm-sized unit features data storage capacities of 170 MB, 260 MB, and 344 MB, and is ideal for storage expansion, file transfer, backups, and PC installations.

Disk Shuttle features a 2.5” form factor hard disk that installs in seconds on any parallel printer port without additional hardware for DOS and OS/2 systems. The printer can be plugged into the other end of the Disk Shuttle for transparent use. No CONFIG.SYS changes or controller cards are needed and software is provided to complete the installation.

Disk Shuttle features an average access time of 8-12 ms and a data transfer rate of 6 MB/s. The unit measures 6″ x 3” x 1” and weighs 12.5 ounces. The Mean Time Between Failures (MTBF) is 300,000 hours.

Disk Shuttle is offered as a complete kit. It comes in a carrying case and includes power and data cables, driver software, manual, and 2-year warranty. Prices start at $459 for the 170-MB model.

Computer Connections America
19A Crosby Dr. **.** Bedford, MA 01730
(617) **271-0444 •** Fax: (617) 271-0873 **#508**



FEATURES

14 A Two-channel Printer Recorder

28 Get Precise, with the précis A/D Converter

38 Calibrating Seismic Velocity Transducers with précis

44 Copyrights and Patents for Protecting Software

A Two-channel Printer Recorder

FEATURE ARTICLE

Brian Millier

Replace that Expensive Strip-chart Recorder with a Dot-matrix Printer

The strip-chart recorder eliminates worries about lost memory or post-processing data to graph it, but it can be expensive. Get most of the features without the cost by using a common dot-matrix printer.

One of the byproducts of the constantly advancing technology associated with the personal computer market is the availability of many useful devices that can be obtained “dirt cheap.” This occurs when some new technology makes a particular product less attractive than it was when the manufacturer ordered all the parts needed to make thousands or more of these widgets. Working in a university chemistry department, I often take advantage of this situation to design instruments using such inexpensive components and assemblies.

While research and teaching labs have embraced computers and modern data acquisition systems with open arms, there still exists a substantial need for the trusty old strip-chart recorder. The cost of a two-channel recorder still exceeds $1000 in most cases. A dot matrix printer, on the other hand, fighting off low-cost laser and inkjet printers, can currently be purchased for well under $200. Add a microcontroller with A/D conversion, a large (and cheap) LCD display, and you have the makings of an inexpensive two-channel strip-chart recorder with some added advantages thrown

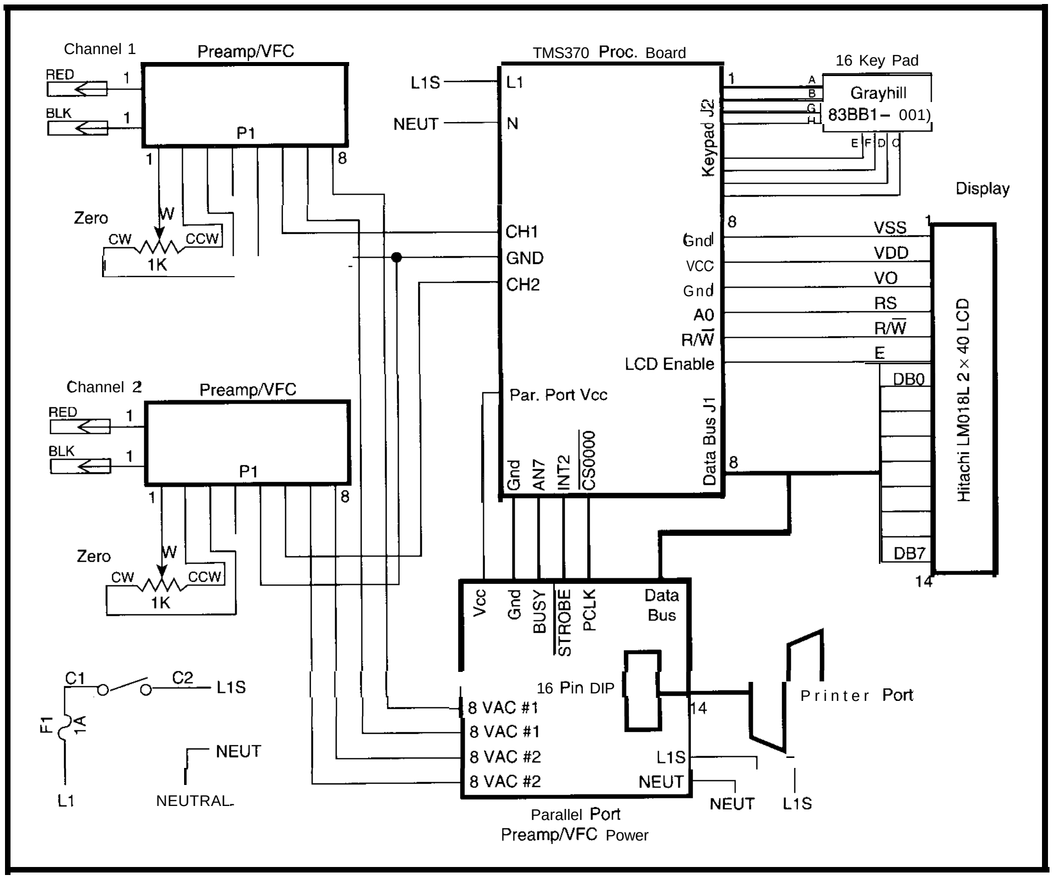


Figure I--The *printer recorder is built in modules so different front ends may be connected to the same core processor board.*

in. Printer paper is much cheaper than recorder paper, and this recorder will print out pertinent information such as the date, chart speed, and so forth at the end of the run. Due to the print speed limitations of the dot matrix printer, though, the fastest chart speed available is two inches per minute, so this project is definitely best suited for slow data acquisition.

BASIC CONCEPTS

There are several basic considerations to be addressed in designing a strip chart recorder. First and foremost, a strip chart recorder is designed so its input terminals float with respect to ground so signals that ride on some voltage may be measured. This voltage may be high, and may in fact exceed the magnitude of the signal itself by several orders of magnitude. Another reason to have the input terminals float is to minimize the noise produced by *ground loops.* Ground loops occur when both the device being measured and the measuring device are ground-referenced, but wiring limitations make the individual grounds exist at somewhat different potentials. This is particularly troublesome when measuring low-level signals. I also chose to power each channel with its own floating power supply.

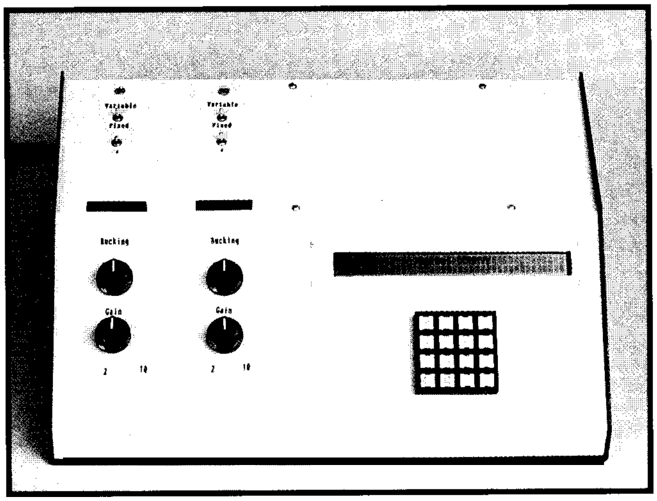


Photo I-A*ll configuration of the printer recorder is done through the front panel. In addition to displaying text prompts, the LCD display is a/so used to set the "pen" limits.*

The actual analog-to-digital conversion is performed by an Analog Devices AD654JN voltage-to-frequency converter. This device produces a square wave pulse train proportional to the voltage applied to it. An inexpensive opto-isolator passes the pulse train to the microcontroller while still maintaining the floating nature of the input circuitry.

Next, to prevent paper waste in a lab environment where measurement parameters vary greatly from experiment to experiment, it is crucial that the user be able to operate a recorder with the pen(s) moving, but with the chart paper stopped. This feature allows you to adjust the gain and zero controls to place the pen where you want it, as well as to get some idea how far the pen is going to move for different experimental conditions.

Since a dot-matrix printer doesn't have a pen to move across the paper, I use a 40-character by 2-line LCD to set the limits. During setup, the display prompts the user for various parameters. After parameter entry, but before the data collection and printing actually starts, I use the LCD as an analog needle display with a resolution of 128 positions to set the ranges. It is something like the bar display you see on some digital multimeters, but with much higher resolution.

The third design consideration involves the data printing itself. I chose to work with Epson-compatible printers which have a 960-dot mode. In a strip chart mode, this corresponds to a vertical resolution of 960 points over 8 inches or $\frac{1}{120}$″. This rivals commercial strip chart recorder resolution.

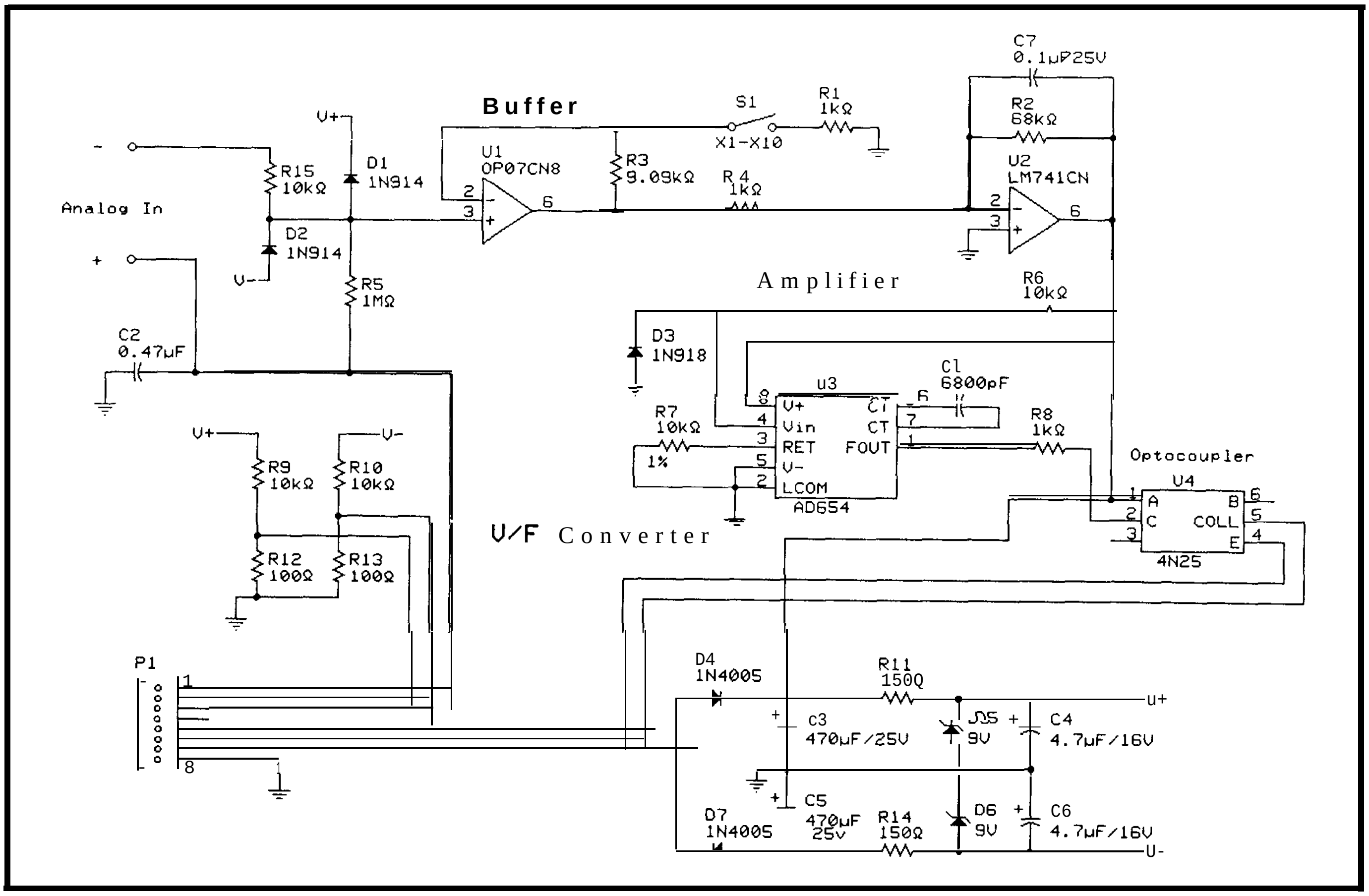


Figure 2— *The front end of the* recorder buffers *the input* signal and converts if *to* a frequency for measurement *by the processor.*

Software that provides bit-mapped graphics output to a dot matrix printer generally does so by filling a memory array with the 960 bytes of data that the printer needs to print each graphic strip. While 960 bytes of RAM is trivial in a desktop machine, RAM is valuable in a microcontroller, which typically has only 256 bytes of RAM. I wrote an algorithm which eliminates the need for this 960-byte memory buffer and allows me to get by with only the 256 bytes of RAM in the microcontroller.

The final consideration involves filtering. Most data acquisition involves some noise. It is generally best to filter this noise out, particularly before it goes to the hard copy device. I use a fixed one-pole RC filter in the preamplifier and a variable-time-constant, weighted-average digital filter in firmware.

THE NUTS AND BOLTS

While many embedded projects use the ubiquitous 803 1 or 68HC 11, I've done numerous successful projects over the years using the Motorola 68701 microcontroller (with EPROM). For this project, though, I decided to upgrade to a more powerful micro family. Texas Instruments had won me over with its DSPs, its imaginative analog and mixed signal parts, and its generous attitude toward universities, regarding technical literature and free samples. I, therefore, chose the TMS370C250 microcontroller and designed a PC board for it that would act as a platform for a number of different projects. The sidebar on page 21 contains a short summary of the features of this device.

Figure 1 contains a block diagram of the entire recorder. The preamplifier and voltage-to-frequency converters are built on two small, identical PCBs. I can, therefore, build different preamps for different applications and plug them into the common circuitry that makes up the rest of the recorder.

Referring to Figure 2, you will see a half-wave-rectified, zener-regulated power supply providing ±9 volts to run the circuitry on board. There is no transformer on this board; instead, a small transformer with a dual 8-VAC secondary mounted on another board powers each preamp individually. This allows each preamp input to be completely floating, both with respect to ground and to each other.

U1 is an OP07CN8 op-amp used as the input buffer. This high-quality device has very low input offset current and is very stable. The gain of this stage is either 1 or 10 depending upon the setting of switch S1 (changing R1 would allow for other gain ratios to be selected if desired). The input signal is referenced to a variable bucking voltage that comes from the wiper of a lo-turn zeroing pot that is located off-board. The values of R12, R13, and the pot itself could be changed to suit a particular application or eliminated completely. This particular preamp has a full-scale input of 10 mV or 100 mV, depending upon the setting of S 1.

U2 is an inverting amplifier with a gain of 68. The preamp is designed to produce 6.8 V at the output of U2 with

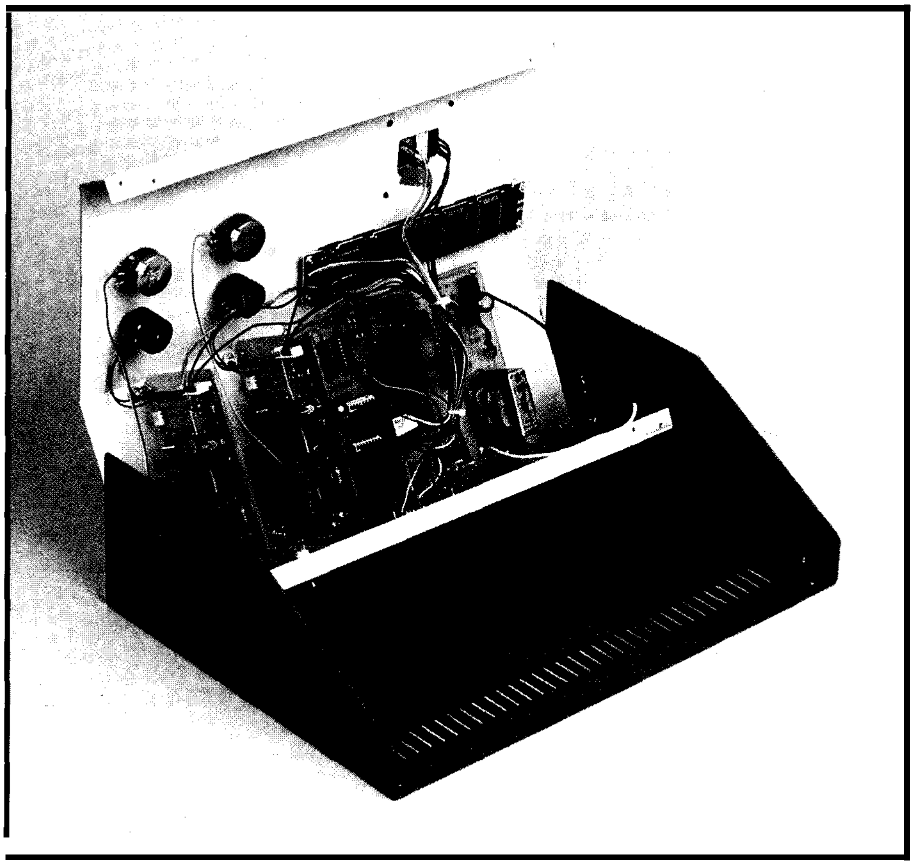


Photo 2—*Inside the printer recorder, the amplifier* modules may *be unplugged and replaced to allow the recorder to be used in many different applications.*

the rated input voltage applied. The 741 will not produce much more output voltage than this with a 9-V supply, and I make use of this clipping to ensure the AD654 voltage-to-frequency converter is never driven beyond its full scale (it acts erratically when overdriven). A single-pole RC filter made up of R2 and C7 filters out high-frequency noise.

The Analog Devices AD654JN VFC is a very simple, yet versatile, device. The value of the resistor (R7) from pin 3 to ground sets the full-scale input range (V full scale = 0.001 amp x R7). In this case, it is 10 V full scale. The combination of R6 and D3 remove any negative input signal greater than one diode drop. The full-scale frequency is set by Cl (6800 pF) to provide a nominal value of 14700 Hz. With the rated input signal applied to the preamp, input to the AD654JN is 68% of its full-scale input voltage. This leads to a full-scale pulse train frequency of 14700 x 0.68, or 9996 Hz.

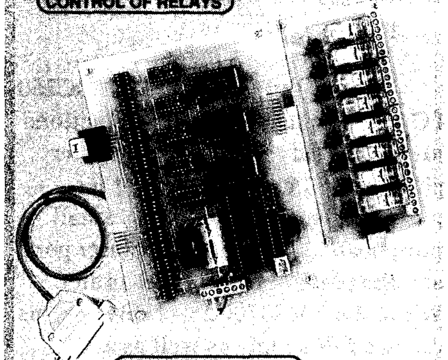
The counting time for each sample is 83.3 ms, yielding a full-scale count value of 833. The software maps these counts directly into a vertical-axis dot position. Since the printer's dot resolution is 960, there is about a 15 % overrange capability. I must note that the input signal to the AD654JN should never exceed (V+ – 4 volts). To satisfy this criterion, the AD654JN is fed from the unregulated power rail (approximately 12 VDC).

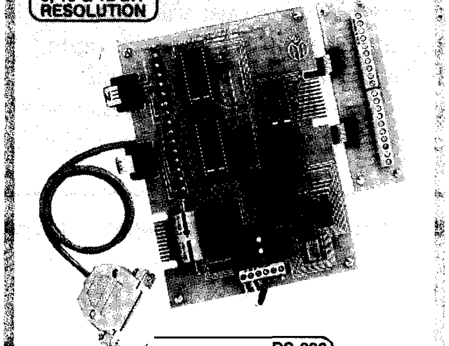
A simple 4N25N optocoupler with a transistor output is used to couple the pulse train to the microcontroller while maintaining ground isolation.

THE DIGITAL SIDE

As shown in Figure 1, the microcontroller handles five discrete functions:

1) electronic switching (4052 CMOS multiplexer) to rapidly switch between the pulse trains from the two channels,
2) pulse accumulation (part of the TMS370C250) to count the pulses,
3) reading a 16-key matrix keypad for parameter entry,





4) sending data to a 2 x 40 LCD module, and

5) sending printer data to a Centronics parallel port.

My general-purpose TMS370C250 PC board contains all of the functions just described except for the printer port (which I didn't need in other projects). Therefore, I wired a small protoboard to contain the printer port and the preamp power transformer. While the TMS370C250 PCB contains its own 5-V supply as well as an isolated supply for other circuitry, I did not have two isolated 8-V transformer secondaries to spare.

Referring to Figure 3, the optically isolated pulse trains from the two preamp/VFCs, labeled CH1 and CH2, are fed to U8, a CMOS 4052 dual 4-channel multiplexer. R23 and R25 are collector loads for the 4N25 phototransistors. I don't use the TMS370 Interrupt 3 pin as an interrupt input, but instead set it up as a general-purpose output to act as address line A of the 4052 mux to do the channel switching. The pulse train enters the microcontroller via the T1EVT pin. This pin can be used to clock either the Timer 1 module or the watchdog counter. In most cases, including on this project, I use the watchdog counter to accumulate pulses and leave the other two multifunction counter/timer modules free. Transformer T2 has dual 8-V secondaries to power up the two preamp/VFC PCBs, as I described earlier.

While it is possible to scan a small keypad directly using an 8-bit port and software, many of my projects require the microcontroller to do accurate time-related functions, but also be responsive to a keypress at any time. Since the TMS370C250 has no internal EPROM or ROM, it must supply a complete data/address bus, resulting in very few general-purpose I/O pins left over. Therefore, I used a 74C922 keypad scanner chip, which directly connects to the data bus as a peripheral and contains all the key debounce circuits and a Data Available pin.

Two 5-V power supplies are shown in Figure 4, one of which runs the entire microcontroller board. The other supply was present on the PCB for other purposes, but I use it in this case to power the Centronics parallel port, though that could also have been connected to the main 5-V supply.

Figure 3 also shows the actual microcontroller and associated support circuitry. The TMS370C250 runs at 20 MHz, which it divides internally by 4, giving a cycle time of 200 ns. Note the existence of a TL7702A power supervisor IC controlling the -RESET line. Since some of the projects using this board control critical devices such as large heaters or high-voltage power sources, I felt it critical to add a supervisor IC like this to shut down the controller if the power supply drops or to reset it if a power surge momentarily disrupts the V_{CC} supply. Also, the integrity of the on-board EEPROM contents is assured by this supervisor since the processor will not execute code wildly every time it is powered down. For this particular

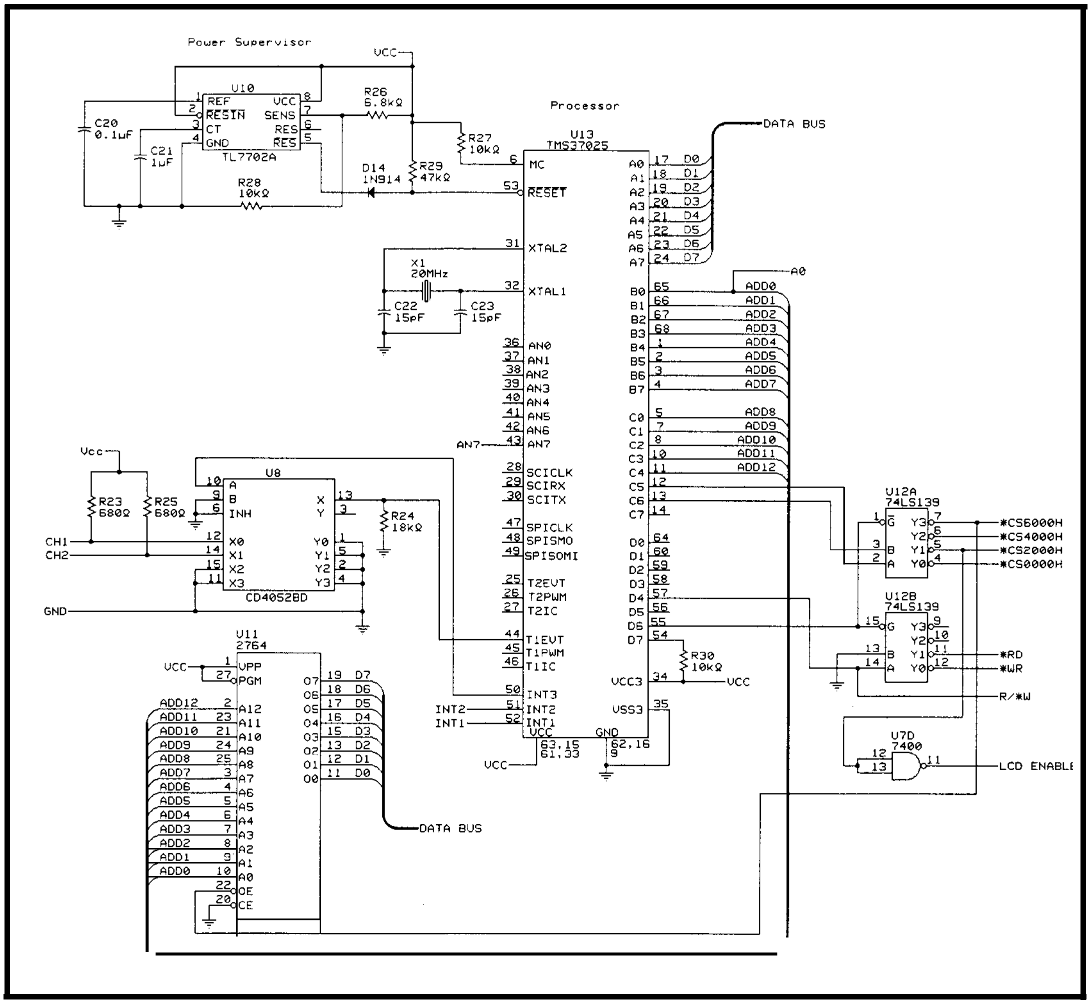


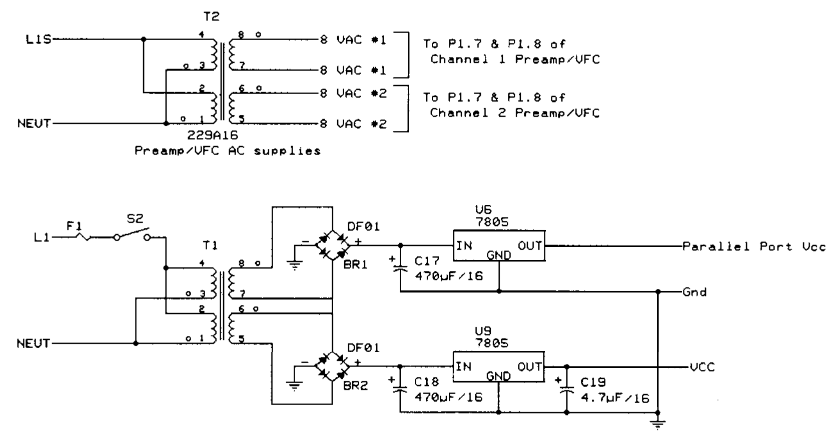
Figure 3—*At the core of the printer recorder is the TMS370C250 processor and 8 KB of program memory. A TL7702A ensures a reliable reset, even after brief power interruptions.*

application, a simple RC reset circuit would suffice.

The single 2764 (or 27C64} EPROM interfaces directly to the TMS370C250 controller. The *OE signal is derived from the 74LS139 device decoder, which also acts as the device selector for all other peripherals. Rounding out this figure is an active-low device select signal provided by the 74LS139 decoder. The signal is inverted by a single section of a 7400 NAND gate.

Figure 5 shows the printer port circuit. I use a 74LS374 octal latch as the printer data latch. The printer -Strobe signal is generated by INT2 of the TMS370C250. INT2 is another external interrupt input on the micro that I'm using instead as an output bit. The only status signal from the printer that is read is the Busy signal, which is connected to AN7—an analog input set in software to a digital input.

Figure 1 is an interconnect diagram for the whole project. The photos depict a unit designed to measure current from two photomultiplier tubes. As such, the preamps differ from the ones described. The photos also show a new PCB which replaces the general purpose TMS370 PCB and printer port/power supply board that were used when the article was submitted.



	GND	+5	+	V-
U1			7	4
U2			7	4
U3	5	8		
U4				
U5	10	20		
U6	3	2		
U7	7	14		
U8	7,8	16		
U9	3	2		
U10	4	8		
U11	14	28		
U12	8	16		
U13	9,16,62	15,33,61,63		
U14	9	18		

Figure 4—*The power supply section develops separate 5-V supplies for fhe parallel port and onboard circuitry, plus if provides unregulated AC power to the preamp boards.*

THE FIRMWARE

The firmware that operates the printer/recorder uses less than 4K of

The TMS370 Microcontroller Family

The 8-bit TMS370 family of microcontrollers from Texas Instruments is a diverse one. The TMS370Cx10 devices are low-end controllers in 28-pin DIP and PLCC packages targeted at large-volume applications using mask-programmed ROM. The ‘x30 and ‘x40 devices contain more functionality and come in larger DIP and PLCC packages. The top of the line is the ‘x50 group of controllers, which is what I have chosen to use in my designs. While there are devices in this group containing either mask ROM or OTP EPROM for program storage, I chose the inexpensive TMS370C250FNA ROMless version (64-pin PLCC). This is the most cost-effective approach since 27C64 EPROMs are very inexpensive.

When the TMS370C250FNA is configured for external program memory, it yields a microcontroller with the following features:

1) 256 bytes RAM,
2) 256 bytes EEPROM (block protectable),
3) 8 channels of fast 8-bit A/D conversion,
4) two very flexible counter/timer modules with features such as programmable prescaling, PWM function generation, and pulse accumulation,
5) watchdog timer (associated with one of the above timers, but more or less independent),
6) a full-duplex Serial Communication Interface with programmable baud rate (independent of the timers above),
7) a bidirectional three-wire Serial Peripheral Interface with programmable transfer speed,
8) three external interrupt inputs which can be either level or edge sensitive and are polarity programmable, and
9) pins associated with functions 3-8 above that are not needed for their original purposes may be reassigned as general-purpose I/O lines (some are input only).

The TMS370C250 device provides 16 memory address lines and 8 data lines, so there is no need to use a separate latch to demultiplex the address bus, simplifying the PC board design.

The maximum clock speed is 5 MHz using an external 20-MHz crystal. Most instructions execute in 6–10 cycles, with the 8-bit divide instruction taking the longest at 63 cycles.

The instruction set is much like the Intel 8031 family in that it has a rich mix of instructions, addressing modes, and bit operations, but lacks the 16-bit operations of the Motorola 68xx and 68HC11 devices. The only pseudo-16-bit operations supported are 16-bit MOVW instructions and the INCX instruction, which can add a signed 8-bit constant to a 16-bit register.

The assembly language conventions are a bit hard to get used to for anyone who has used both Intel and Motorola parts. While the MOV opcodes use Intel convention, the source-destination ordering of the operands is Motorola convention.

Texas Instruments sells a TMS370 application board that works in conjunction with a host PC computer via an RS-232 link. The board itself contains two ‘x50 devices: a master unit, which communicates with the host PC, and a slave unit.

The slave shares memory space with the master (which loads it) and runs the user’s application in real time, with access to the peripheral ports and other I/O devices. The ’x10 group of devices is also supported on this board. The host PC runs a windowed monitor/debugger/tracer program which is supplied with the application board. Cross-assembler software to run on the host PC is also provided, and a C cross-compiler and linker are available separately.

Texas Instruments runs a BBS devoted to this product family and it contains free software and information. Of particular use to new users is the M5 monitor program which fits in a 27C64 EPROM. Adding a TMS370C250 and a MAX232 produces a fully functional stand-alone evaluator. Floating point and other useful core routines are also available.

memory, residing in the upper half of a 27C64 EPROM. I will first briefly describe the overall program before going into detail about the PRINT_STRIP routine which is the core.

Upon reset, the program prompts for the current date and then goes into the parameter mode loop which allows the operator to select a chart speed and a filter time constant. Once those are entered, the program goes into the first phase of the acquisition mode loop.

All data acquisition is performed by an interrupt service routine (ISR) invoked by the Timer 1 interrupt (10 Hz). Timer 1 not only interrupts at a 10-Hz rate, but also generates a PWM gating signal for the V/F pulse accumulation. The V/F pulses are counted for 83.3 ms out of the 100-ms interrupt period. I chose 83.3 ms because it equals five line cycles, thereby giving some line-noise reduction due to integration.

The Timer 1 ISR performs the following functions in sequence:

1) reads the watchdog counter, which has been accumulating the V/F counts, then resets this counter.

2) converts this count to a floating-point number.

3) toggles a variable and the INT3 pin, which drives an address line of the 4052 analog multiplexer IC, thereby alternately reading both of the preamp input channels.

4) adds the current count (in floating point) to an accumulator and multiplies the result by a constant selected by the user’s choice of filter time constant. This result is then

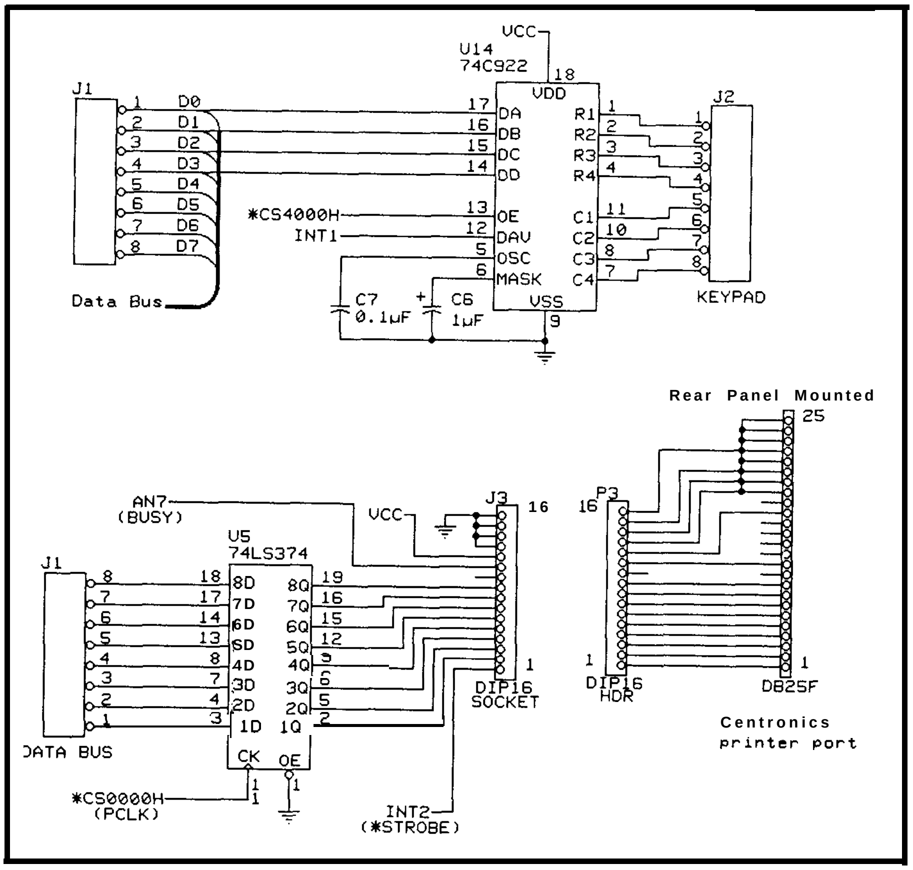


Figure 5—*A 74C922 encoder keeps the keypad interface ample, while a buffer/latch is used to interface to the printer.*

restored to the accumulator (this is a moving weighted-average algorithm that acts like a one-pole RC filter).

5) decrements the counter variable SAMCT and when it equals zero, the floating-point accumulator value is scaled and converted back into an integer in the range of O-959 (the number of dots of printer resolution across the page). This value is saved as one of eight points which will later make up a strip to be printed using the PRINT_STRIP routine. The value of SAMCT is determined by the user-selected chart speed.

6) calls the routine NEEDLE for each printer data point stored above to place a thin line on the LCD. The line acts as a pseudoanalog meter for the user's convenience in setting up the gain and zero of the recorder. The 2-line LCD allows one line per channel. I use the first 32 characters of each line for the analog "needle" and the last eight for display of the elapsed run time. The 32 characters allocated for the "needle" are further broken down into four positions per character, giving a resolution of 128, which is quite adequate for the purpose.

7) updates the Elapsed Time Clock.

All of these functions take place in an interrupt-driven fashion while the user "fusses" with the gain, zero, and the device being measured. When satisfied that all is well, the operator hits any key on the keypad and the real fun begins. In addition to the Timer 1 ISR functions listed above, after eight data points [per channel) are collected into an array, the PRINT_STRIP routine is called. This routine prints out the data to the Epson-compatible printer through the Centronics parallel port on the microcontroller board.

After printing these eight points, the pointers to the two arrays are set back to the beginning and the process repeats until the user hits a key on the keypad to end the run. At that time, the microcontroller sends to the printer ASCII strings representing the date, elapsed time, chart speed, a device name, and a sequential run serial number. The little report at the end of each run is often very useful in keeping data straight for those who do not keep good lab notes of their own!

THE PRINT-STRIP ROUTINE

This routine takes the eight data points for each channel and prints the equivalent of the two pen traces on the paper. This requires a transform of the eight amplitudes of two channels into the printer's 960-byte graphic bitmap array. That is to say, we must supply the printer with 960 bytes of graphics data per strip of data printed. Since the TMS370C250 contains only 256 bytes of RAM for all program variables, the program must do this data transform on the fly. Of course, the micro is also collecting and filtering data and updating a real-time clock at the same time, so the TMS3 70 is kept very busy indeed.

Listing 1 contains the assembly code for this subroutine. Prior to calling this routine, nine data points

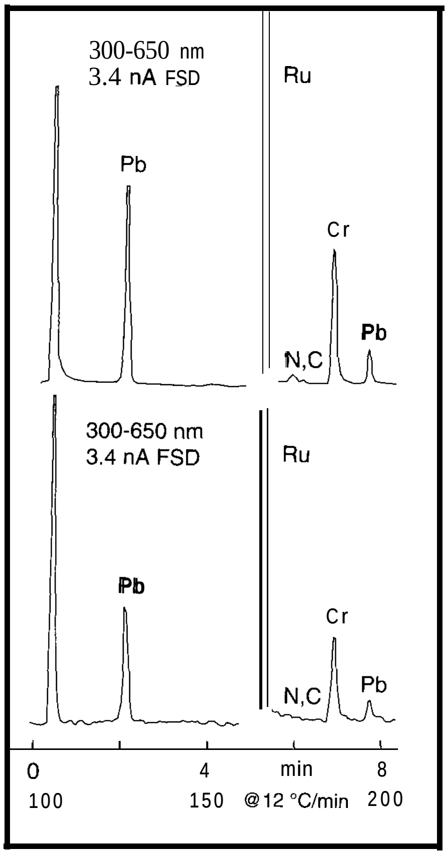


Figure **6**—*The actual output of the printer recorder is every bit as useful as that from a real strip recorder. Note that the labels were added after the printout was complete and are not automatically printed.*

Listing 1—The PRINT_STRIP routine *must convert the linear data into the dot array required by the* printer.

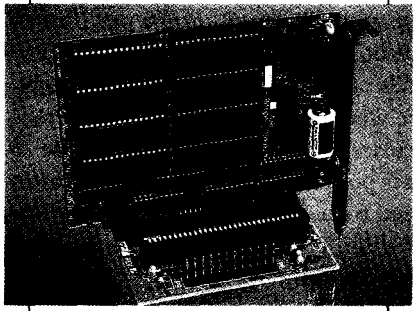
```
$PRINT_STRIP
        movw   #grafmess1,strptr   ; pnt to Epson graphics cmd str
        call   $prints_cent
        clr    pbyte

; CHANNEL #1
; make an 8-word array of dot on/dot off values: 1 for each pixel
; point to start of three arrays: data1, dot1on,dot1off

        mov    #8,loopct
        movw   #sy1,pptr
        movw   #Y1on,aptr
        movw   #Y1off,bptr
ldloop1:
        call   ldDotOnOff
        incw   #2,pptr
        incw   #2,aptr
        incw   #2,bptr
        djnz   loopct, ldloop1

; CHANNEL #2

        mov    #8,loopct
        movw   #sy2,pptr
        movw   #Y2on,aptr
        movw   #Y2off,bptr
ldloop2
        call   ldDotOnOff
```

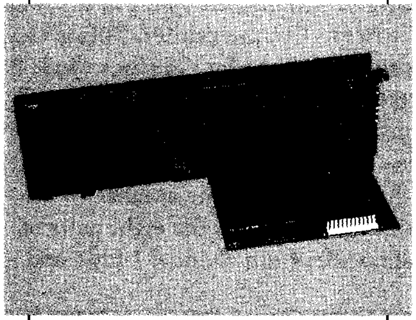
(continued)

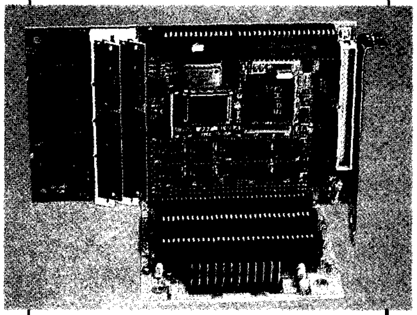
from each of the two channels have been stored in arrays SY1 and SY2. These are unsigned 16-bit integer values, prescaled to the range O-959. Nine values are needed since we must know the "position of the pen," so to speak, from the last strip printed. This adds one extra data point to the eight we wish to print. Although the 9-pin printer has the ability to print nine dots per pass, only eight of them are used to keep all operations in byte-size chunks. The main program has already sent the printer a command sequence to set its line spacing to 8/72" (each dot is 1/72" and eight dots are printed per pass]. This is done only once at the start of a run.

The actual PRINT_STRIP routine starts by sending the string at the label grafmess1 to the printer. This is the Epson command for Graphics Mode 2 (960 dots resolution). This command *must* be repeated for each printing pass.

The code starting at label ldLoop1 examines the eight pairs of data points from channel 1 (dp1 and







Listing I-continued

```
        incw   #2,pptr
        incw   #2,aptr
        incw   #2,bptr
        djnz    loopct, ldloop2

        mov    #255,a
        call   $print_cent        ; print axis

        clr    markmask
        djnz   marker,$9
        mov    #10,marker         ;9 passes of 8/72 inch = 1 inch
        mov    #16,markmask       ; axis tick at center of strip
$9      movw   #1,j               ; 1-959 (pixels across page)

pbyteLoop:                        ;check 8-bit locations for dot
        clr    b
        mov    #8,loopct
        mov    #128,mask
bitlp:
        call   RangeCompare
        inc    b                  ; advance ptr to next data point
        clrc
        rrc    mask
        djnz   loopct,bitlp
        mov    pbyte,a
        cmp    #0,j-1
        jne    $8
        cmp    #7,j
        jhs    $8
        or     markmask,a          ; add in time blip
$8      call   Sprint-cent        ; output the byte to printer
        incw   #1,j               ; advance to next Y pixel
        cmp    #3,j-1
        jne    pbyteLoop
        cmp    #192,j
        jne    pbyteLoop
                                   ; all 960 strips sent to printer
        movw   #grafmess2,strptr
        call   $prints_cent       ; send CR-LF to printer
        rts

RangeCompare:                     ; RANGE COMPARE CHANNEL #1
        clr    flg
        mov    Y1off(b),a
        mov    a,temp1-1
        mov    Y1on(b),a          ; load MSBs of Y1(b/2)on,off
        mov    a,temp2-1
        inc    b
        mov    Y1off(b),a
        mov    a,templ
        mov    Y1on(b),a          ; load LSBs of Y1(b/2)on,off
        mov    a,temp2
        movw   j,temp3
        sub    temp3,templ        ; form Y1off(b/2)-j
        sbb    temp3-1,templ-1
        jn     rcl

        movw   j,temp3
        sub    temp2,temp3
        sbb    temp2-1,temp3      ; form j- Y1on(b/2)
        jn     rcl
        mov    #1,flg             ; set the flg for a dot on event

rcl:                              ; RANGE COMPARE CHANNEL 2
        dec    b
        mov    Y2off(b),a
        mov    a,templ-1
```

(continued)

dp2, dp2 and dp3, etc.) and determines where the dot for that data point should start printing and where it should stop printing. It produces an array of eight "dot on" words and eight "dot off" words, designated Y1ON and Y1OFF, respectively. The code starting at 1dLoop2 performs the same for channel 2.

Once I know the range in which a dot must be printed for each of the eight dot positions per pass, I then set up a loop to send out 960 bytes to the printer. That loop starts at label pbyteLoop in the listing.

The print head in the Epson printers is set up so its upper pin is designated by the most-significant bit of the graphics data byte sent to it (i.e., 128 decimal). The way the printer is used for this application, this uppermost pin corresponds to the first data point taken in the group of eight that are to be printed per strip. The pbyte-Loop initially sets up the variable MASK equal to 128 and calls subroutine RangeCompare to see if either channel needs that dot turned on or off. Depending on the outcome of that check, it either sets or clears the bit in the position specified by the mask. This is repeated eight times: each time the mask value is changed to correspond to the next dot position (by a "rotate right" of MASK).

After these eight iterations of the loop, the variable pbyte is almost ready to be sent to the printer. Before sending it to the printer, however, some checks are made to see if anything must be added to the data. As a convenience to the user, a baseline axis is printed and a small tick is added to this baseline for every inch of paper travel. Examine the code following the label bit1p for details of this operation.

After 960 iterations of the pbyteLoop routine, a carriage return and a linefeed are'sent to the printer. At this point, the PRINT_STRIP routine returns to the main program.

"OUT OF PAPER"

That's the overall description of the printer recorder. We are using a number of these instruments successfully in our research labs. By placing

Listing 1-continued

```
        mov    Y2on(b),a             load MSBs of Y2(b/2)on,off
        mov    a,temp2-1
        inc
        mov    Y2off(b),a
        mov    a,temp1
        mov    Y2on(b),a           ; load LSBs of Y2(b/2)on,off
        mov    a,temp2
        movw   j,temp3
        sub    temp3,temp1         ; form Y2off(b/2)-j
        sbb    temp3-1,temp1-1
        jn     rc2
        movw   j,temp3
        sub    temp2,temp3
        sbb    temp2-1,temp3-1     ; form j- Y2on(b/2)
        jn     rc2
        mov    #1,flg              ; set the flg for a dot on event
rc2:
        cmp    #1,flg
        jne    clearbit

setbit:
        or     mask,pbyte
        rts
clearbit:
        push   mask
        xor    #255,mask           ; invert mask
        and    mask,pbyte
        pop    mask
        rts

ldDotOnOff:
        mov    0(pptr),a
        mov    a,temp1-1           ; get first of two dpoints
        mov    1(pptr),a
        mov    a,temp1
        mov    2(pptr),a
        mov    a,temp2-1           ; and second
        mov    3(pptr),a
        mov    a,temp2
        push   temp2
        push   temp2-1
        sub    temp1,temp2         ; form Y(i+1)-Y(i)
        sbb    temp1-1,temp2-1
        jn     $1
        mov    temp1-1,a
        mov    a,0(aptr)
        mov    temp1,a
        mov    a,1(aptr)           ;Yon1=Y(i)
        pop    temp2-1
        pop    temp2
        mov    temp2-1,a
        mov    a,0(bptr)
        mov    temp2,a
        mov    a,1(bptr)           ;Yoff1=Y(i+1)
        rts

$1      mov    temp1-1,a
        mov    a, 0(bptr)
        mov    temp1,a
        mov    a,1(bptr)             Yoff1=Y(i)
        pop    temp2-1
        pop    temp2
        mov    temp2-1,a
        mov    a,0(aptr)
        mov    temp2,a
        mov    a,1(aptr)             Yon1=Y(i+1)
        rts
```

the preamp/VFC on a separate PCB, I've been able to design several different preamp modules for different applications. Figure 6 shows an actual printout from the device connected to a chromatography apparatus. Special thanks is extended to Dr. Walter Aue, whose large research group never seems to have enough instrumentation. It was this need which spawned the idea in the first place. Possibly some of the concepts outlined here could also find some use in low-cost hard-copy data logging in industrial process control. ❑

Brian Millier has worked as an instrumentation engineer at Dalhousie University, Halifax, NS, Canada in the Chemistry Department for the past 12 years. In his leisure time, he operates Computer Interface Consultants and has a full electronic studio in his basement. He may be reached at bmil@chem1.chem.dal.ca.

CONTACT

Texas Instruments, Inc.
9301 Southwest Fwy.
Commerce Park, Ste. 360
Houston, TX 77074
(713) 7786592

TI Microcontroller Technical
Hotline: (713) 274-2370
BBS mentioned in the TMS370
sidebar: (713) 274-3700

Analog Devices
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
(617) 329-4700
Fax: (617) 326-8703

LCD Display
Timeline, Inc.
23605 Telo Ave.
Torrance, CA 90505
(310) 784-5488
Fax: (3 10) 784-7590

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

J. Conrad Hubert

Get Precise, with the précis A/D Converter

Collect Lots of Precise Data with this 16-bit, 100-kHz ADC

“High speed” and “high resolution” are usually mutually exclusive, but not with précis. This ISA-bus board works with any PC and has the capability to overrun even the fastest PC-compatible system.

Oraditionally, high-resolution ADCs have relied on the techniques of successive approximation and dual-slope integration to achieve accuracy greater than 15 bits. Although these techniques have served well in the past, they are not without drawbacks. Precision successive-approximation converters require complicated trimming and/or calibration schemes and are expensive. Similarly, dual-slope converters require accurate comparators and expensive sample-and-hold circuits. They are also extremely slow.

During the last five years, ADCs based on what is called ***sigma-delta modulation*** have become commercially available. Although sigma-delta modulation techniques have been around since the early 1960s, they were not often implemented because they impose a substantial digital signal processing burden.

Converters based on the sigma-delta architecture do not require precisely matched components. Instead, they use a 1 -bit quantizer (comparator) in a feedback loop. High resolution is achieved by oversampling (which shifts noise to higher, out-of-band frequencies) and on-chip digital filtering.

Unlike conventional ADCs, sigma-delta converters don’t require sophisticated antialiasing filters or sample-and-hold amplifiers. This is because their input sample rate is higher than the rate for other techniques which provide the same bandwidth.

What’s more, the sigma-delta technique lends itself to implementation in a digital CMOS process, Usually, a silicon process is optimized for either analog or digital circuitry. It is difficult to add high-performance analog circuitry to primarily digital real estate. DSP chip makers are excited about sigma-delta because it will allow them to embed the ADC right on the DSP chip itself. In fact, because sigma-delta ADCs are reasonably tolerant of switching noise, the architecture is uniquely suited to life inside digital computers.

Now that I’ve extolled the virtues of the sigma-delta architecture, I will remind you that there is, of course, no free lunch. Remember the “l-bit quantizer in a feedback loop”? Well, it points to perhaps the only disadvantage of the sigma-delta architecture: poor DC stability. (Other more complicated implementations of the sigma-delta architecture can provide excellent DC stability at the expense of lower conversion rates. Crystal Semiconductor makes such parts, but refers to them as delta-sigma A/D converters.) One other potential disadvantage is that multichannel systems usually require a separate sigma-delta converter for each channel. Multiplexing is possible, however, provided sufficient time is allowed for the digital filter to “settle” prior to accessing data from the next channel.

The bottom line is that sigma-delta converters are best suited to applications which require high sampling rates along with an extremely good signal-to-noise ratio and excellent differential linearity. Typical applications include signal processing, digital audio, communications, and ISDN (Integrated Services Digital Network).

One example of a sigma-delta converter implemented in silicon is the Motorola DSP56ADC16S. This low-cost ($24 in 100s) ADC provides 16-bit resolution at up to 100,000 samples per second (Sps) while consuming less than 0.5 W from a single 5-V supply. Recently, Analog Devices began second-sourcing this part as the AD776.

In the remainder of this article, I'll describe how to build a simple PC-based data acquisition board around the DSP56ADC16S. The schematic for the design appears in Figure 1. The board was christened *précis* after the French word for "exact". I'll describe the precis hardware in six sections, starting with the ADC and working my way to the bus interface. The printed circuit board and software will be covered last.

ADC

Because the DSP56ADC16S (U3) samples its analog input 64 times more often than it produces a digital output, a high-order antialiasing filter is not required. Motorola does, however, recommend installing a simple single-pole filter prior to the ADC. This filter should be made from a high-quality polystyrene capacitor (C5) and two metal film resistors (R2 and R3).

An input range of 4 V peak-to-peak is realized by using U3's on-chip voltage reference (nominally 2 V). Thus, an input of +2 V results in a 16-bit two's complement output word equal to 32,767. A -2-V input produces an output code of -32,768. Shorting the input connector yields an output code of 0. From these values, one can calculate a sensitivity of 61 µV per LSB.

Bipolar overvoltage protection for the ADC is accomplished via back-to-back zener diodes Dl and D2. If the absolute value of the voltage into the BNC is greater than the zener voltage plus a forward diode drop (3.8 V), these diodes short-circuit the source driving precis and clip the input signal.

One measure of the "goodness" of an ADC is its ENOB, or *effective number of bits.* ENOB is to an ADC's accuracy what word length is to an ADC's precision. The ENOB for any analog-to-digital converter is related to its SNR specification by the equation

$$\mathrm{ENOB} = \frac{\mathrm{SNR-}\ 1.76\,\mathrm{dB}}{6.02}$$

Motorola quotes the DSP56ADC16S's SNR at 90 dB; however, that figure represents best case conditions. A more realistic figure appears in the Analog Devices data sheet, which states that a 90-dB SNR is achieved with a 48-kHz sampling rate, and at 100 kHz the SNR drops to 86 dB. This yields an ENOB of 14.66 at 48 kHz and just under 14 at 100 kHz.

The input impedance of precis is nominally 2 kΩ. However, the actual input impedance is 2 kΩ in parallel with the dynamic input impedance of the DSP56ADC16S. The converter's impedance is a function of its clock frequency, and is related by the equation

$$Z_{\mathrm{in}} = \frac{10^{12}}{3F_{\mathrm{clock}}}$$

The input impedance ranges from 1.98k at 12.5 kHz to 1.85k at 100 kHz.

Because the ADC's input impedance is a function of clock frequency, the loading on the input bias circuit changes somewhat with clock frequency. Therefore, potentiometer R6 is provided to null the input offset. With the BNC shorted, R6 should be adjusted in real time to yield an offset of zero for a given acquisition rate. (The actual offset may fluctuate slightly.)

You may infer (correctly) from the preceding paragraph that the input

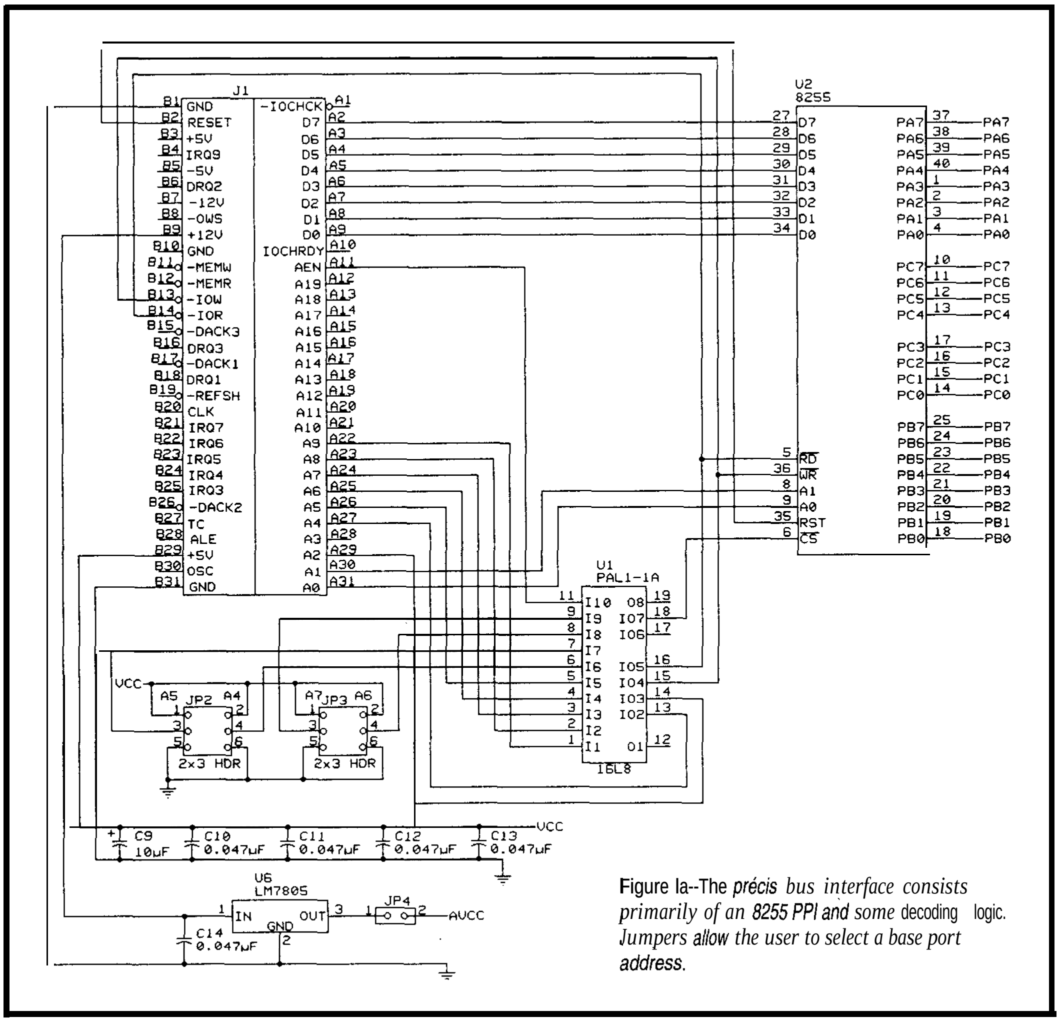


Figure la--The *précis bus interface consists primarily of an 8255 PPI and some decoding logic. Jumpers allow the user to select a base port address.*

offset must be nulled separately for every acquisition rate. If this is inconvenient, remember that it's possible to cancel the input offset by adding a constant to the data once it has been stored in main memory. The constant is determined [for a given acquisition rate) by shorting the analog input terminals and observing the resulting output code. The difference between the observed output code and zero must be added to each data point to cancel the input offset. A typical value might be between 10 and 30, and may well be negative.

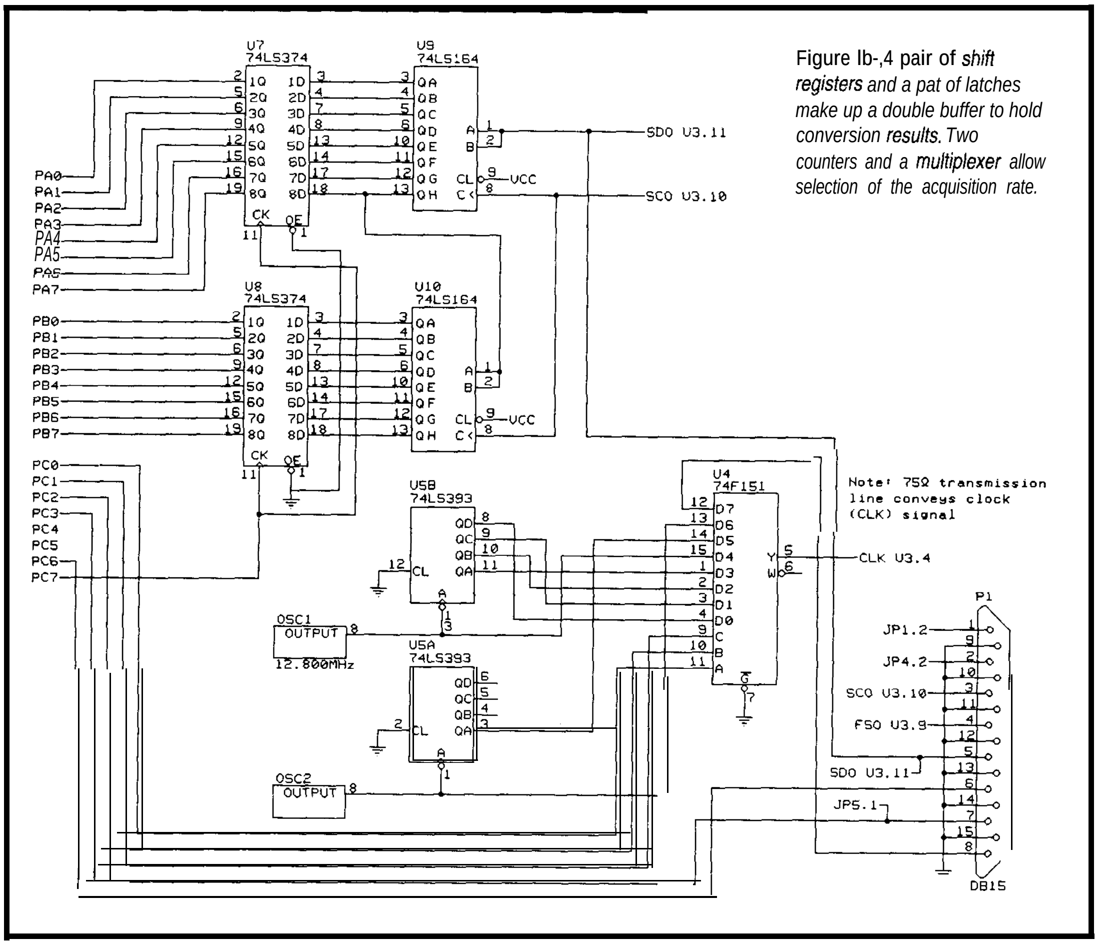
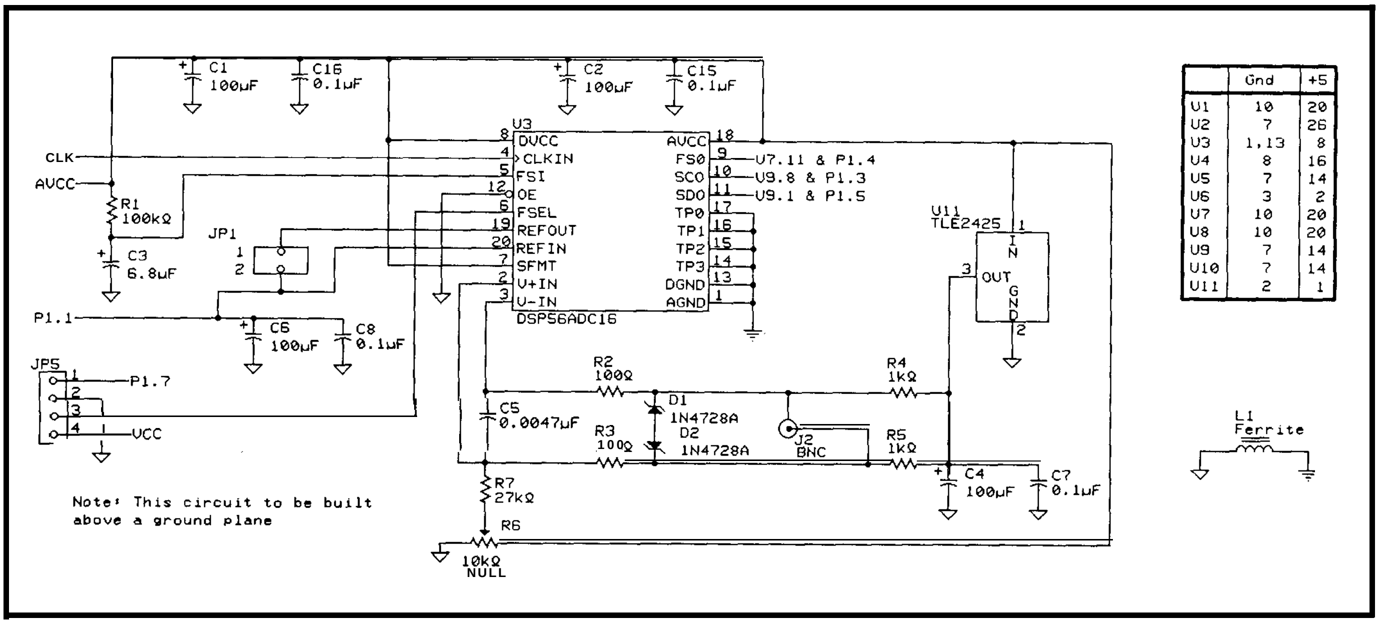


Figure lb-,4 pair of *shift registers and a pat of latches make up a double buffer to hold conversion results. Two counters and a multiplexer allow selection of the acquisition rate.*

Another interesting feature of the DSP56ADC16S is that it is really two ADCs in one. At the flip of a switch, it can become a 12-bit (resolution), 400-kHz ADC. This is possible because the part uses two internal digital filters (a fast, low-resolution comb filter and a FIR filter with slower, higher-resolution). By holding pin 6 (marked FSEL) high, the digital output is taken prior to the FIR filter. The intent of JP5 was to provide access to this 400-kHz data rate. Unfortunately, even fast computers (33-MHz '486) cannot keep up with that data rate. In practice, only the 100-kHz data rate is usable. Therefore,



	Gnd	+5
U1	10	20
U2	7	26
U3	1,13	8
U4	8	16
U5	7	14
U6	3	2
U7	10	20
U8	10	20
U9	7	14
U10	7	14
U11	2	1

Figure 1 c—*At the core of précis is the DSP56ADC16 A/D converter. The TLE2425 virtual ground generator produces a voltage precise/y halfway between 5 V and ground.* Note *that C7,C8,C15, and Cl6 are 1206-series SMT capacitors mounted on the solder side of the PC board. They may or* may not *be* required depending on *the ESR of the electrolytic capacitors used.*

JP5 should be configured to tie pin 6 of u3 low.

That about covers the DSP56ADC16S's analog side as it applies to précis. However, let's cover two of the part's idiosyncrasies. Pins **14-17** of the ADC are manufacturing test points and must be held low during normal operation. Also, please observe that the converter is always running, and that there is no "start" command as is typical with other types of ADCs. For more information about the ADC itself, order document DSP56ADC16/D directly from Motorola.

Finally, the other IC in précis's front end is due a few words of explanation. Called a virtual ground *generator* by Texas Instruments, the TLE2425 (U11) produces a voltage precisely halfway between 5 V and ground. It is used to bias the ADC's differential input amplifier at one-half V,,. [A similar part, the TLE2426 is called a rail *splitter.* This part is slightly more general purpose, having an input range of 4-40 V.)

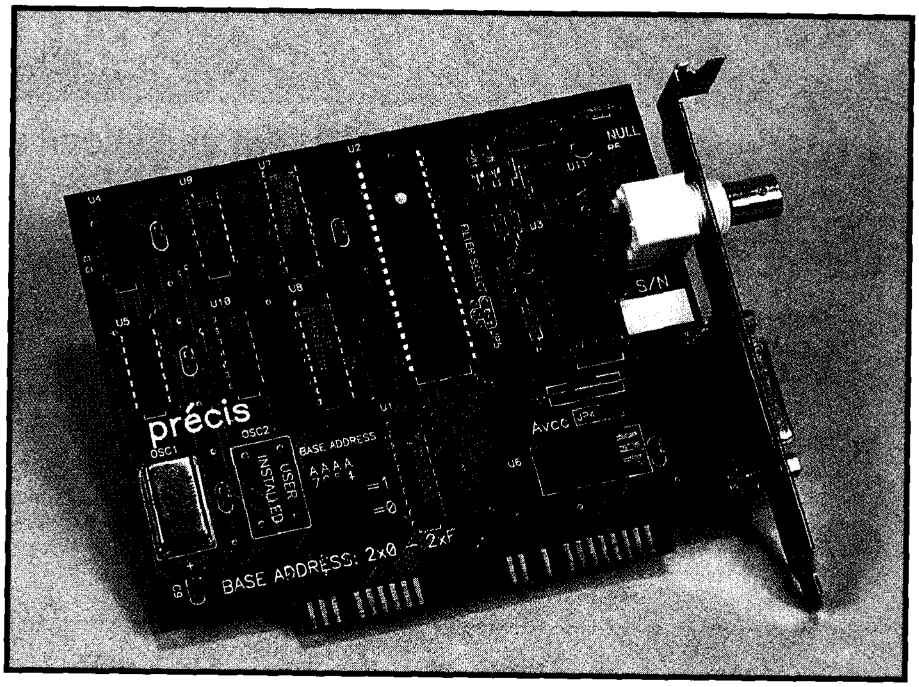


Photo *l-The précis high-speed, high-resolution A/D converter* works in any ISA-bus computer.

SERIAL-TO-PARALLEL

The output of the DSP56ADC16S is serial rather than parallel because it is intended to mate with the high-speed serial port on Motorola's 56001 DSP engine. Unfortunately, the serial data rate is far too high for a PC to accept directly. At the 100-kHz sample rate, the ADC emanates a new 16-bit word every 10 µs. However, valid data is available only during the second half of that period. This means the time available to read the output word is only 5 us. A 16-bit word read in 5 us implies a system-bus bandwidth of 400 kbps. However, by resorting to a double buffering scheme, the full 10 µs is available to read the word and the data transfer rate is decreased to 200k bytes per second. Although double-buffering eases the bus bandwidth requirements enough for précis to run on an AT-class machine, the 100-kHz sample rate is well above the capabilities of an 8086-class machine-even using an assembly language data transfer routine.

Conversion Rate	Clock Frequency	From	Mux channel
100 kHz	12.8 MHz	OSC1	D4
50 kHz	6.4 MHz	U5B-QA	D3
25 kHz	3.2 MHz	U5B-QB	D2
12.5 kHz	1.6 MHz	U5B-QC	D1
6.25 kHz	0.8 MHz	U5B-QD	D0
User rate	128 • rate	osc2	D6
User rate/2	64 * rate	USA-QA	D5
External	28 * rate	External	D7

Table *l-The précis can handle conversion rafes from* 6.25 *kHz up to 100 kHz. The user may* also *supply* an externally generated clock.

Before detailing the hardware implementation of the double buffer, I'll first describe the digital side of the DSP56ADC16S. It has three outputs: serial clock out (SCO), serial data out (SDO), and frame sync out (FSO). The format of FSO is determined by the serial format (SFMT) input. If SFMT is pulled low, FSO is compatible with DSP56001- and TMS32020-family processors. Conversely, if SFMT is high, FSO is compatible with the NEC7720. I chose the NEC format because it was easier to interface.

The double buffer itself comprises a pair of 74LS164 8-bit shift registers (U9 and U10) and a pair of 74LS374 octal latches (U7 and U8). U3 emits data MSB first while U9 and U10 are clocked in parallel until all 16 bits have been shifted into place. Each time FSO goes high, a valid word at the shift register's output gets clocked into the latches. Additionally, the FSO signal is polled by the software to determine when to read these latches.

TIMEBASE

The DSP56ADC16S must be clocked at 128 times its data output rate (12.8 MHz for a 100-kHz conversion rate). Clocking the ADC slower results in the conversion rates shown in Table 1.

You may ask, "Why not just clock the ADC at its maximum rate, then store every nth conversion to achieve a lower conversion rate." Not a bad idea, but it does have one drawback. Part of the elegance of the sigma-delta architecture is that no anti-aliasing filter is needed because the signal is processed by an internal brick-wall FIR digital filter. From the Nyquist Sampling Theorem, we know that a reconstructed signal will contain all of the information present in the original signal as long as we

Listing l-Board *decoding is simplified by using a PAL instead of discrete logic.*

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

CHIP     ADDRESS-DECODE PAL16L8

;PINS     1    2    3    4    5    6    7    8    9    10
          A9   A8   A7   A6   A5   B4   B5   B6   B7   GND
;PINS     11   12    13   14   15   16   17    18    19   20
          AEN  E8253 A4   A2   IOW  IOR  COMP  E8255 DMA  VCC

EQUATIONS

COMP  = /(A7 :+: B7) * /(A6 :+: B6) * (/IOR + /IOW)

/E8253=A9 * /A8 * COMP * /(A5 :+: B5) * /(A4 :+: B4) * /A2 * /AEN

/E8255=A9 * /A8 * COMP * /(A5 :+: B5) * /(A4 :+: B4) *  A2 * /AEN

/DMA  = (/IOR + /IOW) + AEN
```

sample at slightly more than twice the bandwidth of the input signal. Since the clock rate alters the pass-band frequency of the brick-wall filter, the likelihood of aliasing the input signal is very small. If you simply "throw away" data to simulate a lower sampling rate, the likelihood of aliasing increases. Presumably, the reason one would desire a lower sampling rate is to diminish the data storage requirement. Throwing away data is still possible via software, as long as you understand the implications of possibly violating the Nyquist limit.

Although Motorola specifies a minimum clock frequency of 1 MHz (7.8125-kHz conversion rate), I have included the 6.25-kHz conversion rate because it was available "free." Even though, the SNR specification suffers, you may find this acquisition rate useful. This is also true for clock rates higher than 12.8 MHz supplied via the external input. We have experimented with frequencies higher than 20 MHz. Again, the SNR suffers, but the data may still prove useful.

Similarly, please note that there is nothing magical about a 12.8-MHz oscillator. For example, a 12-MHz oscillator would result in acquisition rates about 6% lower than those quoted in Table 1.

Preliminary data provided by Motorola indicated that a worst-case clock symmetry of 52.5/47.5 was allowed. I found this curious since their example circuit, which showed a simple crystal/inverter oscillator, was extremely unlikely to provide symmetry that tight. Motorola has since eased that specification to allow for a clock symmetry of 67/33 ratio. (Point of fact: A simple method of obtaining excellent symmetry is to start with twice the desired frequency and use a flip-flop to divide by two.) What's more important than clock symmetry, however, is a low-jitter clock source. Generally, ready-made 4-pin oscillators exhibit lower jitter than ones built from a crystal and inverters.

The various acquisition rates are generated by dividing down a 12.8-MHz oscillator via a 74LS393 dual 4-bit ripple counter (US). U5 produces 6.4-MHz, 3.2-MHz, 1.6-MHz, and 800-kHz square waves at its QA, QB, QC, and QD outputs, respectively. U4, a 74F 15 1 multiplexer, is used to select the appropriate clock rate and present it to the DSP56ADC16S's clock input.

EXTERNAL I/O

Access to the signals listed on the next page is provided via a 15-pin female D-shell connector. Where applicable, the signal levels are TTL compatible.

Base Address Hexadecimal	Jumper Settings A7	 A6	 A5	 A4	Possible Conflict
200	0	0	0	0	Game port
210	0	0	0	1	Expansion unit
220	0	0	1	0	Reserved by IBM
230	0	0	1	1	Reserved by IBM
240	0	1	0	0	Reserved by IBM
250	0	1	0	1	
260	0	1	1	0	précis factory setting
270	0	1	1	1	LPT2
280	1	0	0	0	
290	1	0	0	1	
2A0	1	0	1	0	
2B0	1	0	1	1	
2co	1	1	0	0	
2D0	1	1	0	1	
2E0	1	1	1	0	
2F0	1	1	1	1	COM2

Table 2-*The précis may be set up for any of 16 base port addresses, though not all potential addresses are without conflict.*

Pin l-External Reference In/Out: Permits you to monitor the 2-v reference built into the ADC. By removing jumper JP1, an external voltage reference may be injected on this pin. This facilitates using a more precise reference or simply a different reference voltage (maximum of +2 VDC).

Pin 2-Analog Supply In/Out: The +5-VDC analog supply to the ADC. It is isolated from the PC's switching power supply by U6, a 3-terminal regulator. You may use Pin 2 to provide power to external circuitry. (Maximum current draw is 200 mA.) By removing jumper JP4, the ADC may alternatively be powered from an external, linear power supply.

Pin 3-Serial Clock Out: Comes directly from the ADC. It is not buffered.

Pin 4-Frame Sync Out: Comes directly from the ADC. It is not buffered.

Pin 5—Serial Data Out: Comes directly from the ADC. It is not buffered.

Pin 6-General-Purpose TTL Input: Software accessible via port PC6 in the 8255.

Pin 7-General-Purpose TTL Output: Software accessible via port PC3 in the 8255.

Pin 8-External Clock In: If used, this input must be a TTL-compatible square wave. It is directed to the 74LS 15 1 and is selected via software as mux channel D7. It becomes the master clock for the 12.8-l MHz may be used to acquire data at a rate unavailable internally or to use a higher-stability/accuracy clock than the onboard oscillator(s).

Pins 9-l 5—Analog/Digital Ground.

Note: Since the SCO, SDO, and FSO signals are available at this connector, it is possible to use précis as an evaluation platform for the DSP56ADC16S even without a PC. Just remember to tie the appropriate mux select lines to provide a clock source.

ADDRESS DECODING

U1 is a 16L8 PAL. Although this PAL provides more functionality than is required, it is used in another ADC board we manufacture (see *Circuit Cellar INK* issues 13 and 15 for construction details regarding an 8-bit, 25-MHz digitizer called *étude*). A JEDEC fuse map for this PAL is available on the BBS if you would prefer not to work with the PALASM source code.

The PAL's job is to generate a chip-select signal for the 8255. It does this by comparing the précis's Base Address Select Jumpers with the PC's corresponding address lines. This occurs in parallel with decoding the PC's I/O signals A9, A4, A2, AEN, IOR, and IOW, according to the equations found in Listing 1.

Alterability of the base address allows you to locate précis at any one of sixteen base addresses and permits multiple boards in one computer. The précis board occupies four I/O port locations and may be mapped into specific base addresses in the range 200–2F0 hex (see Table 2). A read from or write to I/O space is "in range" when the following conditions are met:

1) address line A9 is high and A8 is low. This corresponds to a hexadecimal base address whose most-significant digit is 2.

2) the next-most-significant hex digit is determined by comparing A7, A6, A5, and A4 with précis's Base Address Select jumpers.

3) the least-significant hex digit is always zero. When A2 is high, the 8255 is selected. Specific registers within the 8255 are accessed by Al and AO. Note: A3 is not decoded—therefore the Base Address + 8 is redundant.

BUS INTERFACE

Interface to the system data bus is via U2, an 8255 programmable peripheral interface. Although some engineers do not advocate attaching NMOS LSI devices directly to the system bus, I have reliably used the method in several products. The advantage of connecting the 8255 directly to the bus is loading limit of two LSTTL loads).

The 8255 is mapped into four locations in the PC's I/O space. Port A is assigned as

Port	Direction	Use
		(DB-15
		General-purpose output (DB-15 pin 7)
PC2	output	MUX S2 (timebase select)
PC1	output	MUX SI (timebase select)
PC0	output	MUX SO (timebase select)

Table 3—*Port C on the 8255 is split between status inputs and control outputs.*

the converter's least-significant byte. Port B is assigned as the converter's most-significant byte. Port C is split into an input nybble and an output nybble. Bits 7-4 are "read-only," whereas bits 3-O are "write-only." Since it is not possible to perform nybble-wide I/O operations, a full 8 bits must be read from, or written to, an I/O port. Writing an 8-bit value to port C has no effect on its read-only portion. Likewise, when reading an 8-bit value from port C, bits 3-O are undefined. Individual bit definitions for port C are given in Table 3.

PCB

Although the digital portion of this design can, for the most part, be implemented in wire wrap, the analog portion requires a great deal more care. Originally, I prototyped the ADC and analog circuitry "dead bug style" over a copper ground plane. I made all component leads as short as possible and the circuit performed well.

However, the first version of the PCB did not work properly. A trace slightly over 5½ inches long connected the ADC (U3) clock input to the mux (U4) output. This proved to be too long for the edge-rate involved. Although I found several solutions to the problem, the simplest was to replace the 74LS151 with a 74F151 and replace the 5½-inch PCB trace with a piece of 75-Q coax.

In the second version of the PCB, I decided that changing the chip layout to position U4 extremely close to U3 would have required more effort than I cared for. Instead, I decided to etch the 75-Ω transmission line directly onto the PCB. Now, it is well known that empirical formulas for fabricating transmission lines in printed circuit boards exist (see Motorola's seminal work, the ECL data book]. Unfortunately-for two-sided boards at least—these formulas are based on having an "infinite" ground plane on the opposite side of the controlled-impedance trace. Since there was little room for any sort of ground plane above the trace, an "infinite" one was out of the question.

Fortunately, I found an easy solution. A friend of mine had recently begun slinging code for a purveyor of signal integrity software. My friend convinced his boss to use their software to solve for the impedance of a PCB transmission line based on its physical characteristics. Working together, it took only a few iterations before we specified the correct physical layout for a transmission line (having both its signal and return lines on the same side of the PCB} which yielded a 75-Ω differential impedance.

What we ended up with is probably best described as a two-dimensional coaxial cable. The center conductor was a 0.062" trace bounded on either sided by a 0.022" shield trace separated by a 0.008" space. The artwork was created this way with the expectation that after acid had etched the board, there would be a 0.020" ground trace, a 0.010" space, a 0.060" signal conductor, another 0.010" space, and finally a 0.020" ground trace. As I recall, we couldn't quite get down to 75 Ω in the available space, but the board worked beautifully nevertheless.

SOFTWARE

Complete source code, in Pascal, for a crude digital storage oscilloscope can be found on the BBS. Listing 2 defines all of the crucial hardware-specific routines. I chose to show the examples in Pascal because it seemed like a good compromise for those who favor C and those who favor BASIC.

Acquiring data with précis is extremely simple. Programming consists of three principal functions: configuring the 8255, selecting a clock source via the multiplexer, and reading the ADC.

When the PC is booted, all 8255 ports are set to the high-impedance state. This protects the hardware until the procedure AssignRegisters can configure the 8255. Writing 9A (hex) to the 8255's control port (Base Address + 7) configures the 8255 properly.

A clock source is selected by writing the appropriate bit pattern to the 74F15 1 multiplexer through port C. This is detailed in the procedure MUX_ChannelSelect.

The application program must have access to the data whether it is

Listing 2—Since using interrupts to service précis would require a great deal of overhead, polling is the besf way to acquire data.

```
procedure AssignRegisters(BaseAddress: word);

{     8255 port Address    Definition                    Direction

      Port A     base + 4 ADC least significant byte     input
      Port B     base + 5 ADC most significant byte      input
      Port C     base + 6 Bits 7 3       status          input
      Port C     base + 6 Bits 4 0       control         output
      Control    base + 7 8255 mode register         not applicable

  When passed the Base Address, this procedure sets the global
  addresses of hardware registers relative to the Base Address and
  initializes the 8255
}
begin
  ADCLo     := BaseAddress + 4; { 01 ???? 0100 }
  ADCHi     := BaseAddress + 5;  { 01 ???? 0101 }
  LatchC   : = BaseAddress + 6; { 01 ???? 0110 }
  Ctrl      := BaseAddress + 7;  { 01 ???? 0111 }

  port[Ctrl] := $9A;{ 1001 1010 A=in, B=in, Chi=in, Clo=out}

  BitsC := $0F;
  port[LatchC] := BitsC
end:

procedure MUX_ChannelSelect(Channel: byte);

{ Direct a clock source to the ADC via the 74F151}

begin
  BitsC := BitsC and $F8;{ 1111 1000 MUX bits PCO, PC1, PC2 }

  case Channel of
    0 : BitsC := BitsC or $F0;{ DO = xxxx x000 6.25   kHz
    1 : BitsC := BitsC or $F1;{ D1 = xxxx x001  12.5   kHz
    2 : BitsC := BitsC or $F2;{ D2 = xxxx x010 25      kHz
    3 : BitsC := BitsC or $F3;{ D3 = xxxx x011 50      kHz
    4 : BitsC := BitsC or $F4;{ D4 = xxxx x100 100     kHz
    5 : BitsC := BitsC or $F5;{ D5 = xxxx x101 User Osc.
    6 : BitsC := BitsC or $F6;{ D6 = xxxx x110 User Osc. / 2
    7 : BitsC := BitsC or $F7; { D7 = xxxx x111 External Clock
  end;

  port[LatchC] := BitsC
end:

{ This code fragment actually does all of the work.

   The commented-out code shows how to call the assembly language
   subroutine which performs the same function, albeit faster.

(*
   MACHINE(DataBuffer, LatchC, ADCLo, LastBufferElement);
*)
  Pass := 0:
  repeat
    repeat { Wait for PC7 to go HI ==> vaid data }
    until Port[LatchC] and $80 = $80;

    DataBuffer[Pass] := portw[ ADCLo ];
    inc (Pass);

    repeat { Wait for PC7 to go low }
    until Port[LatchC] and $80 <> $80;
  until Pass = LastBufferElement;
```

processed in real time or stored in a buffer. Since the overhead required to perform 100,000 interrupts per second would be extremely high, the code fragment shown in Listing 2 simply polls the ADC until a valid word is available. Each time the converter's Data Ready flag (hardware signal called Frame Sync Out) goes high, a valid word exists at the shift register's outputs. A new value is latched each time Data Ready goes high, and this value must be read in real time. To prevent rereading the same value because Data Ready was still high after that value had been processed, it is necessary to wait for Data Ready to go low before reading the next valid word. Another way to think of this is that the polling routine must be "edge", rather than "level" sensitive.

The Pascal implementation works fine on 10-MHz 80286 O-wait-state [or faster) machines. Slower machines will require an assembly language subroutine like the Pascal-specific one shown in Listing 3.

Finally, it is well known that Intel processors use a byte-swapped memory architecture. The architecture is often called "little-endian" because it employs both reverse byte ordering and reverse bit ordering. By contrast, the Motorola 680xx architecture is big-endian with respect to bytes, but little-endian with respect to bits. However, it is perhaps less well known that 80x86 processors follow the Motorola scheme as far as I/O operations are concerned. As far as précis is concerned, that's why the 8255's port A gets the ADC's LSB and port B gets the MSB.

Reiterating, although 16-bit data is stored in the PC's memory in byte-reversed order, 16-bit data read from and written to I/O ports is not. This means for I/O operations, the lower (numeric) address byte contains the least-significant data byte in a 16-bit word.

So start building that board, downloading that software, and collecting some data.

Listing 3—*While low-/eve/routines implemented in Pascal work fine on relatively fast machines, slower ones require fhe use of assembly language.*

```
                .MODEL          TPASCAL
                .CODE
Machine         PROC            FAR

DataBuffer:WORD,LatchC:WORD,ADCLo:WORD,LastBufferElement:WORD

                PUBLIC          Machine

                MOV             BX,[DataBuffer]
                MOV             CX,LastBufferElement
                MOV             DX,LATCHC

WAIT1:          IN              AL,DX
                AND             AL,80H
                JZ              WAIT1

                MOV             DX,ADCLo
                IN              AX,DX
                MOV             [BX],AX
                INC             BX
                INC             BX

                MOV             DX,LATCHC
WAIT2:          IN              AL,DX
                AND             AL,80H
                JNZ             WAIT2

                LOOP            WAIT1
                RET
Machine         ENDP
CODE            ENDS
                END
```

J. Conrad Hubert is a principal in Deus Ex Machina Engineering Inc., where he provides consulting services for the development of hardware and software for embedded systems, data acquisition, and digital signal processing. He may be reached at (612) 645-8088.

CONTACT

Motorola, Inc.
P.O Box 20912
Phoenix, AZ 85036

Crystal Semiconductor Corp.
4210 S. Industrial Dr.
Austin, TX 78744
(512) 445-7222
Fax: (512) 455-7581

Texas Instruments, Inc.
9301 Southwest Fwy.
Commerce Park, Ste. 360
Houston, TX 77074
(713) 778-6592

Analog Devices
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9 106
(617) 329-4700
Fax: (617) 326-8703

SOURCE

précis [pray-see] nf. exact

Deus Ex Machina Engineering, Inc.
1390 Carling Dr., Ste. 108
St. Paul, MN 55108
(612) 645-8088
Fax: (612) 645-0184

1. Assembled & tested board with source code in C, BASIC, and Pascal (includes FedEx economy shipping) $354.00
2. Printed circuit board only (includes Priority Mail shipping) $99.00

Visa and Mastercard accepted.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

Calibrating Seismic Velocity Transducers with précis

An A/D converter board in hand is worthless without having something to use it for. Chris picks up where J. Conrad left off in the previous article by applying the précis ADC board to a real application.

FEATURE ARTICLE

Christopher J. Peoples

Seismic velocity transducers are used to convert mechanical ground motion into an electrical signal. Exploration geophysicists use these devices to sense seismic wave energy that has been reflected and/or refracted off layers within the earth's interior. A seismic velocity transducer, more commonly known as a *geophone* (Figure 1}, will generate a voltage that is proportional to the rate at which the ground displacement changes with time (i.e., it senses the ground's velocity and not the ground's displacement). Inside the geophone, differential motion between a spring-suspended coil and a fixed magnet generates the voltage that is proportional to the ground motion. This electrical signal is then digitized and recorded for later processing. Using data from an array of these sensors, geophysicists process the data to enhance and identify subsurface features that may contain oil and gas.

Basic to exploration seismology is the need for improved seismic resolution of deeper layers that reflect and refract seismic waves. Signals in the seismic bandwidth range of1–60 Hz, and a major goal is to increase the seismic bandwidth to a few hundred hertz. To improve seismic resolution, you must begin with a better understanding of the sensing device used to record seismic waves.

For the research of my Master of Science thesis at Texas A&M University, my advisors and I developed the theory and a prototype system for calibrating geophones *in situ*. Using parameters derived from the calibration process, I can model the broadband amplitude and phase response for a planted geophone when it acts as a seismic wave sensor. I present here a brief description of the geophone/earth-coupling phenomenon and how I used the précis analog-to-digital data converter card (*see* the article describing précis starting on page 28 of this issue) to measure a geophone's broadband amplitude and phase response.

GEOPHONE/EARTH-COUPLING PHENOMENON

Early seismic explorers noted that at high frequencies (100 Hz and above),

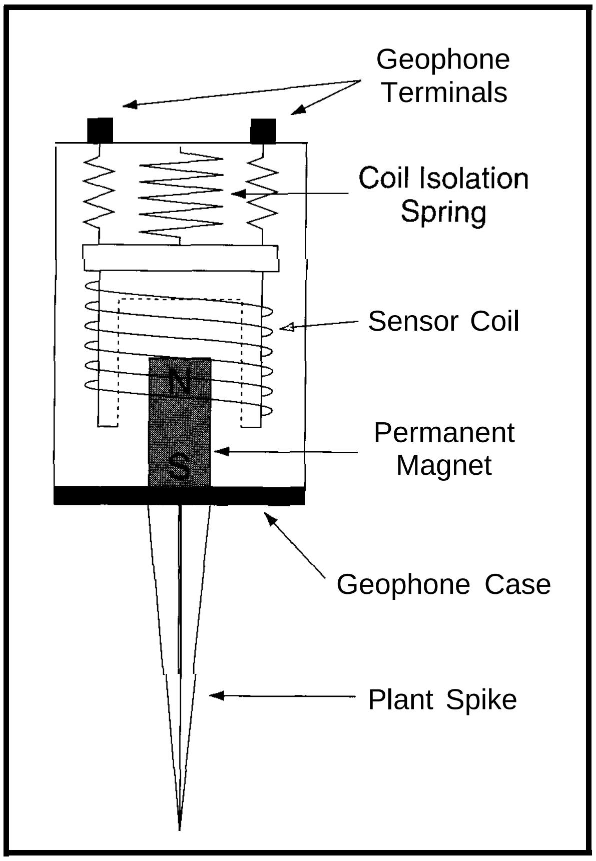


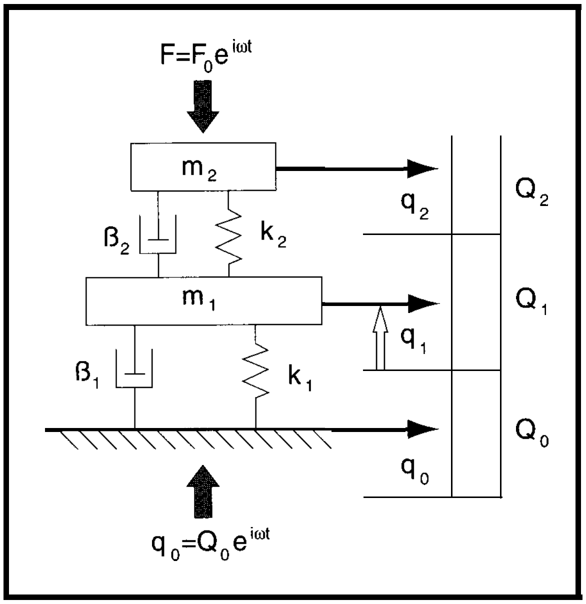
Figure 1-A *geophone, or seismic velocity transducer, generates a voltage that is proportional to the rate at which the ground displacement changes with time.*

the motion of the geophone did not necessarily follow the ground motion of the earth itself, and they attributed the causes of these departures to coupling effects between the geophone and the earth. Poor coupling results when the geophone's plant spike fails to develop good cohesion between it and the soil in which it is planted (as happens when planted in loose sand). This effect causes a high-frequency resonance, which in turn acts as a low-pass filter limiting seismic resolution at frequencies above the coupling-related resonant frequency. In firmer soils, where the cohesion between the geophone and the soil are good, coupling problems are not as apparent (Krohn, 1984).

Several studies (e.g., Washburn and Wiley, 1941; Hoover and O'Brien, 1980; and Krohn, 1984) have demonstrated that the geophone/earth-coupling phenomenon resembles a damped resonant system that can be modeled using a system of two simple harmonic oscillators stacked in series (i.e., a compound harmonic oscillator, see Figure 2). The smaller mass (m,) represents the geophone coil coupled to the geophone magnet and case assembly (m,) by a spring (k_2) and dashpot (β_2). The coupling of the geophone to the earth is described using an additional spring (k,) and dashpot (β_1) combination. The mechanical-to-electrical transfer of energy (geophone transduction) is assumed linearly proportional to the velocity difference between the geophone coil and the case-mounted magnet.

The amplitude and phase response for the differential motion between the masses, m_1 and m_2, corresponding to the geophone coil and case-mounted magnet, respectively, is dependent on how the mechanical system is driven. Solving the differential equations for the case where the system is driven from the ground direction by simple harmonic ground displacement ($q_0 = Q_0e^{i\omega t}$) yields the predicted geophone response to a seismic impulse (see graphs in Figure 3). The solid line represents a poorly coupled situation where the coupling effect is underdampened (i.e., a low β_1 value), while the dashed line shows a better-damped condition indicating somewhat better coupling between the geophone and the earth.

Figure 2—The geophone/earth-coupling behavior can be modeled by a compound harmonic oscillator system.



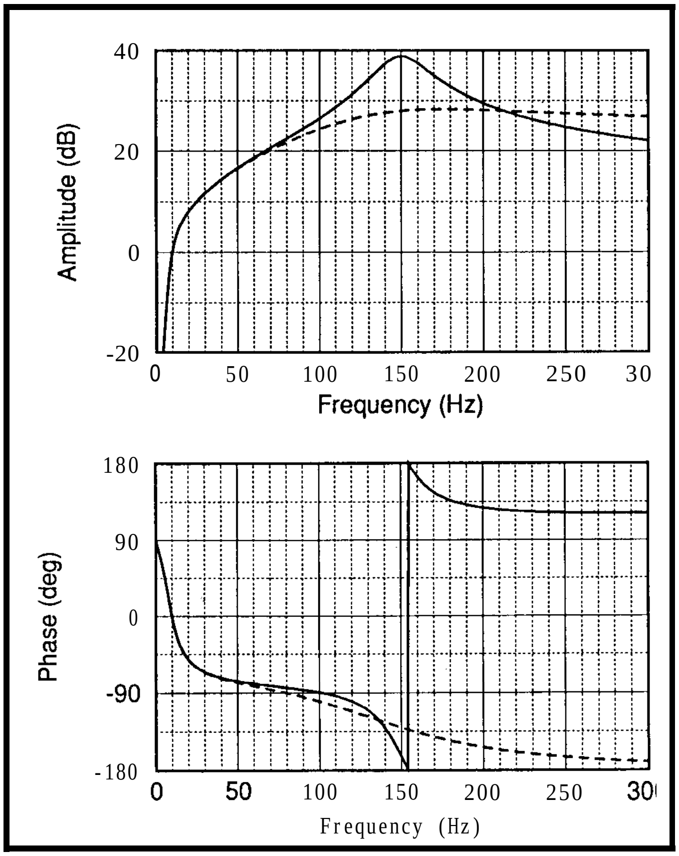


Figure 3—Modeled amplitude and phase response of a geophone for the case when it is driven by simple harmonic ground motion for both poor geophone/earth-coupling (solid curves, i.e., low β_1) and good coupling (dashed curves).

A very different response becomes apparent (Figure 4) when a simple harmonic forcing function ($F = F_0e^{i\omega t}$) is applied to the modeled geophone's mass, m_2. In this case, we are modeling the application of an electrical impulse to the geophone coil, assuming a bidirectionally linear electromechanical system. The peak amplitude response corresponds to the geophone's own resonant frequency (10 Hz), while at the coupling response frequency (150 Hz), a notch appears. This notch is caused by the geophone/earth-coupling effect, where it acts as a mechanical vibration damper. Higher degrees of damping (that is, increased values of β_1) mitigate this mechanical vibration damper effect, making it unapparent.

CALIBRATION SYSTEM DESIGN

The geophone calibration system I developed is a prototype, designed primarily for testing my geophone/earth-coupling theory. I chose to develop the calibration system using an off-the-shelf technological approach by fitting my 80286 MS-DOS personal computer with a digital data acquisition card. In selecting a data acquisition card for this project, I had to overcome many of the problems common to engineering, the primary being price versus performance. For my research, I needed a data acquisition card with 16 bits dynamic resolution and a least-single-bit resolution of around 50 µV. To determine the geophone's response at high frequencies (a maximum of approximately 500 Hz], I needed to do data sampling at about 2000-4000 samples per second (Sps).

I ended up selecting the précis data acquisition card because it met several of the design criteria, and it offered several features that eased the development of the calibration system. In addition to that card, the calibration system also included a battery-powered external unit that functions as a high-input-impedance buffer/amplifier and a calibration pulse amplitude controller. The calibration system is schematically shown in Figure 5.

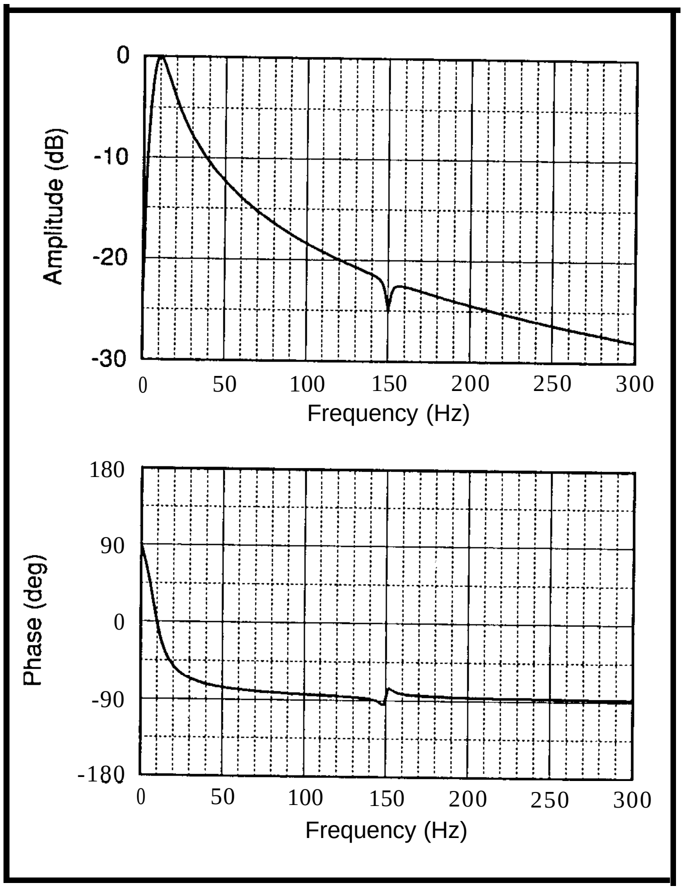


Figure 4—*Poorly coupled geophone amplitude and phase response for the case when a simple harmonic forcing function is applied to the geophone coil. The trough located at 150 Hz is caused by coupling behavior, where the lower mass (m₁ in Figure 2) acts as a mechanical vibration damper.*

The précis board solved several calibration system design problems. First, the card acquires data using sigma-delta modulation, which eliminates the need for a sample-and-hold amplifier and an antialias filter. Data sampling rates are software selectable at 12500, 25000, 50000, and 100,000 Sps, with the cutoff frequency located at 45.5% of the chosen sampling rate. Even though my bandwidth of interest for calibrating geophones isl-500 Hz, the cutoff frequency for the précis's slowest data acquisition rate of 12500 Sps is still above 5000 Hz and, therefore, did not affect my measurements.

Signal input to the précis is via a BNC connector and can have a range of -2 to 2 V, and because it has a 16-bit analog-to-digital converter (i.e., the Motorola MC56ADC16S), it has a least-significant-bit resolution (LSB) of 61 µV. Also incorporated into précis's design are several external connections available through a DB- 15 female connector, among them TTL-compatible general-purpose input and output ports. The calibration system uses the TTL output port to provide the pulse function (discussed below) to the geophone under test.

The only drawback in the précis's design is its 2 kW input impedance. Because a geophone has a nominal impedance of 250-300 Ω, this low-valued input impedance may adversely distort the geophone's transient response by excessively damping the geophone. To increase the input impedance, I placed an Analog Devices AD524 instrumentation op-amp between the geophone and the précis signal input (Figure 6, upper portion). I chose this op-amp because it has an input impedance of 1 GΩ; pin-selectable gains of 1, 10, 100, 1000; and it has a flat response from DC to 1000 Hz-well above the bandwidth of interest for calibrating geophones. During the calibration tests, the transient response (signal-to-noise ratio) of the geophone tested was sufficiently strong to only need the AD524 as a unity-gain amplifier.

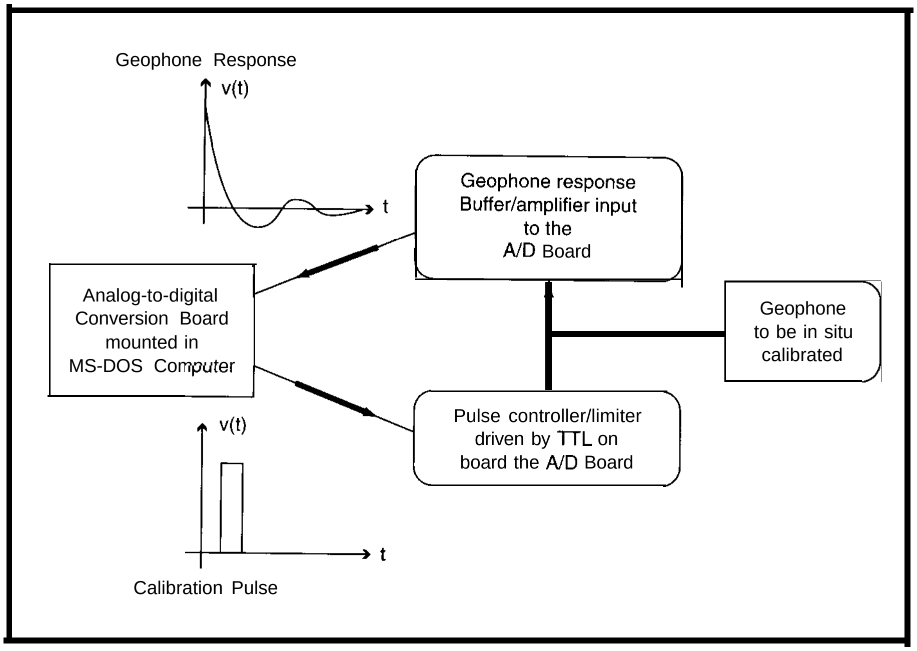

Geophone Response
v(t)
t
Geophone response Buffer/amplifier input to the A/D Board
Analog-to-digital Conversion Board mounted in MS-DOS Computer
Geophone to be in situ calibrated
Pulse controller/limiter driven by TTL on board the A/D Board
v(t)
t
Calibration Pulse

Figure 5—*System schematic of the geophone calibration system developed for this research.*

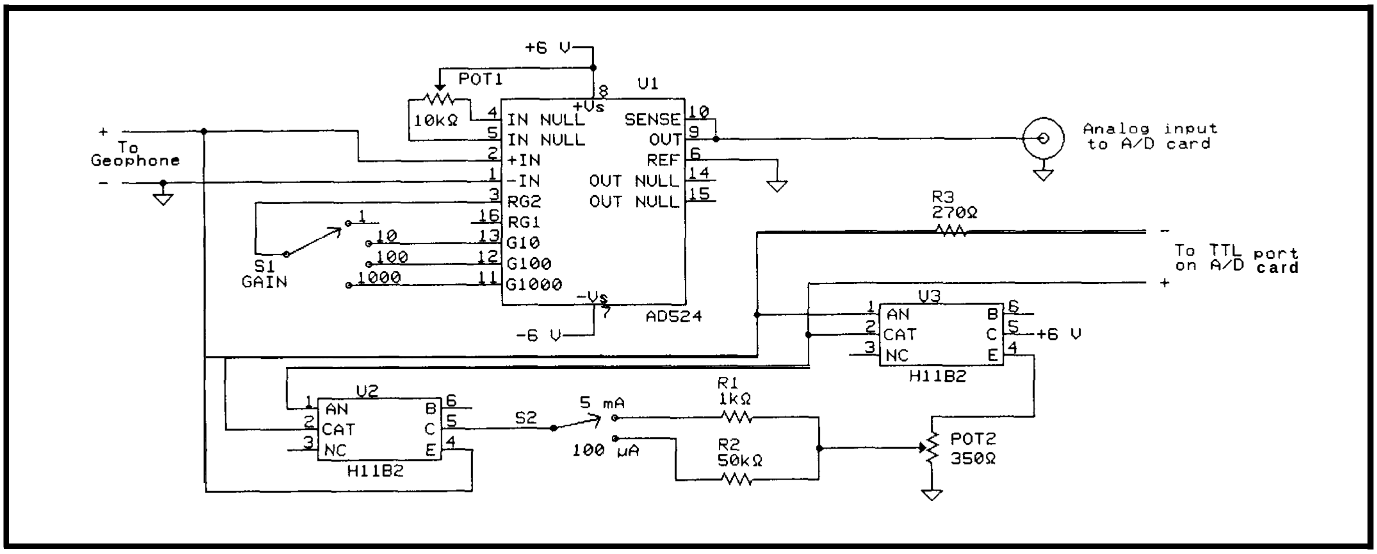
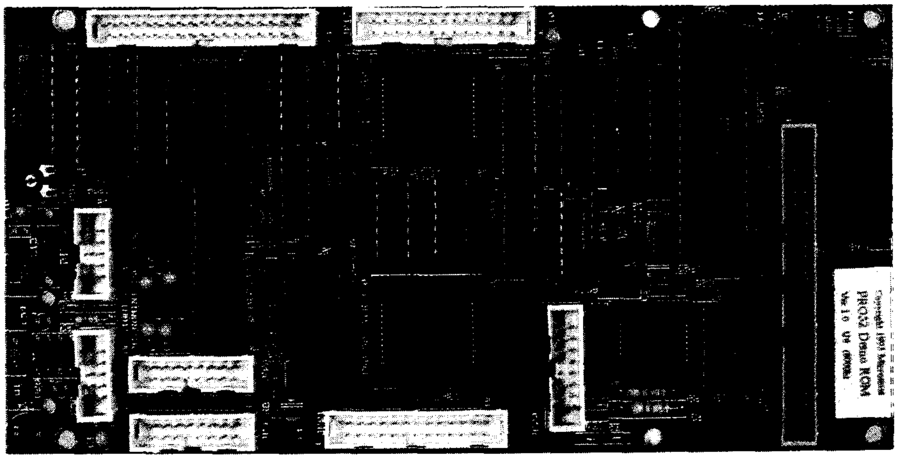


Figure 6—To increase the input impedance, an Analog Devices AD524 instrumentation op-amp is used between the geophone and the précis signal input. The external unit a/so functions as a calibration-pulse amplitude controller.

In addition to being a high-input-impedance amplifier, the external unit also functions as a calibration-pulse amplitude controller (Figure 6, lower portion). It limits the geophone's calibration-pulse current to a switch-and-potentiometer maximum of either 5 mA or 100 µA. The two different current supplies were originally intended so the calibration system could be used to perform both impulse and step-pulse calibration testing (discussed below). The calibration-pulse signal is generated using the data acquisition and control software to cycle the précis's TTL output port on and off.

I accomplished software timing control for the calibration-pulse duration using the PCHRT (V3.0) High-resolution Timer Toolbox by



Ryle Design Inc. To prevent damaging both the ADC card and the computer during the calibration process, I passed the calibration-pulse signal through a optoisolator (Motorola H11B2) in the external unit. Thus, the calibration-pulse current is drawn from the external unit's battery power supply (a pair of 9-V batteries stepped down to 6 V) instead of the computer. I added the second H11B2 (labeled #2 in Figure 6) after the initial testing of the calibration system revealed there was an additional inadvertent path to ground, causing unwanted additional damping of the geophone's transient response.

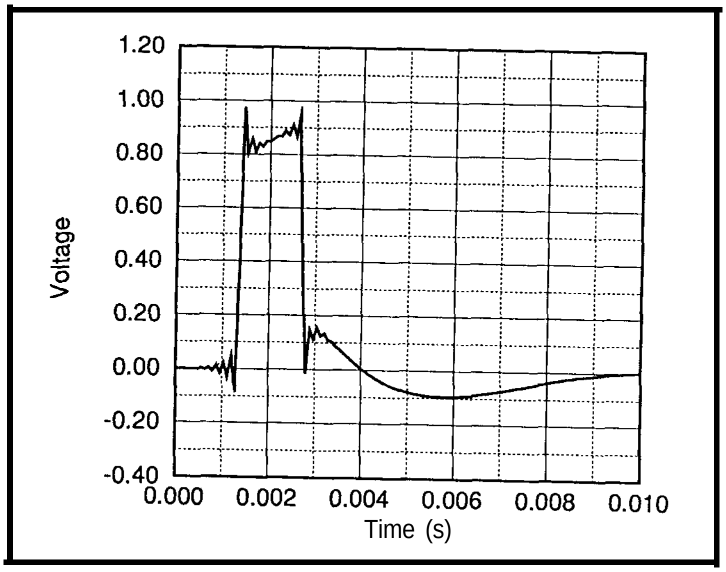


Figure 7—*Pulse-calibration-record beginning, showing the* applied calibration pulse (for a *pulse duration of2 ms) and the onset of the geophone transient response.*

CALIBRATION THEORY, PROCEDURE, AND RESULTS

The calibration system as developed is capable of performing both step-pulse- and impulse-type calibrations. Step-pulse calibrations are performed by using the data acquisition and control software to switch the précis's TTL output port on, and then waiting a few seconds for the geophone to stabilize before switching the TTL output port off. After switching off the TTL output port, the software starts recording the geophone's transient response. Step-pulse calibrations were performed in the initial testing of the calibration system, but were not included on the research into geophone/earth-coupling because this technique limits resolution at higher frequencies. A step-pulse has a 1/f amplitude decay with increasing frequency, and when passed through a convolution with the geophone's transient response, the high-frequency resolution is severely impaired [Chapman et al., 1988). Using an impulse-type of calibration permits better resolution of a geophone's high-frequency response.

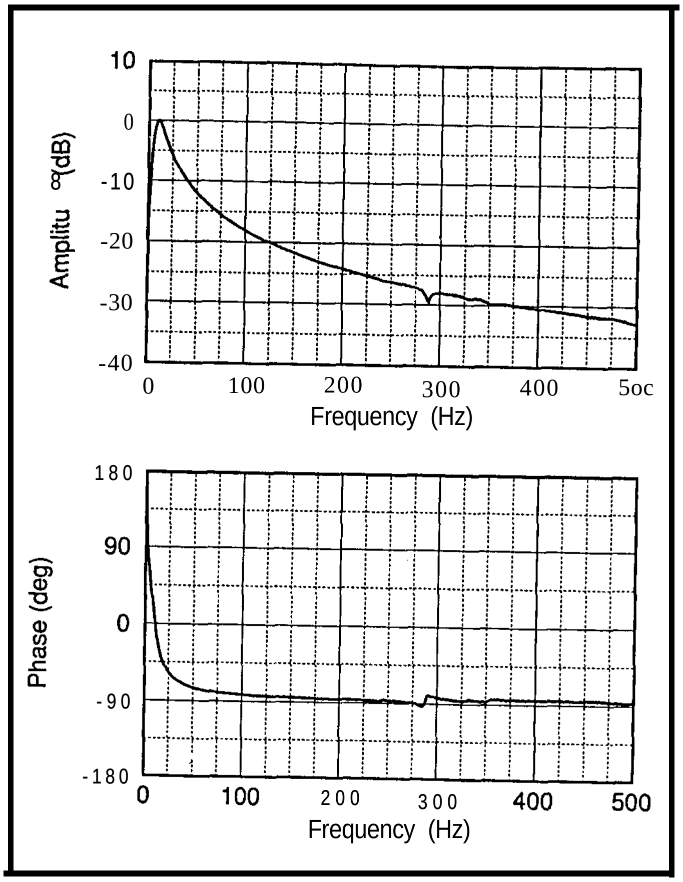


Figure 8—*Amplitude and phase response for a 10-Hz geophone planted in loose sand, with a poorly damped coupling-related resonance located at* 288 *Hz.*

Chapman et al. (1988) demonstrate that impulse calibration provides more power at higher frequencies, yielding an improved signal-to-noise ratio for the geophone's transient response. Ideally, a Dirac delta function [a "spike" of energy that has infinite height and infinitesimal duration) would be used to impulse calibrate the geophone. Physically, it's not possible to generate a pure Dirac delta function, but as Chapman et al. (1988) show, a reasonable approximation can be made by using a short-duration rectangular function. In making the duration of the rectangle function small, it has a near flat spectral response over the bandwidth of interest, and for all intents and purposes can be looked upon as a "bandwidth-limited" delta function. A short-duration pulse offers more energy at high frequencies, at the expense of a degraded signal-to-noise ratio.

Conversely a longer-duration pulse delivers more energy, increasing the signal-to-noise ratio at the expense of decreasing high-frequency resolution. Impulse calibrations were done using a process similar to that used for a step calibration, except I significantly shortened the time period the TTL port was left. The data acquisition software, as programmed, allows for four different pulse durations ranging from 0.62 to 2 ms (I controlled pulse duration timing using the PCHRT software that I discussed above). Initial testing revealed that pulse durations of 1-2 ms are sufficient for analyzing geophone/earth-coupling behavior. Calibration system tests also revealed that a recording period of five times the natural resonance frequency of the geophone under test (e.g., 0.5 s for a 10-Hz geophone) proved to be ample. Longer recording periods only served to incorporate additional noise.

A major concern about the calibration process is the coordination of the onset of data acquisition with the end of the applied calibration pulse. The data acquisition software starts the calibration procedure by first turning the TTL output port on, and then waiting the specified duration before switching it off. After that, data acquisition commences. Any delay between the two actions would cause an error in the amplitude and phase determination. As it turns out (rather serendipitously), this is not a problem because the optoisolators (two Motorola H11B2s) used in the external unit have a slow response and cause an approximate 1.2-ms delay. The data acquisition software executes faster and is capable of recording the calibration pulse in addition to the geophone transient response (Figure 7). Prior to determining (i.e., Fourier transforming) the geophone's amplitude and phase response to the calibration pulse, the pulse is deleted from the recorded data. The ringing in the calibration pulse is possibly by either Gibb's phenomenon caused by the digitization of the pulse by the précis or some artifact caused by the optoisolators.

I calibrated a variety of geophones with different natural resonance frequencies under similar conditions to examine their geophone/earth-coupling behavior. The geophones I used for testing the calibration theory and system consisted of ones with natural resonance frequencies of 10, 40, and 100 Hz.

Planting the geophones in a large bucket of loose sand and performing the calibration tests as described above revealed a coupling-related resonance in the vicinity of 285 Hz for the three geophones tested. Figure 8 shows the amplitude and phase response for the 10-Hz geophone with its coupling resonance located at 288 Hz. Each geophone has a slightly different coupling resonance frequency because of small differences in mass between the geophones.

I also performed calibration tests on the same geophones when they were planted in soils that were significantly more consolidated. Because of the greater cohesion between the geophone plant spike and the soil, the dampening was greater and the coupling phenomenon was not as pronounced and, thus, not observable.

CONCLUSION

I developed the calibration theory and system presented here to gain a better understanding of a geophone's ***in situ*** amplitude and phase behavior and the deleterious effects caused by geophone/earth-coupling. Based on a better understanding of these effects, exploration geophysicists can predict and adjust their seismic surveying practices to mitigate these effects. The calibration system design is probably not optimal, but it did succeed in demonstrating and proving the theory developed for describing the phenomenon.

The précis data acquisition card performed admirably throughout the project, recording accurate and precise measurements of each geophone's transient response to a calibration pulse. The only drawback in using the précis is that its slowest data sampling rate is 12,500 Sps, which yields a cutoff frequency well above 5000 Hz, and well above any frequency relevant to seismic exploration (approximately 1-500 Hz).

The external amplifier/pulse controller could have been designed for better performance, but given the time and monetary constraints present at the time, it was sufficient. For the most part, this project is a success because it did establish and demonstrate the theory on geophone/earth-coupling (in addition to contributing to the completion of my Master of Science research).

Special thanks to my thesis research advisors, Drs. Steven H. Harder and Anthony F. Gangi, and to J. Conrad Hubert of Deus Ex Machina Engineering Inc.

Christopher J. Peoples is presently a part-time community college physics instructor in Southern California. In addition to his obvious interests in geophysical data acquisition and processing, he is also interested in industrial process control and automation. He may be reached at *geofizman@aol.com.*

REFERENCES

Chapman, M.C.; Snoke, J.A.; and Bollinger, G.A., 1988, "A procedure for calibrating short period telemetered seismograph systems": Bull. ***Seis.*** *Soc.* ***Am., 78,*** no. 6, 2077–***2088.***

Hoover, G.M. and O'Brien, J.T., 1980, "The influence of the planted geophone on seismic land data": ***Geophysics, 45,*** no. ***8, 1239-1253.***

Krohn, C.E., 1984, "Geophone ground coupling": ***Geophysics, 49,*** no. ***6, 722-731.***

Menke, W.; Shengold, L.; Hongshen, G.; Ge, H.; and Lerner-Lam, A., 1991, "Performance of the short period geophones of the IRIS/PASSCAL array": *Bull.* ***Seis.*** *Soc.* ***Am., 81,*** no. 1, ***232-242.***

Washburn, H. and Wiley, H., 1940, "The effect of the placement of a seismometer on its response characteristics": ***Geophysics, 6,*** 116-13 1.

SOURCES

Timer Toolbox
Ryle Design, Inc.
P.O. Box 22
Mt. Pleasant, MI 48804
(517) 773-0587

Deus Ex Machina Engineering, Inc.
1390 Carling Dr., Ste. 108
St. Paul, MN 55108
(612) 6458088
Fax: (612) 645-0184

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

FEATURE ARTICLE

Barry Rein, Esq.

Copyrights and Patents for Protecting Software

Recent Progress in Finding the Proper Balance

"Can I protect my code?" Developments in copyright and patent law suggest that finally a slightly definitive "yes" is coming through.

Ohe last year has seen significant developments in both copyright and patent law since both were made applicable to computer software at the beginning of the last decade. In 1992, ***Computer Associates v. Altai*** in the Second Circuit and ***Brown Bag Software v. Symantec Corp.*** in the Ninth markedly reduced the scope given to software copyright, specifically rejecting the *Jaslow* broad brush of as "too facile." Significantly, ***Apple v. Microsoft and Hewlett Packard*** (also with the Ninth Circuit) adopted a similar approach to user interfaces. However, with ***Lotus v. Borland*** in the First Circuit, which seemed to be following a similar approach, we were taken a step backward. In this case, Judge Keeton held that Lotus's macro language is protectable by copyright, and that Borland's use of it was an infringement. Keeton's judgment is now awaiting decision by the First Circuit where it was heard by a panel including soon-to-be Justice Breyer.

Although the court's position on infringement has developed, it is still not without surprises. Within the last few years, ***Sega Enterprises Ltd. v. Accolade Inc.*** in the Ninth Circuit and ***Atari Games Corp. v. Nintendo of America Inc.*** in the Federal Circuit, interpreting Ninth Circuit law, have established a right under copyright law to decompile or reverse engineer software to extract ideas and related elements which cannot be protected to produce non-infringing products. In 1992, Arrhythmia v. ***Corazonix*** lent greater rationality to the law on when software constitutes statutory subject matter for a patent. However, the law is not free of mysterious anomalies as this year's ***In re Schrader*** indicates.

The net result is that we are much closer to relegating the patent and copyright regimes to their proper constitutional roles, undoing the damage done principally by the inability of the patent system to embrace the new science of software in a timely fashion. We have come a long way, finally closing in on where we should have been at the outset.

COPYRIGHT PROTECTION FOR SOFTWARE

The current analytical framework for evaluating the scope of protection for software by the copyright law is best set forth in ***Computer Associates v. Altai*** [1], decided June 22, 1992 and amended December 17, 1992. ***Computer Associates*** developed specific applications which could run on a number of different operating systems through the use of a so-called "adapter module." Rather than rewriting the particular application to make it compatible with each different operating system, Computer Associates divided the application into two parts, one constituting the application proper and the other, the adapter module, constituting the coupling between the application and each particular operating system. Thus, to rewrite an application for a different operating system, it was only necessary to rewrite its adapter module.

Altai hired Arney, a Computer Associates programmer, who suggested that Altai also use adapter modules. What he did not tell Altai, according to the facts established at trial, was that he had brought with him purloined adapter code from Computer Associates and was using it to write the Altai programs. Altai found this out when Computer Associates brought suit, and immediately stopped selling the accused programs. However, Altai then rewrote the programs from unprotected functional specifications using programmers who were not privy to Computer Associates code. Altai admitted that its first version of the programs was an infringement of Computer Associates' copyright, but denied that its second "clean room" version infringed.

The case establishes a model for understanding the various levels within software and software development which lends itself to copyright analysis. I will summarize the bulk of the court's model which, by the way, the court said was not applicable to categorically distinct works such as screen displays:

The Copyright Act defines a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result" (17 U.S.C. § 101). In writing these directions, the courts presume that a programmer works "from the general to the specific."

The model begins by identifying a program's ultimate function or purpose. The model then maintains that a programmer "decomposes" or breaks the purpose down into "simpler constituent problems or 'subtasks' . which are also known as subroutines or modules." Sometimes, depending on the complexity of its task, a subroutine may be broken down further into nested subroutines.

Having sufficiently decomposed the program's ultimate function into its component elements, within this model, a programmer then arranges the subroutines or modules into organizational or flow charts which map the interactions between modules which achieve the program's end goal.

To accomplish these intraprogram interactions, a programmer must carefully design each module's parameter list. According to expert Dr. Randall Davis, appointed and fully credited by the district court, a parameter list is "the information sent to and received from a subroutine." The parameter list also includes the form in which information is passed between modules and the information's actual content. The courts recognize that interacting modules must share similar parameter lists so that they are capable of exchanging information.

According to the model, "The functions of the modules in a program together with each module's relationships to other modules constitute the 'structure' of the program." Additionally, the term "structure" may include the category of modules referred to as "macros" which the courts define as "a single instruction that initiates a sequence of operations or module interactions within the program." They note that users frequently accompany a macro with an instruction from the parameter list to refine the instruction. For example, a macro would give a current total of accounts receivable, but the parameter list would limit the search to the specific range of 8/91 to 7/92 and customer number.

❑ ***We are much closer to relegating the patent and copyright regimes to their proper cons titutional roles, undoing the damage done principally by the inability of the patent system to embrace the new science of software in a timely fashion.***

With this structural concept in mind, the court adopted a three-step procedure, which it refers to as the *Abstraction-Filtration-Comparison* test, a test used to determine substantial similarity when a dispute involves the nonliteral elements of computer programs. The first step is abstraction, an analytical method first enunciated by Learned Hand in *Nichols v. Universal Pictures Co.* in *1930.* Hand maintained that abstraction involves superposing a number of patterns or levels of abstraction with increasing generality on a particular work (more and more specifics are left out with greater abstraction). The court must then determine the level at and above which protectable expression ends (otherwise authors could monopolize ideas). Clearly, at the highest level, everything constitutes idea and, at the lowest, there is a great deal of individual expression. It is usually at the same intermediate level that a court will find the transition from the unprotected to the protected. The level is one that must be determined in each case based largely on the public policy purposes of the copyright law rooted in Article 1 Section 8 of the Constitution.

The *Whelan* case also used a rudimentary abstractions test to separate idea from expression, but did so with a butcher's knife, rather than a surgeon's scalpel. In Whelan, upheld by the Third Circuit in 1986, a software developer wrote a program for operating a dental laboratory and, subsequently, left to write a different program in another programming language which the owner of the first program asserted to be an infringement of his copyright. The court agreed that it was. This case has long been criticized for concluding that the only idea in this case involved the operation of a computer to support a dental laboratory and that all else (i.e. everything at lower levels of abstraction) constituted protected expression. In contrast, Altai's characterization of programs recognizes that software usually consists of many ideas and expressions related to its modular and functional nature and which, therefore, should not be protected.

To execute the abstraction step, a court must dissect the structure of the accused program and isolate each level of abstraction, beginning at the level of code and ending with the program's ultimate function. The court recommended doing so by viewing a program as a hierarchy of functional modules. At each higher level of abstraction, the instructions in lower level routines are replaced conceptually by the modules' functions until the ultimate function of the program is finally reached. At low levels of abstraction, a program's structure is usually very complex; at the pinnacle, it is trivial.

In filtration, the second step in the *Altai* analysis, the program components are evaluated at each level of *abstraction* to determine whether, at that level, the specific program component constitutes an idea or an expression. Importantly, not only ideas are to be filtered out, but also unprotectable expression, including elements "dictated by considerations of efficiency, so as to be necessarily incidental to idea," "required by factors external to the program itself," and "taken from the public domain" (982 F.2d at 707).

Filtration should remove uncopyrightable subject matter so that what remains is a core of protectable expression. Focusing on the individual elements of the program at each level, the process is sometimes referred to as "analytic dissection." The following discussion addresses what kinds of elements are to be eliminated (i.e., are *not* protectable expression).

- those dictated by efficiency:

An clement is deemed dictated by efficiency and hence nonprotectable when there is essentially only one way to accomplish the task within a computer framework. According to the merger doctrine, "a court must inquire whether the use of this particular set of modules is necessary to efficiently implement that part of the program's process. If the answer is 'yes,' then the expression represented by the programmers' choice of a specific module or group of modules has merged with their underlying idea and is unprotected." Significantly, Altai used this argument in their defense recognizing that, although many ways may theoretically exist to accomplish a particular task, only a few of them do so in a realistically efficient manner, and this lack of alternatives may warrant application of the merger doctrine.

The utilitarian nature of programs justifies such an application of the merger doctrine under traditional copyright precepts. As the court perceptively put it, a creative composition such as a narration of Humpty Dumpty's demise does not serve the same ends as a recipe for scrambled eggs.

- those dictated by external factors:

The classical copyright doctrine, *scènes à faire,* dictates that similarities due to hardware constraints or constraints imposed by having to interact with other programs would be deemed dictated by external factors and hence not protectable. An example of its application in the software field is found in the 1987 case of *Plains* Cotton Co-op v. Goodpasture Com*puter Services Inc.* in which the similarities between the accused and the defendant were concluded to be dictated by the externalities of the cotton market and the Cotton Exchange (807 F.2d 1256).

- those taken from the public domain:

Elements in the public domain are deemed unprotectable and, in and of themselves, cannot be made protectable by incorporation in a new program. However, the scope of this exclusion from protectable expression can be somewhat murky. For example, the *Altai* court said that public domain extended to any element that is "if not standard then commonplace in the computer software industry."

Importantly, elements which are unprotectable because of their public domain status may become protectable as part of a constellation of elements, where the totality is allegedly to be protected and to have been copied. In a compilation, however, the test for infringement of works as a whole is not the usual "substantial similarity" test; instead, "virtual identity" must be found. The unprotectable elements (or a mix of protectable and unprotect-

able elements) become a compilation, and may be protected as such. The courts sometimes speak of this standard as requiring a finding of "bodily appropriation of expression." According to an order issued April 14, 1993 by Judge Walker, this standard was applicable for determining whether user interfaces of Microsoft and Hewlett Packard infringe on those of Apple. Whether this standard has any applicability to software problems other than screen displays remains to be determined.

Comparison is the third and final step in determining substantial similarity of the accused program with the core of protectable expression. In its comparison, a court must assess not only whether defendant copied any protected expression, but also the importance of the copied portion relative to the plaintiff's overall program. In the *Altai* case, the core of protectable expression became known as the "Golden Nugget."

Altai is notable for its criticism of *Whelan*, particularly its much broader concept of protectable expression, and distanced itself from *Whelan*. *Whelan*'s breadth was partially dictated by the policy consideration of providing an incentive for programmers to engage in the hard work of writing programs. However, as the Altai court noted, the death knell of the "sweat of the brow" doctrine was sounded by the Supreme Court in *Feist* [2]. The undercutting of that policy as a basis for copyright protection by *Feist* warranted narrowing *Whelan's* scope and application to the *Altai* case.

As well, the trial court appointed its own expert to guide it through the intricacies of computer science who, in this case, happened to be Professor Randall Davis of MIT. As expert, Davis promoted a narrow scope of copyright protection for software, a direct contrast to the *Whelan* conclusions.

Professor Davis also clearly explained programs in a way that illuminated the error of *Whelan's* synonymous use of catchwords such as "structure," "sequence," and "organization," which it deemed protectable. Davis explained all programs have two dimensions—static and dynamic. Written code represents the static dimension, and machine behavior, as directed by the code, represents the dynamic dimension. The court recognized, albeit not in the clearest fashion, that this dynamic behavior constitutes process or method and, hence, such properties in this dimension are probably better suited to protection under the patent laws than under the Copyright Act.

Interestingly, this argument was already made by the defendant in an early phase of *Lotus Development Corporation v. Borland International Inc.* [3]. Citing Section 102(b) of the Copyright Act which provides that "[i]n no case does copyright protection ... extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work," the defendant sought to have the court declare that any element of a copyrighted program that was shown to be a process or method should automatically be deemed beyond the scope of copyright protection. Judge Keeton, denying summary judgment in March 1992, refused to make such a rule.

One must question how the *Whelan* court could possibly have cast such a broad net in view of this explicit limitation on the scope of copyright. The way Judge Keeton incorporated §102(b) into his analysis was to place all of the items excluded from protection together with ideas in the filtration step. Although it would seem under Judge Keeton's analysis that, to the extent something constitutes a method, it would be excluded from copyright protection, he expressly declined to go this far on the basis that Congressional intent did not warrant it and that a program's right to be patented did not preclude it from also being protectable under the copyright law. Such an extreme position was not, of course, advanced by *Borland*. Judge Keeton's reluctance to give Section 102(b) its just due squares with his expansive view of copyright in this area.

Perhaps the clearest expression as to the intersection of the copyright and patent laws in this respect is set forth in the September 1992 decision of the Court of Appeals for the Federal Circuit in *Atari v. Nintendo* [4]. The CAFC, whose jurisdiction is limited to cases involving patent issues, decided this copyright appeal because the case involved patent issues as well, implicating its jurisdiction, but it decided the copyright issues under Ninth Circuit law. Judge Rader reasoned that while the nonliteral aspects of software were protectable under the copyright law, Section 102(b) precluded extending this protection to ideas, procedures, processes, systems, methods of operation, and so forth. Section 102(b), he said, "is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law." Fitting Section 102(b) into the abstractions test, and recognizing that the patent laws "provide protection for the process or method performed by a computer in accordance with a program," he concluded, "Patent and copyright laws protect different aspects of a computer program. [The patent law] protects the process or method performed by a computer program; [the copyright law] protects the expression of that process or method. While [the patent law] protects any novel, nonobvious and useful process, [the copyright law] can protect a multitude of expressions that implement that process. If the patentable process is embodied inextricably in the line-by-line instructions of the computer program, however, then the process merges with the expression and precludes copyright protection."

Just where these decisions leave the state of the law is not entirely clear. Neither *Altai* nor *Lotus* engages in an analysis based squarely on the dynamic/static dichotomy advanced by Professor Davis, and the facts are not set forth in the opinions in sufficient detail so that one can infer that the courts meant to do so. Certainly, it is true that one can write very different looking programs, in terms of their static dimension, which will result in essentially identical machine function in the dynamic dimension. If such a case were squarely presented for decision, it would be difficult to predict whether a court would hold arguably nonessential components of the program's *behavior* subject to protection. Judge Keeton has clearly evidenced more of a tendency to hold them protectable than the courts of the Second or Ninth Circuits.

PREEMPTION

According to Section 301 of the Federal 1976 Copyright Act, any state or common-law right that is equivalent to those spelled out in Section 106 of the Act is preempted. What is or is not an equivalent right has been the subject of much litigation. If the facts established about copyright infringement are the same facts that establish the state or common-law right, the latter will be preempted. However, if there are additional elements beyond the mere reproduction of a copyrighted work, the state or common-law claim will usually stand. Thus, a claim involving breach of a confidential relationship or a fiduciary duty or misappropriation of trade secrets would usually not be preempted at the Federal level. On the other hand, "other elements," such as awareness or intent, do not alter the nature of the alleged tort, and therefore are subject to preemption.

One way of helping to ensure that a claim for misappropriation of trade secrets is not preempted is to allege and prove it in terms of noncopyrightable elements. In particular, processes, methods, ideas, and systems are not within the scope of copyright by virtue of Section 102(b). Thus, if you allege and prove that those components are the things which have been misappropriated, you should be able to avoid preemption.

In its original decision in *Computer Associates,* which was handed down in June 1992, the Second Circuit held Computer Associates' claim for misappropriation of claim secrets to have been preempted. In December 1992, on petition for rehearing of that part of the case, the court reversed itself, holding that the trade secrets claim had not been preempted. *Computer Associates* did not establish a legally sufficient difference between the copyright and trade secret claims. The focus on the code stolen by the former employee, rather than on the unique nature of the adapter module, seems to have been a good bit of the problem. Thus, if Computer Associates could establish that Altai's second version of its product, written in a "clean room," employed misappropriated trade secrets, Computer Associates may be entitled to damages far beyond those it was awarded, a defeat that would be particularly important given the increasingly narrow view of copyright protection that has evolved.

Another particularly interesting example of a preemption case, also involving Computer Associates, was decided by the Eighth Circuit Court of Appeals last year [5]. There, National Car Rental leased accounting software from Computer Associates, and the lease limited use of the software to National's own data. When National used the software to process the data of other companies, Computer Associates claimed that this processing was in violation of the lease and constituted a breach of contract and an infringement of the copyright in its software.

After a lengthy analysis, the court found that the limitation on the use of the software contained in the lease was an "extra element," beyond the elements necessary to establish copyright infringement, and therefore the breach of contract claim was not preempted by Federal law. However, the copyright law expressly prohibits copying a program. To process data that was not their own, National copied the software into memory. Had Computer Associates challenged National for copying, rather than "using", its software in the alleged breach of contract, the court should have found that the contract claim was preempted.

THE PERMISSIBLE SCOPE OF REVERSE ENGINEERING

One area of significant debate in copyright law has been the extent to which one can legally reverse engineer a competitor's program by decompilation to ascertain features necessary to understand how to engineer and design one's own competitive, noninfringing product and *not* to plagiarize the competitor's work. The argument for reverse engineering claims that, because competition is in the public's best interest, anyone should be free to decompile copyrighted code to understand its ideas, methods of operation, interaction with the operating system, other programs and the hardware, and any other aspects not protectable by copyright. The counter argument, usually presented by those favoring a broad scope of copyright protection, maintains that there should be no special privilege for copying software, and that the end result of such decompilation, albeit not an infringement, should be deemed an infringement because its creation stemmed from the intermediate infringing step.

In 1992, *Atari v. Nintendo,* decided in September by the Federal Circuit applying Ninth Circuit law, and *Sega v. Accolade,* decided in October by the Ninth Circuit [6] came down clearly on the side of establishing a limited right of decompilation. In *Atari v. Nintendo,* the court affirmed an injunction against selling arguably noninfringing works derived by decompilation. However, it would not have sustained the injunction except that Atari, the copier, had "unclean hands" because it obtained the copyrighted Nintendo code from the Copyright Office by falsely alleging the existence of a lawsuit when, in fact, there was none at the time.

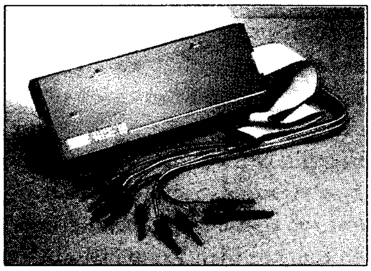
Both cases involved decompilation to get at the "lock and key" mechanism which enabled only the manufacturer's games or games licensed by the manufacturer to play on the manufacturer's hardware. Both cases established that such "lock and key" schemes were functional and copyable provided the copier did not appropriate code beyond what was necessary to design programs that could unlock the software and play it on the appropriate hardware. However, the scope of the decompilation right was limited to situations in which decompilation is the sole means of gaining access to the unprotected aspects of the program and to which the copier has a legitimate interest in doing so (i.e., one that will lead to a lawful competitive use and not to an infringement).

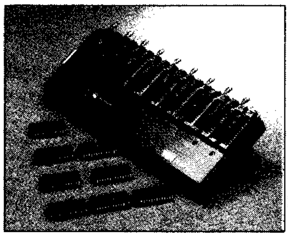
In *Sega v. Accolade* four arguments were advanced about why intermediate copying through decompilation should not be deemed an infringement. The first argument sought a rule that intermediate copying should not be proscribed under Section 106 unless the final commercial product was also an infringement. The second maintained that decompilation should be deemed lawful under Section 102(b) since that section exempts ideas and functional concepts from copyright protection. The third argued that decompilation was authorized by Section 117, which permits the lawful owner of a program to copy and use it in a derivative work in which doing so is essential to the lawful functioning of the program. Although these three arguments were all rejected, the court nonetheless determined that there was no infringement of the fair use provision embodied in Section 107 of the Act.

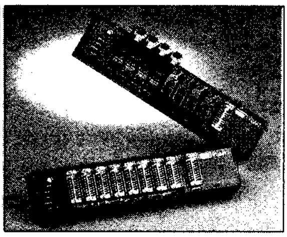
Broadly, Section 107 lists four factors to be considered in determining whether an act that otherwise would infringe is exempt as a fair use. The factors, which are not exclusive, include:

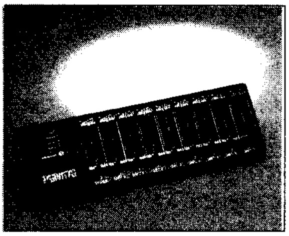
- the purpose and character of the use (a commercial versus nonprivate, educational use)
- the nature of the copyrighted work
- the amount and substantiality of the portion used in relation to the copyrighted work as a whole
- the effect of the use on the potential market or value of the copyrighted work.

Analyzing these factors in light of the public policy underlying the Copyright Act, the court found that what Atari did was a fair use. The court recognized that Atari's reverse









engineering promoted public policy interest through competition and the dissemination of other creative works and, citing *Feist*, they pointed out that the copied work contained unprotected ideas not protected from decompilation.

In addition, the court invoked the long-established principle that not all copyrighted works are entitled to the same degree of protection. The more functional a work is, the less protection it is awarded. Since the court determined that computer programs are "in essence utilitarian articles ... that accomplish tasks," less protection was warranted.

Finally, although the ideas in most copyrighted works (including many computer programs in which the user interface and the program's operation are apparent without delving into the code) are visible to the naked eye, the court found that decompilation was the only method available to get at the ideas and other unprotected elements of the program.

The scope of this reverse engineering exception under the fair use doctrine remains to be determined.

THE USE OF TRADEMARKS TO PROTECT SOFTWARE

Some authorities have long counseled that, by embedding a trademark in copyrighted code, one can in effect force copiers to "use" the trademark so that they explicitly become infringers. (I am not referring to the situation in which a trademark is buried in code by the copyright owner, so that finding the trademark in the infringing work establishes copying.) In the *Sega* case, Sega tied the particular code sequence that unlocked the hardware and allowed the program to play, with the code sequence which displayed Sega's trademark on the screen. This measure was taken by Sega specifically to assure that plagiarizers in countries where it is difficult to establish copyright infringement would unwittingly become trademark infringers as well.

The posture of the case was wholly different from the usual trademark situation in which one party alleges infringement and the other denies it. Here, both parties acknowledged trademark infringement and each side claimed damage from it while, at the same time, each accused the other of responsibility for the situation. The court held that, by making trademark involvement a condition of legitimate copying, Sega was responsible and hence Accolade was not an infringer.

PATENTS

The Federal Circuit's March 1992 decision in *Arrhythmia v. Corazonix* [7] introduced the most significant and rational opinion about the applicability of patent laws to software that we have seen in a long time. Although decisions of lower courts cannot do away with those of superior courts, the Federal Circuit in *Arrhythmia* seemed to do just that. It swept away the debris of the Supreme Court's 1972 ruling in *Gottschalk v. Benson* that has cast a long and painful shadow over most and possibly all decisions in computer patentability for twenty years.

Davis explain[s] all programs have two dimensions—static and dynamic. Written code represents the static dimension, and machine behavior, as directed by the code, represents the dynamic dimension.

In *Benson*, the Supreme Court held that a method of converting binary-coded-decimal to binary numbers in a computer system was a "mathematical algorithm" and, consequently, nonstatutory subject matter (nonprotectable). In making its decision, *Benson* had invoked Section 101 of the Patent Act which defines what constitutes patentable subject matter. The court concluded that a mathematical algorithm involved or was akin to a "law of nature," "natural phenomenon," or "formula," all of which were not patentable subject matter. Its definition of a mathematical algorithm as "a procedure for solving a given type of mathematical problem" offered little enlightenment since many processes in science and engineering are subject to mathematical description. The breadth of *Benson* was so startling, even at the time, that it caused Professor Chisum, author of a leading treatise on patent law, to write, "A recent Supreme Court decision eliminated patent protection for computer software." Similarly, Judge Rich, an author of the 1952 Patent Act, remarked in 1973, "The Supreme Court in *Benson* appears to have held that claims drafted in such terms are not patentable—for what reason remains a mystery."

In the 1979 *Parker v. Flook*, the Supreme Court plunged into its next software problem. In *Flook*, the process that the court held nonstatutory employed a mathematical formula calculating alarm limits for a catalytic conversion process. In the process, limits were reset so that the alarm would go off if the process strayed beyond calculated boundaries. The Supreme Court seized this opportunity to establish a unique patentability analysis for software. In particular, the claim was evaluated in its entirety, but the algorithm was to be treated as part of the prior art.

Finally, in *Diamond v. Diehr*, decided in 1981, the Supreme Court was confronted with a software development couched in terms that seemed less threatening. The claimed process was one for molding rubber, and the algorithm at issue was one periodically used to compute the temperature throughout the mold and to turn the heating elements off at the appropriate point. The court did not overturn its prior decisions, including *Benson*, but instead adopted the view that the presence of an algorithm in an otherwise statutory claim did not make the subject matter unpatentable.

During the period from 1978 to 1982, the Court of Customs and Patent

Appeals fashioned what became known as the Freeman-Walter-Abele test for statutory subject matter. If a mathematical algorithm were recited in the claim, either directly or indirectly, it became necessary to determine whether the claim preempted all uses of the algorithm. Only if it did not was the claim statutory under Section 101, except that if the algorithm were circumscribed *merely* by limitation to a particular field of use, or in terms of nonessential, post–solution activity (e.g. using an output signal to set alarm limits), that would not rescue it from being nonstatutory.

In the recent *Arrhythmia v. Corazonix,* it was held that a method of analyzing electrical signals from electrodes monitoring a patient's heart and processing them to detect the presence of a certain low-amplitude, high-frequency signal indicative of a dangerous type of heart arrhythmia called *ventricular tachycardia,* was statutory subject matter under Section 101.

In giving the opinion of the court, Judge Newman managed to fit the result within the tortured confines of the rule laid down in *Gottschalk v. Benson.* In contrast, Judge Rader confronted the situation more forthrightly concluding that, in the 1981 case of *Diamond v. Diehr,* the Supreme Court itself abandoned the *Benson* rule. He cautioned against reading anything into the plain words of the statute (which is what caused the Court's misunderstanding in *Benson*) and admonished lower courts to simply follow the statute. Analyzing the problem under the language of the patent statute itself, Judge Rader found the claim to comply with the provisions of Section 101.

Interestingly, in finding the subject matter statutory, the majority had no trouble dealing with the electrical signals involved. They perceived the electrical signals' representation in numerical form, not as "mathematical laws" or abstract ideas, but as representations of tangible physical structure or process steps. Claim 1 of *Arrhythmia* recited a method for analyzing electrocardiograph signals to determine the presence or absence of a predetermined level of high-frequency energy in the late QRS signal which involved:

- converting a series of QRS signals to time segments, each segment having a digital value equivalent to the analog value of the signals at that time
- applying a portion of the time segments in reverse-time order to high-pass-filter means
- determining an arithmetic value of the amplitude of the output of the filter
- comparing the value with the predetermined level

Although Corazonix argued that the process defined no more than a mathematical algorithm that calculates a number, the court found each claim limitation to correspond to physical structure or function and hence to be statutory.

While *Arrhythmia* is a strong positive clarification of the protect–ability of algorithms, we can expect minor surgeries along the way such as in the recent *In re Schrader,* decided by the Federal Circuit in April 1994. The invention in *Schrader* was an algorithm for evaluating bids at an auction by making them comparable even if submitted in a form which was not comparable. The court found that the claim didn't meet the Freeman-Walter-Abele test because there was no physical transformation of data by the algorithm. It is not altogether clear whether the court objected to the form of the claim under consideration or, more fundamentally, to protecting this kind of algorithm.

CONCLUSION

Historically, patents have protected technology while copyright has protected literary works. When protection for computer programs was demanded in the 1950s and 1960s, the patent system failed to respond. The 1976 Copyright Act became the primary legal vehicle for protecting programs, clearly a technology, rather than patent law. That turnabout has now largely been undone as the patent law has appropriately shouldered its burden of protecting software technology, and the role of copyright has commensurately narrowed.

We are left with the lesson that the fundamental time scale of our legal systems is much slower than the pace of technological change. ▣

Barry D. Rein is a specialist in patent and copyright for high-tech companies at the intellectual property law firm of Pennie & Edmonds where he is a senior partner. Barry holds an electrical engineering degree from MIT and a law degree from Georgetown University Law School. He may be reached at (212) 790-6546.

REFERENCES

[1] *Computer Associates International, Inc. v. Altai, Inc.,* 982 F.2d 693.

[2] *Feist Publications, Inc. v. Rural Telephone Service Co.,* U.S., 111 S. Ct. 1282 (1991).

[3] 788 Federal Supplement 78 (D. Mass. 1992).

[4] *Atari Games Corp. et al. v. Nintendo of America Inc. et al.,* 1992 U.S. Appeals Lexis 21817.

[5] *National Car Rental System, Inc. v. Computer Associates International, Inc.* 1993 WL 98043 (Eighth Circuit (Minn.)).

[6] *Sega Enterprises Ltd. v. Accolade, Inc.,* 977 F.2d 1510 (Ninth Circuit 1992).

[7] *Arrhythmia Research Technology, Inc. v. Corazonix Corp.,* 958 F.2d 1053 (Federal Circuit 1992).

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

DEPARTMENTS

52 Firmware Furnace

60 From the Bench

68 Silicon Update

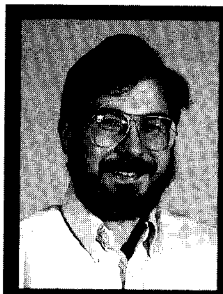
74 Embedded Techniques

82 ConnecTime

FIRMWARE FURNACE

Ed Nisley

Journey to the Protected Land: Segments All the Way Down



The protected land where 80x86-type processors try to be like the big boys is fraught with peril, so you must be careful. Travel with Ed as he tries to make the journey smoother for those who follow.

One view of cosmology has it that the world is a vast disc supported on the back of a mighty turtle. Upon learning this, one thoughtful adept asked his tutor, "What does the turtle stand upon?" The master replied, "The turtle stands upon the back of another turtle, even more mighty."

The adept paused, thought this over, and asked, "Well, then, what does that turtle stand upon? The master quickly replied, "Yet another turtle."

"And that turtle?"

"Listen, lad, it's turtles all the way down."

Last month we pushed the CPU into 16-bit protected mode through the BIOS knothole designed for that task. This month we'll JMP into 32-bit mode on our own and explore some of the new territory. In particular, we need to know about various types of segments.

Contrary to popular opinion, the 80386's segmented architecture doesn't mystically vanish when it enters 32-bit mode. You need new code segments, data segments, stack segments, table segments, and segment segments on both sides of the border.

Yes, folks, it's segments all the way down....

RIGHT...WHAT'S A SEGMENT?

To judge from popular sentiment, Intel invented segments specifically for the bedevilment of PC programmers. The truth, as always, is more complex and we must discuss what a segment is and what it can do before the 80386 architecture makes any sense.

Long before PCs ruled the land, the notion of a segment was simple and had nothing at all to do with hardware. Segments provided a convenient way to partition the contents of a program: instructions would be in one segment, initialized data in another, constants in another, the stack in yet another. Each source file in a program could place information in any of several different segments; the linker would then collate the sections so all the code segments were contiguous, all the data lined up together, and so forth.

Intel's hardware segments solved an entirely different problem: how to access more than 64 KB of memory with the newfangled 8086 16-bit processor while remaining more-or-less compatible with the older 8080 8-bit CPU. In this context, a segment is a chunk of memory that has no special attributes. If they'd called it something else, the world would surely be better off today.

The 8086 included four 16-bit segment registers: CS for code, DS for data, SS for the stack, and ES for anything else (extra). Each segment register pointed to the start of a 64 KB block of memory that was further addressed by a 16-bit offset quantity. Using a value within the 64 KB region, starting at the address in any segment register was easy; anything else was hard.

Because the CS, DS, and SS registers were used implicitly by many instructions, their associated segments were known as the code, data, and stack segments. The linker now collated the various source program segments and associated

Listing 1—*The layout of segment descriptors involves several nested fields. These C structures mapping the fields provide bit-by-bit control over the descriptor contents.*

a) Segment Attribute Byte. This structure includes the high-order four bits of the Segment Length field so that it's aligned on a byte boundary.

```
struct SEG_ATTR {
   WORD SegLimitHigh : 4;    // Segment limit high nybble
   WORD Available : 1;       // 4 “available” bit    for OS use
   WORD Zerobit : 1;         // 5 must be zero
   WORD DefOpSize : 1;       // 6 operation size 0=16-bit 1=32-bit
   WORD Granularity : 1;     // 7 segment granularity 0=byte 1=4K
};
```

b) Segment Access Rights Byte. Memory segments, those with the S bit = 1, use the SEG_ACCESS structure. System segments, with S=0, require the SYS_ACCESS fields.

```
struct SEG_ACCESS {       // for Code, Data, and Stack segments
   WORD Accessed : 1;     // 0 0=unused 1=descriptor loaded
   WORD ReadWrite : 1;    // 1 Code: 0=execute only, 1=readable
                          //   Data: 0=read only, 1=writable
   WORD Conforming : 1;   // 2 Code: 0=nonconforming 1=conforming
                          //   Stack: 0=expand up 1=exp down
   WORD Executable : 1;   // 3 0=data/stack 1=code
   WORD SegType : 1;      // 4 0=system 1=code/data/stack
   WORD DPL : 2;          // 5:6 descriptor privilege level
   WORD Present : 1;      // 7 0=not in memory, 1=present
};

typedef struct SEG_ACCESS seg_access;

struct SYS_ACCESS {      // for System segments
   WORD Type : 4;        // 0:3 type of segment
   WORD SegType : 1;     // 4 must be zero for system segment
   WORD DPL : 2;         // 5:6 descriptor privilege level
   WORD Present : 1;     // 7 0=not in memory, 1=present
};

typedef struct SYS_ACCESS sys_access;
```

c) Segment Descriptor. The DESC_NORM and DESC_SYS structures handle most of the possible segments. Call, trap, and interrupt gates use slightly different field definitions that I'll cover in next month's column.

```
struct DESC_NORM {
   WORD SegLimit;          // segment limit, bits 0:15
   WORD SegBaseLow;        // segment base, bits 0:15
   BYTE SegBaseMid;        //  … bits 16:23
   seg_access Access;      // access bits
   seg_attr Attributes;    // attribute bits & limit 16:19
   BYTE SegBaseHigh;       // segment base 24:31
};

typedef struct          DESC_NORM desc_norm;

struct DESC_SYS {
   WORD SegLimit;          // segment limit, bits 0:15
   WORD SegBaseLow;        // segment base, bits 0:15
   BYTE SegBaseMid;        //  … bits 16:23
   sys_access Access;      // access bits
   seg_attr Attributes;    // attribute bits & limit 16:19
   BYTE SegBaseHigh;       // segment base 24:31
};

typedef struct DESC_SYS desc_sys;
```

them with the CPU's hardware segments.

The catch was that the hardware had no way to verify that the segment registers were actually loaded with the correct values. Hardware segments were simply addresses, so any association with a program segment was just a promise made by the linker. Worse, there was no restriction on the offset values: even if the source program's code segment was only 10 KB long, you could branch to an address 50 KB from the start of the hardware segment in CS and execute whatever you found there, much to the amazement of all concerned.

That is the essence of real-mode programming: you promise the CPU you'll set up the segments correctly and never make a mistake. The CPU promises that it will hide in the bushes when it finds the mistake you didn't expect to make. You get to reboot the system and track down the problem without any help from the CPU.

What you really want is a way to tell the CPU what each code, data, and stack segment contains, then allow it to enforce those properties and report on discrepancies. That's the basis of protected mode: promises with teeth.

It's sharks all the way down....

FLIPPING THE BIT

As you learned last month, every chunk of memory accessible to the CPU is defined by a descriptor in the Global or Local Descriptor Tables. Figures 1 and 2 in that column described all the descriptor fields; glance back at Issue 48 for a refresher. Listing 1 this month shows the C structures I used to build the descriptors.

The only difference between a 32-bit code segment and a 16-bit code segment is the D (Default Size) bit in their segment descriptors. When D=1, the CPU assumes all operands and addresses are 32 bits long, allowing full use of the hardware. D=0 forces 16-bit operations for compatibility with real mode and 80286 protected mode. The operand size prefix (0x66) and the address size prefix (0x67) allow you to modify a single instruction in either type of segment to act as though it were the other size.

Listing 2—*A program running in 32-bit mode requires only a 32-bit code segment, but a stack less than 64K bytes long can be a "16-bit" stack using SP instead of ESP and data segments have no inherent length. This fragment of code from the BuildGDT() function creates three new GDT descriptors with the D bits set just for neatness.*

```
++pDesc;                                  // step to 32-bit CS entry
pDesc->Access.SegType = 1;                // code/data/stack
pDesc->Access.Present = 1;                // always present
pDesc->Access.Executable = 1;             // code is executable
pDesc->Access.ReadWrite = 1;              //  ... and readable
pDesc->Attributes.DefOpSize = 1;          // 32-bit ops
SetDescAddr(pDesc,
        MakeLinear(PMCodeBase),
        (DWORD)&PMCodeLength);


++pDesc;                                  // step to 32-bit DS entry
pDesc->Access.SegType = 1;                // code/data/stack
pDesc->Access.Present = 1;                // always present
pDesc->Access.ReadWrite = 1;              //  ... and readable
pDesc->Attributes.DefOpSize = 1;          // ignored for data
SetDescAddr(pDesc,
        MakeLinear(&PMDataBase),
        (DWORD)&PMDataLength);


++pDesc;                                  // step to 32-bit SS entry
pDesc->Access.SegType = 1;                // code/data/stack
pDesc->Access.Present = 1;                // always present
pDesc->Access.ReadWrite = 1;              //  ... and readable
pDesc->Attributes.DefOpSize = 1;          // 32-bit stack!
SetDescAddr(pDesc,
        MakeLinear(&PMStackBase),
        (DWORD)&PMStackLength);
```

Note that you cannot have a 32-bit code segment in real mode because, by definition, segment descriptors aren't used in real mode. It Would Be Nice, but....

The D bit is also used in stack segments, which are just data segments referred to by a selector in the SS register. The CPU uses all 32 bits of ESP when D=1. When D=0, it uses only the low-order 16 bits in SP, so if you need more than 64 KB of stack space, the stack descriptor must have D=1. A code segment with D=1 can use a stack segment with D=0 and vice versa, but there are some distressing gotchas.

Data segments that aren't accessed by SS don't have an inherent size because the D bit in the CS descriptor determines the size of the instruction's operands. It is probably a good idea to set the data descriptor's D bit just so you remember what you were thinking about when you created the segment, but it doesn't really matter.

Contrary to what you might think, the G (Granularity) bit in the code or data descriptor does not affect the 32-bithood of the segment. Setting G=1 simply multiplies the Segment Limit field by 4096 to allow segments up to 4 GB in size. For data segments up to 64 KB, every byte can still be reached with a 16-bit offset in D=0 code segments. Obviously there is little benefit to G=1 data segments in 16-bit mode, but it does work nonetheless.

To run 32-bit programs, we need at least a 32-bit code segment. I also created 32-bit stack and data segments, even though this isn't strictly necessary. Listing 2 shows the lines in BuildGDT() that initialize the three new GDT entries.

The assembler normally produces 16-bit code because that's all you need in real mode. The next step is persuading it to emit 32-bit code in our new code segment.

CODE MODES

The compatibility barnacles forced a difficult decision on the 80386 CPU designers: creating a CPU that can process either 16- or 32-bit instructions and operands. The solution was simple: the same instruction opcodes—the identical binary values—will execute in either mode depending on the code segment descriptor's D bit.

But because the default operand size changes from 16 to 32 bits, you cannot simply transplant 16-bit code into a 32-bit segment and run it. The CPU will fetch the wrong number of immediate data bytes (for just one example) and quickly fail with, yes, a protection exception. In this case, the CPU may stumble along for a few instructions while doing entirely the wrong thing, but eventually the protection hardware will catch an invalid operation.

The assembler directives in Listing 3 create the 32-bit code, data, and stack segments corresponding to the descriptors in Listing 2. The USE32 keyword tells the assembler that the default operand and address sizes are now 32 bits. Simple labels and EQU statements provide the starting and ending addresses for the GDT segment descriptors; in effect, we are performing relocation "by hand" on these few values.

The MODEL directive establishes the assembler's memory model and the function-calling protocol. In our case, the SMALL model is appropriate: code and data are in separate segments. The FARSTACK modifier tells the assembler that the stack is not part of the data segment. The C language specifier allows us to use the C calling convention for assembly language functions; although I don't plan to use a C compiler right away, this keeps the option open.

There are two sets of promises in Listings 2 and 3. Listing 2 promises the CPU that the new segments will have certain properties. Listing 3 promises the assembler that the CPU will treat the segments in a certain way. The CPU will eventually verify that both promises are kept.

If we were writing an application program for a PM operating system, all this would be handled automatically by the linker and loader. The operating system would analyze the program file, assign the proper descriptors, then load the program into memory and update the descriptor addresses. We get to do this by hand because none of that infrastructure exists down at our level.

Once the 32-bit segments are defined, using them within the program is simple. Listing 4, which I extracted from the assembler output file, shows the code generated by the source instructions for the first few lines of the initial 32-bit entry routine. Several characteristics of 32-bit code are worth noting.

You must now use the 32-bit register names: EAX, EDX, and so forth. In 16-bit mode, those registers require an operand length prefix byte, which means you don't use 32-bit registers unless they are essential. The reverse is true in 32-bit mode: 16-bit operands generate the prefix byte shown in line 1987. Single-byte registers, such as AL in line 1978, continue to work as you'd expect.

Immediate data values loaded into 32-bit registers are four bytes long. This tends to plump up your code with fluffy binary zeros since the high-order bytes of most constants are zero...but that's the price you pay for a wide data path. You can use the MOVSX and MOVZX instructions to sign- or zero-extend a byte value to 32 bits on the fly.

The values loaded into segment registers are now numeric selectors rather than addresses. The constants used in Listing 4 correspond to the GDT I described earlier. Loading an invalid selector, or a selector to an invalid GDT entry, causes a protection exception, so it's a good idea to use manifest constants rather than raw numeric values.

Finally, code and data addresses are 32 bits long. The linker's /3 command-line option enables support

Listing 3—*The segment descriptors in Listing 2 refer to these assembly language definitions. You must be sure the assembler and CPU use the same code segment instructions size because executing 16-bit code in 32-bit mode, or vice versa, will cause protection exceptions when the CPU stumbles over incorrect data lengths. The data segment contains a flag value that's displayed during the real mode setup sequence.*

```
      MODEL         USE32 SMALL,C     ; C language calling sequence

;- code segment

      SEGMENT       _prottext PARA PUBLIC USE32 'CODE'

      LABEL         PMCodeBase PROC
      PUBLIC        PMCodeBase

      ENDS          _prottext

;- data segment

      SEGMENT       _protdata PARA PUBLIC USE32 'DATA'

      LABEL         PMDataBase DWORD
      PUBLIC        PMDataBase
      DD            0babefaceh        ; flag the first location

      ENDS          _protdata

;- stack segment

      SEGMENT       _protstack PARA PUBLIC USE32 'STACK'

      LABEL         PMStackBase DWORD
      PUBLIC        PMStackBase
      DD            1023 DUP (?)
      LABEL         PMStackTop
      DD            ?                 ; this is the stack top!

PMStackLength       = $-PMStackBase
      PUBLIC        PMStackLength

      ENDS          _protstack
```

for the larger values. In our case, we won't have address offsets above 64 KB for quite a while, but once you decide to use 32-bit code, there's no turning back!

'JMPING' THE BARRIER

Paradigm designed their Locate utility to handle ordinary real-mode 16-bit code. It will handle our simple 32-bit code because we are taking care of the segment selectors manually. The segment definitions in Listing 3 tell the linker to put the three 32-bit segments in the same classes as the corresponding 16-bit segments. Locate handles the combined classes as it normally does, so the code and data wind up at the expected spots.

The code in Listing 2 that generates the GDT descriptors uses labels and values computed by the assembler to find the starting addresses and lengths. While this may seem like a kludge, it neatly sidesteps the need for a specialized tool to handle a trivial task. Things will get a little more complex later on, but this is a good start.

Transferring control from the 16-bit code segment created by the BIOS function into our 32-bit code segment requires a `FAR JMP` to reload both the segment and offset registers. Real-mode code segments are just address bits, so the arbitrary PM selectors simply won't work. The linker complains that it can't resolve the address of the `JMP` target, which is quite correct because there is no way to tell it that the segment address isn't what it expects.

Listing 5 shows the macro I wrote to synthesize a `FAR JMP` instruction.

Listing 4—*When* USE32 is *in* effect, the *assembler generates 32-bit operands and assumes 32-bit addressing. This section of fhe assembler output listing shows the code generated af the initial 32-bit entry point Consfanfs loaded info EAX and EDX (the 32-bit counterparts of* **AX** *and DX) require four bytes each, while values going info* AL *still use a sing/e byte. The opcodes are the same as those in 16-bit mode because the segmenf's D bit specifies the operand size. When an instruction requires a 16-bit value, such as the* OUT *DX,AX in line 1987, an operand size prefix must precede the opcode.*

```
1970  00000000                 SEGMENT _prottext
1971
1972                           PUBLIC  PMEntry32
1973  00000000                 PROC     PMEntry32
1974
1975                           ASSUME  CS:_prottext
1976
1977  00000000  BA  00000378  MOV  EDX,SYNC_ADDR    ; show entry here
1978  00000005  EC             IN    AL,DX
1979  00000006  OC  20         OR    AL,20h
1980  00000008  EE             OUT  DX,AL
1981
1982  00000009  B8  00000048  MOV  EAX,GDT_DATA32;  set  up  data  seg
1983  0000000E  8E  D8         MOV   DS,AX
1984
1985  00000010  BA  0000031E  MOV  EDX,LED_ADDR
1986  00000015  B8 FFFF9886  MOV  EAX,NOT  6779h    ; show P3 for '386
1987  0000001A  66| EF        OUT  DX,AX            ; 32-bit  mode
1988
1989  0000001C B8  00000050  MOV  EAX,GDT_STACK32;  set  up  stack  seg
1990  00000021  8E  DO        MOV   SS,AX
1991  00000023  BC  00000FFCr MOV  ESP,OFFSET PMStackTop
```

Because it uses a data constant to generate the selector value, the assembler doesn't mark it as relocatable and the linker doesn't try to relocate it. The macro also inserts the prefix bytes that specify 32-bit operands and addresses in a 16-bit segment. Sometimes this high-level assembler stuff just gets in the way.. .

After all that buildup, the single instruction that enters 32-bit mode is anticlimactic:

```
PMJmpFar  GDT_CODE32,PMEntry32
```

The selector value matches up with the 32-bit code segment descriptor in the GDT and the offset marks the first instruction in Listing 4. The CPU loads the selector, verifies the segment's attributes, and fetches the first instruction shown in Listing 4. That's all there is to it.

Once in 32-bit mode, the code twiddles a few LEDs. sets up the SS and DS registers, and enters a loop that

Listing 5—*Because the segment values are now arbitrary numbers the real mode assembler and linker cannot relocate them. This macro synthesizes a FAR JMP with a constant selector value using data definitions rather fhan the usual instruction mnemonics to prevent the assembler from "helping out" with relocation values. The macro a/so inserts the prefix bytes that specify 32-bit operands and addresses in 16-bit segmenfs.*

```
MACRO      PMJmpFar Sel,Offs
IF         0 @32Bit
DB         066h                      ; force 32-bit operand size
DB         067h                      ; force 32-bit address size
ENDIF
DB         0EAh                      ; JMP LARGE FAR (6-byte imm)
DD         OFFSET Offs               ; 4-byte offset
DW         Sel                       ; Z-byte selector
ENDM
```

displays a counter on the FDB's LEDs. A subroutine converts the count to LED segments (talk about function overloading: that's the third "segment" we've met so far) to verify that the 32-bit stack is usable for calls, returns, and ordinary pushes and pops.

Part of the loop transfers the FDB's DIP switches into the ES register. All is well if the switches contain a valid selector, otherwise the CPU will cause a protection exception. This allows you to exercise the IDT and demonstrate that your system really is in protected mode.

```
Linked pointers and values from the PM segments...
 PM Code  123B:0000 = 00012380, length 000000B4 (BA 78 03 00 00 EC OC 20 )
 PM Data 2000:04D0 = 000204D0, length 00000018 (BABEFACE 12345678)
 PM Stack 2252:0000 = 00022520, length 00001000 (00000000)
Allocating IDT at 2000:051C
 Re-vector table 1000:062B = 0001062B
 First IDT entry 062B 0030 8600 0000
Allocating GDT at 2000:0D20
 GDT NULL         0000 0000 0000 0000
 GDT alias        0057 OD20 9202 0000
  IDT alias       07FF 051C 9202 0000
 DS               FFFF 0000 9202 0000
 ES               FFFF 0000 9202 0000
 SS               FFFF 0000 9202 0000
 CS               FFFF 0000 9A01 0000
 BIOS CS          0000 0000 0000 0000
 32-bit code      00B3 23B0 9A01 0040
   data           0017 04D0 9202 0040
   stack          OFFF 2520 9202 0040
```

Figure l--The real *mode startup code dumps the first few byfes of each 32-bit protected* mode segmenf *to verify that the linker and Paradigm's Locate handled the* relocation *correctly. On/y the last three GDT entries specify 32-bit mode because, as in last month's code, the BIOS "Switch to Protected Mode" function requires the other 16-bit segments for compatibility with the Original IBM A T's 80286 CPU.*

The IDT is unchanged from last month, which means the CPU switches from 32-bit mode to 16-bit mode when it enters the error handler. Each interrupt gate descriptor in the IDT specifies the original 16-bit PM code segment descriptor, which remains ready for use in the CDT.

Although we know nothing can go wrong (right?), the real-mode code displays status and tracing information as shown in Figure 1. If your system doesn't make it to the Protected Land, check the LEDs and traces to see where it veered from the trail.

CODE BASE ONE

Although I'm sure you'd like to see the whole FFTS protected-mode task switcher presented in the next column, that's not the way it's going to be. After all, what would I do in October?

What you ***will see*** is a series of columns exploring the fundamental building blocks beneath FFTS. For example, we will explore interrupt gates by writing an interrupt handler, activating hardware interrupts, and measuring the response time. Working step by step gives me enough room to explain what is going on without having to cover everything at once.

Along the way I'll accumulate a variety of utility routines that handle serial I/O, display things on LCD panels, twiddle the FDB hardware, and so forth. Most of this code appeared in C as I built the Firmware Development Board; this time around I'll cast it in 32-bit assembler to show what it looks like in PM. You won't see much of this code unless PM or 32-bit data makes a big difference in how it's handled. For example, accessing the real physical memory of the graphic LCD panel is a little trickier now.

In two or three months, the real-mode code will atrophy to a loader that doesn't display quite so much diagnostic information. The PM code, by then a disk file of its own, will split into several distinct modules. Until then, the code will remain a simple monolithic chunk to reduce the number of files and keep our attention focused.

What you should do is participate: download the code, experiment with it, and report back on the BBS. First of all, if it doesn't play, I want to know! More important, you should modify the code to make sure you understand how the machinery works. Try different interrupts, tweak the timings, add bells and whistles. After all, it's small enough that you can't go too far wrong and safe enough that you won't get hurt!

RELEASE NOTES

The BBS code this month puts your CPU into 32-bit protected mode, setting a variety of LEDs along with a way to track any problems. Once in 32-bit mode, it blinks an LED on the printer port and shows an incrementing count on the FDB LED display. The real-mode code sets up the GDT entries and sends a variety of information on the memory addresses going into each descriptor to the serial port before entering PM.

Next month, I'll examine interrupt and error handlers in the IDT and make a few timing measurements. ❑

***Ed Nisley, as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of the Computer Applications Journal's engineering staff. You may reach him at** ed.nisley@circellar.com **or** 74065.1363@compuserve.com.*

SOFTWARE

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

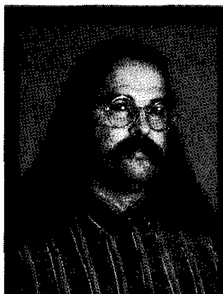
I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

Ta(I) king Control

FROM THE BENCH

Jeff Bachiochi



The dream of a computer understanding the spoken word lives on, and Jeff has found a new chip that makes it easier to fulfill. Find out if his computer is finally doing as he says and not as he does.

The death of radio has been greatly exaggerated. If you are not a connoisseur of (hard or soft) rock, jazz, C&W, rap, pop, hip-hop, swing, or classical music, then one of the public or college stations surely carries programming to please. Talk shows enjoy peak ratings these days. With a twist of the dial you can be enlightened and entertained by programming like "Car Talk" with hosts the Tappet Brothers.

When driving any distance, I like to listen to prerecorded audio dramas I have on cassette. Something about letting your mind paint its own pictures helps to solidify the space between fantasy and reality. Audio is a powerful medium unto itself.

Even television has its audio memories. Some personalities have become synonymous with a particular program from just one single phrase. "Herrrzzz Johnny" is the best example that comes to mind. We can't envision the "Tonight Show" without Ed McMahon. Although in video a picture is worth a thousand words, in audio a couple of well-chosen words can paint an entire picture. Or those same words can take control of the picture.

TAKING CONTROL

When I think of voice recognition today, PCs with ungodly amounts of memory come to mind. Memory for mammoth prerecorded vocabularies needed as baseline comparisons to real-time audio. Memory for the high-speed processing necessary to analyze the live input and attempt to match it with one already in existence.

Today, attempts to create the all-powerful recognition algorithm are cloaked in secrecy, yet great advances have been made in voice recognition. Phrase recognition has finally reach the usable stage, while continuous recognition, in all honesty, still has a ways to go. So, for the near future, voice recognition will remain a slave of memory and speed.

Some silicon has been developed which combines a number of masked preselected phrases with a pattern-matching algorithm. Refer to Michael Swartzendruber's "Control Your

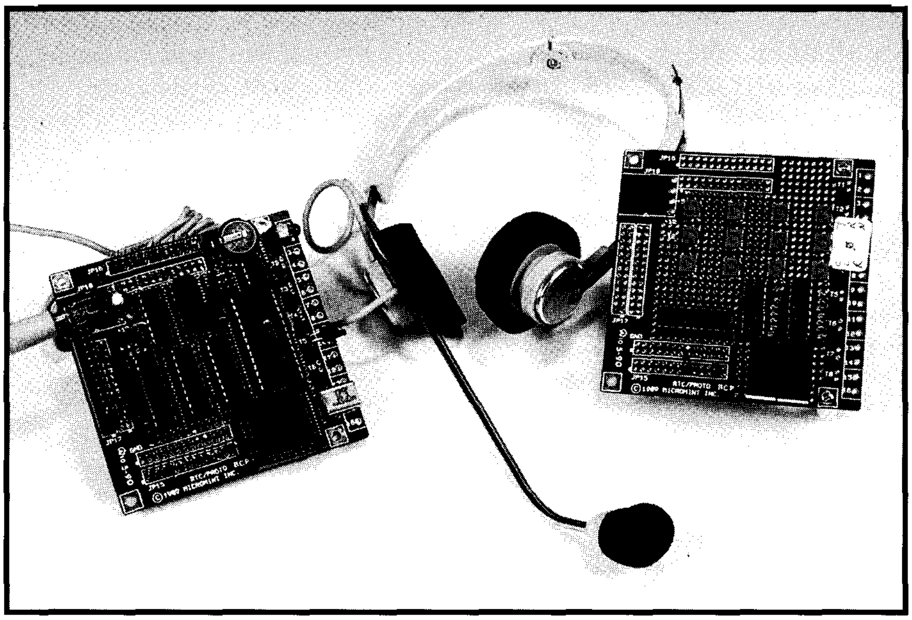


Photo I-The *HM2007* does on a single chip *what* required a *boardful* of electronics just a few *years ago: speaker-dependent, discrete-utterance voice recognition.*

Status Register Value (K-Bus)	Meaning
2	Ready for a command from the CPU
1	Ready for audio input
3	Ready to RD/WR lower nybble
0	Ready to RD/WR upper nybble

Table l-Besides *being busy, there are only four states the HM2007* can *be in.*

Telescope by Voice" in issue 32 (March 1993). Although limited to eight unchangeable phrases, the hardware is simple. Why can't someone take this a step further and allow the vocabulary to be user selected. Well, they can, and they have.

HMC (Hualon Microelectronics Corp.) distributes the HM2007 voice recognition chip through the Summa Group. The HM2007 will store up to 40 0.9-second durations or up to 20 1.9-second durations of audio. Successive audio is sampled and a "best guess" is provided for the closest match.

Information other than that contained in the device's data sheet is difficult to come by. I found it a bit aggravating to have to go though a third party to get answers to the more technical questions I had about the device and its usage. Not having an applications engineer available in the United States will definitely have an affect on how this chip is received by other developers.

There must have been some heavy disagreements during the development of this device. The final design seems to be a weak compromise between a manual and CPU-controlled device. This makes it more difficult to use effectively in either mode.

The handiest package available for the HM2007 is the 52-pin PLCC, although it is also available in a 48-pin PDIP and a 48-pin die. It contains an analog AGC front end, A/D converter, and masked CPU running at a handy 3.58 MHz. The minimum audio signal input necessary is about 20 mV. I've found a bit of preamplification may be required (depends on microphone used) to avoid having to shout into the electret mic.

MANUAL MODE

In manual mode, seven I/O lines create a twelve-key scanned matrix: 0–9, clear, and train. Twenty additional I/O lines create the twelve address and eight data connections supporting the SRAM and a display register. External SRAM is used for phrase storage. An 8-bit display register holds two BCD digits. These digits provide the user

Command Value (K-Bus)	Meaning (parameter)
0	Clear a trained word (input word#)
1	Train a word (input word#)
2	Recognize a word (audio in)
3	Give recognition result (get word# and Score)
4	Upload a word (input word#, get length and stored patterns)
5	Download a word (input word#, length, and saved patterns)
6	Reset (clears all words)

Table 2-The *HM2007 supports seven* CPU-issued commands.

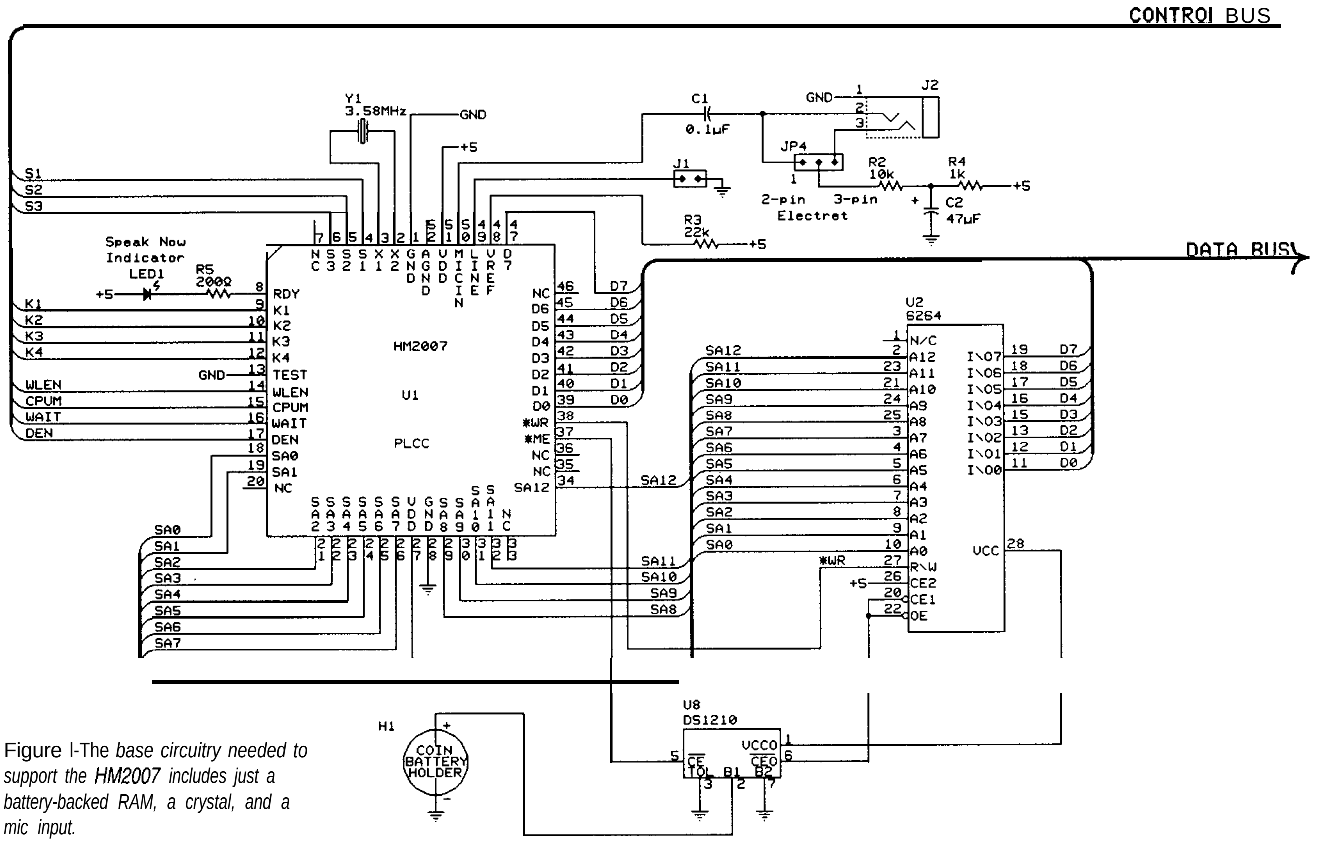


Figure l-The *base circuitry needed to support the HM2007 includes just a battery-backed RAM, a crystal, and a mic input.*

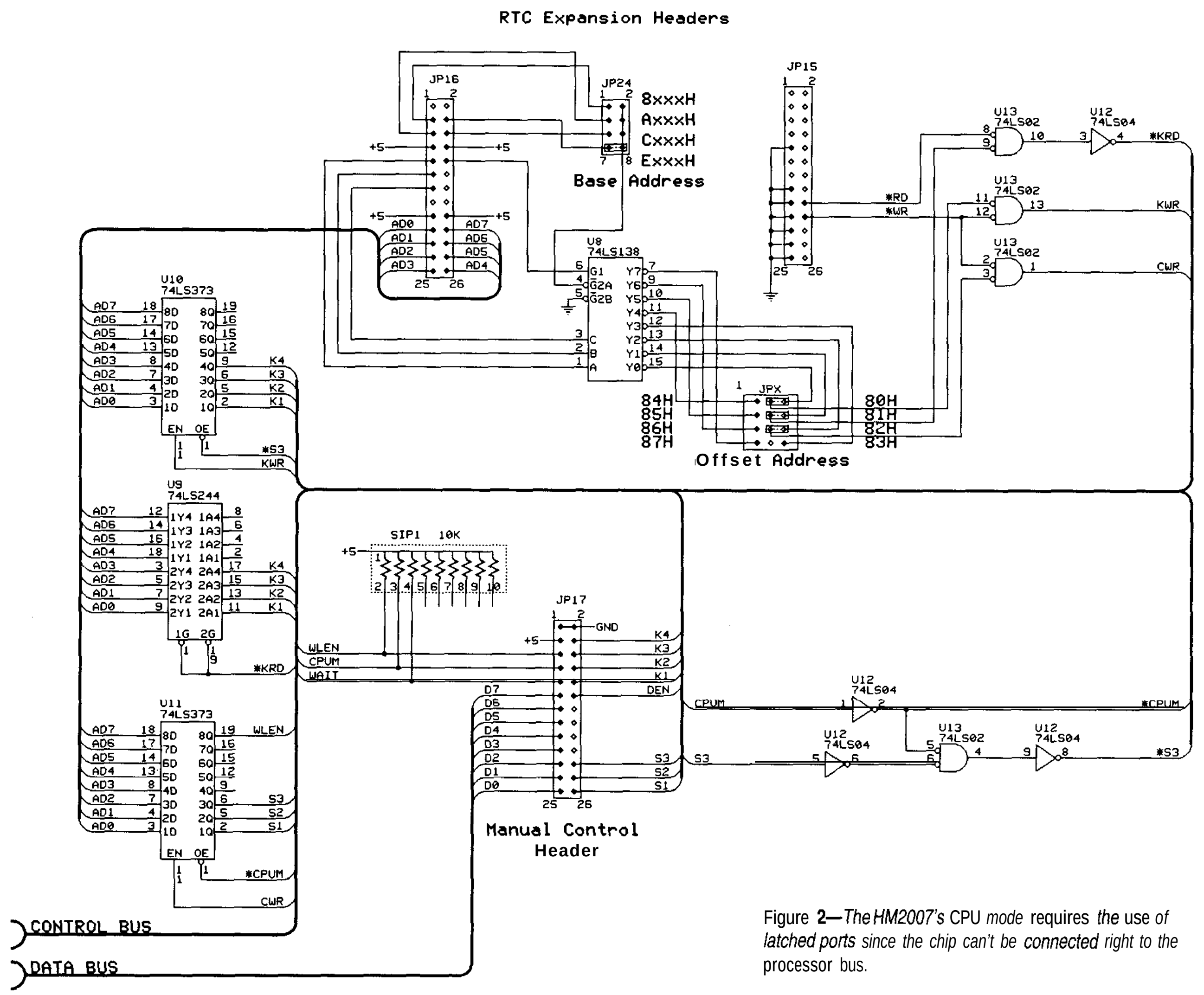


Figure 2—The HM2007's CPU mode requires the use of latched ports since the chip can't be connected right to the processor bus.

visual feedback about the state of the HM2007.

Using the keypad, any word can be cleared, trained, or recognized. During training, two types of errors may be encountered. If the sound duration is either too short or too long, it is flagged by the system and displayed as BCD digits 55 or 66, respectively. The user must then retrain that word. Once training is complete, recognition mode is entered and successive sounds are compared to the trained patterns for a match. There is an internal value [a summation of differences in pattern matching) that must not be exceeded if the device is to indicate a match and display the matching word number. If this value is exceeded, then a 77 is displayed indicating no match.

Although the display register (two BCD nybbles) could be interrogated by a CPU, the manual mode does not offer any indicator of the confidence level for the recognition process. With manual mode, you are limited to the internal confidence setting to determine whether or not a match has been made.

CPU MODE

The CPU mode reconfigures the keypad's seven I/O lines as a bidirectional nybble bus [K-bus) and three control lines (S-bus). The first control line chooses between the HM2007's status register and data register. The second and third control lines are read and write enables. Pulling the RD line high forces the HM2007 to place either the status or data register's 4 bits on the K-bus. Pulling the WR line high forces the HM2007 to read the 4-bit K-bus.

Besides being busy, there are only four states the HM2007 can be in. These are reflected in the status register as the values 0, 1, 2, or 3 as shown in Table 1.

CPU commands instruct the HM2007 to perform a specific function. There are seven such commands as shown in Table 2. [Upon power-up, the HM2007 presents a status "2" and awaits a command.)

The HM2007 was not designed to be directly interfaced as an I/O device. Because of this, it requires a bidirectional &bit-programmable port or two nybble output registers and one nybble input register. I chose to use the latter so the circuit could be used with the RTC stacking series of controllers. Adding another peripheral to the existing RTC line only strengthens its effectiveness and versatility.

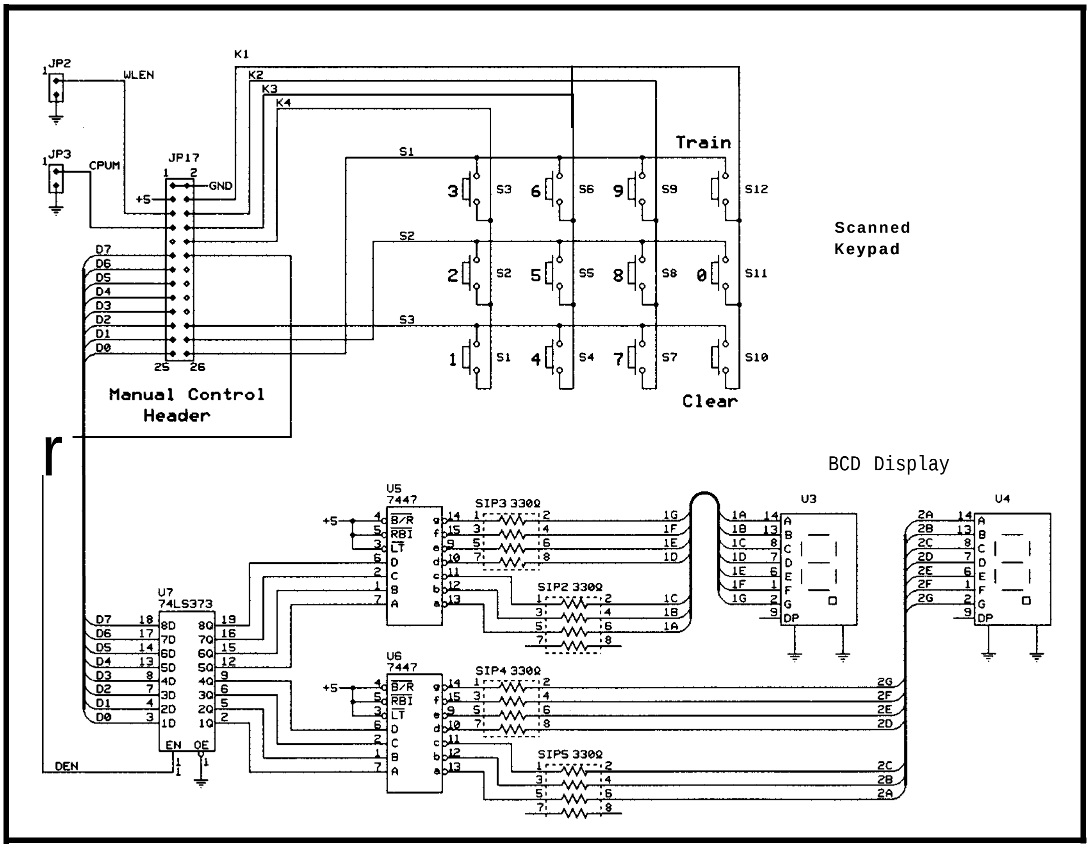


Figure 3-/n *addition to CPU mode, the HM2007 also supports a manual mode using push buttons and LED displays.*

Refer to Figure 1 for the base circuitry needed to use the HM2007. Audio from an external microphone is processed through an internal AGC and fed into the internal 8-bit A/D converter. Both amplitude and frequency data are extracted from the audio. Sampling is done for either 0.9 or 1.92 seconds, depending on the logic level presented to the WLEN input. The sampling length also determines the maximum number of words in the vocabulary (40 or 20, respectively). Note that this is a maximum and not a requirement. Vocabularies which are smaller in size have higher (correct) rates of recognition.

The vocabularies are held in SRAM. By battery backing the SRAM and holding the WAIT input at logic low during power-up, the vocabulary's data patterns can be preserved while the system is off. I placed the CPU interface circuitry shown in Figure 2 on the same proto board with the HM2007. I could have used port 1 of the 8052 as the interface, but I wanted to make the HM2007 look like an I/O device so it could be used with the other RTC processor boards. I also chose to use discrete registers for clarity as opposed to an 8255, 6821, or other multiregis-tered part.

By wiring up a header with all the signals necessary to use the HM2007 in manual mode, I built a second (and optional) piggyback proto board to hold the keypad and display. This can be used as an alternate method of training and testing. When the CPU input is pulled to a logic low (by the jumper on this board), the HM2007 is placed into manual mode. In this mode, external circuitry disables the I/O register outputs which share the HM2007's S- and K-bus I/O lines with the keypad.

HM2007 SEMAPHORE

The CPU mode's K-bus user output register is only enabled when the S3 control line (WR) is pulled to a logic high by the S-bus user output register. A high on the HM2007's S3 input indicates that the user has placed information on the K-bus and the information can now be read by the chip. The HM2007 receives both commands and data using the above procedure.

Likewise, while a logic high is placed on the S2 input (RD), the HM2007 will output information on the K-bus. The user determines whether the information comes from the HM2007's status register or data register by the state of S1: logic low for status and logic high for data.

Since the HM2007 is interfaced through I/O ports as opposed to the CPU bus, the user must constantly poll the status register to determine what state the HM2007 is in. The command sequences, therefore, contain intertwined software handshaking.

EXERCISING THE HARDWARE

Writing a BASIC subroutine for each command was easy (see Listing 1) using the flowcharts in the data sheet as a guide. All seemed like a piece of cake until I ran into problems getting back the word number and score from the HM2007. When WLEN was high (which puts the HM2007 into the higher 40 word, but shorter 0.9 second/per word mode), the chip refused to present the score. The program would just hang. Since there is no reset on the HM2007, things got locked up tighter than Fort Knox.

After considerable frustration with checking and rechecking my wiring, flowcharts, and coding, I broke down and called for assistance. I talked with Louis Gerhardy of the Summa Group.

"Louis. I would like to report a few errors I've found in the data sheets you faxed me," I offered as if a carrot on a stick.

"Sure, what ya got?" the enthusiastic voice replied.

"I'm guessing the status check on flowchart 2 should be a value 1 instead of 2 and the command in flowchart 7 should be 7?"

"Yes, yes that is correct," came the acknowledgment.

"Well, if there are errors in these two charts, is it possible there is a problem with chart 3?" I asked gingerly. A short pause stretched into a long one.

Finally, after what sounded like a bunch of paper shuffling, "I do have a note scratched in the border here. Bug in result command when.. .when...it looks like WLEN, yes, WLEN is high. Work around is drop WLEN to logic low when in result command. Does that make any sense!"

"You're asking me!" I thought to myself, then answered, "No, but let me think about it. Thanks." Louis must be as frustrated as I; he handles this device and is not an applications engineer.

Back in the code, I altered the result routine to twiddle WLEN low while asking for the data (a no-no according to the data sheet). As if by magic the command now returned the proper response. Gotcha. Grrr.

Once I had words trained, I needed a way to offload them so I wouldn't have to retrain them if the memory backup failed. I use ProComm on my PC to give me a smart terminal interface for the RTC52. By using the line pacing protocol, program uploads to the processor are paused after each line (carriage return/linefeed) to allow the RTC52 to process the line and catch up. When ready, the RTC52 automatically returns the ">" character. ProComm resumes sending the next line after receiving that character.

Serial uploading from the PC using BASIC's GET or I N PUT commands is usually not successful. BASIC is slow compared to the rapid rate of a serial transmission, so characters always get lost. Lost, unless handled like the line pacing of a program upload. This means one character at a time.. .or one string.

The word data coming from the HM2007 is in 8-nybble blocks, so why not put those eight nybbles in a string and move it that way! One other

Listing 1—*Following the flowcharts* in the *HM2007 datasheet makes writing* the *actual support code much easier. However, watch for undocumented pitfalls.*

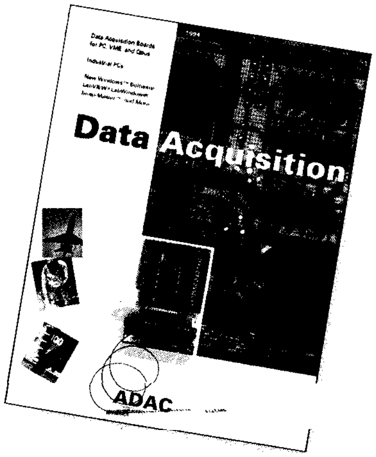
```
10 STRING 3281,80
20 BASE=OEOOOH
30 KRD=BASE+85H
40 KWR=BASE+84H
50 CWR=BASE+86H
60 RWR=BASE+87H
70 PRINT "Press 0 for 20 words @ 1.92 s or 1 for 40 words @ 0.9 s"
80 G=GET : IF (G=0) THEN GOTO 80
90 IF ((G<>30H).AND.(G<>31H)) THEN GOTO 70
100 IF (G=31H) THEN WL=80H ELSE WL=O
110 REC=1 : TRN=2 : RSL=4 : L=5 : DNL=6 : RST=7
170 SRD=2+WL : DRD=3+WL
180 DWR=5+WL : SWR=4+WL
190 REM STATUS 1=READY FOR VOICE INPUT
195 REM STATUS 2=READY FOR COMMAND A COMMAND
200 REM STATUS 3=READY FOR WLSN OR AVAILABLE
210 REM STATUS O=READY FOR MSN OR AVAILABLE
220 PRINT "Speech Recognition Demo using the HM2007"
230 PRINT
240 XBY(CWR)=SRD
250 PRINT "Checking HM2007 status"
260 S=2 : GOSUB 1000
270 PRINT
280 PRINT " Press Menu Selection"
290 PRINT
300 PRINT "1 Train a Word"
310 PRINT "2 Recognize a Word"
320 PRINT "3 Save a Vocab"
330 PRINT "4 Load a Vocab"
340 PRINT "5 Turn ",
345 IF (Q=0) THEN PRINT "on ", ELSE PRINT ɔff "
350 PRINT "status reporting"
360 PRINT "6 Clear all memory"
370 PRINT "7 End"
380 G=GET
390 IF (G=0) THEN 380
400 IF (G<31H.OR.G>37H) THEN 380
410 IF (G=31H) THEN GOTO 3000
420 IF (G=32H) THEN GOTO 4000
430 IF (G=33H) THEN GOTO 6000
440 IF (G=34H) THEN GOTO 7000
450 IF (G=35H) THEN Q=(Q+1).AND.1
460 IF (G=36H) THEN GOTO 2000
470 IF (G=37H) THEN STOP
480 GOTO 230
1000 REM status check
1010 NS=(XBY(KRD).AND.3)
1020 IF (NS=S) THEN RETURN
1030 IF ((Q=0).OR.(OS=NS)) THEN GOTO 1000
1040 OS=NS: PRINT "Checking Status ",
1050 IF (S=1) THEN PRINT "- Ready for Audio Input"
1060 IF (S=2) THEN PRINT "- Ready for Command"
1070 IF (S=3) THEN PRINT "- LSB Ready"
1080 IF (S=0) THEN PRINT "- MSB Ready"
1090 GOTO 360
2000 REM CLEAR ALL MEMORY
2010 XBY(CWR)=0 : XBY(CWR)=SRD
2020 S=2 : GOSUB 1000
2030 XBY(CWR)=0 : XBY(KWR)=RST : XBY(CWR)=DWR
2040 XBY(CWR)=0 : XBY(CWR)=SRD
2050 S=2 : GOSUB 1000
2060 GOTO 230
3000 REM TRAIN
3010 XBY(CWR)=0 : XBY(CWR)=SRD
3020 S=2 : GOSUB 1000
3030 PRINT
3040 PRINT "Enter the word number to train (1-",(WL*20)+20,")",:
     INPUT N
3050 IF ((N<1).OR.(N> (WL*20)+20))) THEN GOTO 3030
3060 PRINT "Type in a description of the audio for word ",N,:
INPUT $(N)
3070 LSNN=N.AND.OFH
3080 MSNN=INT(N/16)
3090 XBY(CWR)=0: XBY KWR)=TRN
```

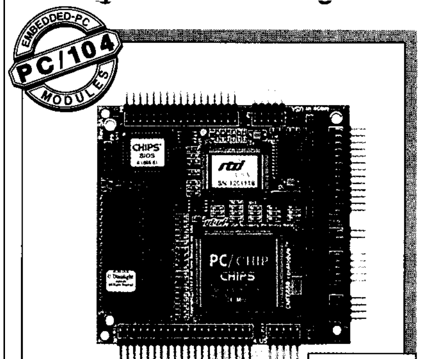
(continued)

Listing 1-continued

```
3100 XBY(CWR)=0 : XBY(CWR)=SWR
3110 XBY(CWR)=0 : XBY(CWR)=SRD
3120 S=3 :  GOSUB 1000
3130 BY(CWR)=0 : XBY(KWR)=LSNN
3140 XBY(CWR)=0 : XBY(CWR)=SWR
3150 XBY(CWR)=0 : XBY(CWR)=SRD
316C S=0: GOSUB  1000
3170 XBY(CWR)=0 : XBY(KWR)=MSNN
3180 XBY(CWR)=0 : XBY(CWR)=SWR
3190 XBY(CWR)=0 : XBY(CWR)=SRD
3200 S=1: GOSUB  1000
3210 PRINT "Please Speak Now..."
3220 GOSUB 5000
3230 IF (R>40) THEN PRINT "Try Again" :  GOTO 3070
3240 GOTO 230
4000 REM RECOGNIZE
4010 XBY(CWR)=0 : XBY(CWR)=SRD
4020 S=2: GOSUB  1000
4030 XBY(CWR)=0 : XBY(KWR)=REC
4040 XBY(CWR)=0 : XBY(CWR)=SWR
4050 XBY(CWR)=0 : XBY(CWR)=SRD
4060 S=1: GOSUB  1000
4070 PRINT "Please Speak Now..."
4080 IF (XBY(KRD)=1) THEN 4080
4090 GOSUB 5000
4100 IF (R>40) THEN PRINT "Try Again" :  GOTO 4000
4110 GOTO 230
5000 REM RESULT
5010 XBY(CWR)=0 : XBY(CWR)=SRD
5020 S=2: GOSUB  1000
5030 XBY(CWR)=0 : XBY(KWR)=RSL
5040 XBY(CWR)=0 : XBY(CWR)=SWR
5050 XBY(CWR)=0 : XBY(CWR)=(SRD.AND.7FH)
5060 P=2
5070 PRINT "Looking for LSN"
5080 S=3: GOSUB  1000
5090 XBY(CWR)=0 : XBY(CWR)=(DRD.AND.7FH)
5100 V=(XBY(KRD).AND.0FH)
5110 XBY(XCWR)=0 : XBY(CWR)=(DWR.AND.7FH)
5120 XBY(CWR)=0 : XBY(CWR)=(SRD.AND.7FH)
5130 PRINT "Looking for MSN"
5140 S=0: GOSUB  1000
5150 XBY(CWR)=0 : XBY(CWR)=(DRD.AND.7FH)
5160 V=V+((XBY(KRD).AND.0FH)*16)
5170 XBY(CWR)=0 : XBY(CWR)=(DWR.AND.7FH)
5180 XBY(CWR)=0 : XBY(CWR)=(SRD.AND.7FH)
5190 P=P-1.
5200 IF (P=1) THEN R=V : GOTO 5070
5210 S=2: GOSUB  1000
5220 PHO. "The word recognized was ",R
5230 PRINT "The score was ",V
5240 RETURN
6000 REM UPLOADING
6010 PRINT
6020 PRINT "Enter the word number to save (1-",((WL*20)+20),")",:
      INPUT N
6030 IF (N<1.OR.N>((WL*20)+20)) THEN GOTO 6020
6040 IF (ASC($(N),1)=13) THEN PRINT "Word ",N," NOT  rained" :
GOTO 6000
6050 PRINT "Press any key when ready to accept data"
6060 G=GET : IF (G=0) THEN GOTO 6060
6070 MSNZ=INT(N/16) : LSNZ=N-(MSNZ*16)
6080 XBY(CWR)=0 : XBY(CWR)=SRD
6090 S=2:   GOSUB 1000
6100 XBY(CWR)=0  : XBY(KWR)=UPL
6110 XBY(CWR)=0 .: XBY(CWR)=SWR
6120 XBY(CWR)=0 : XBY(CWR)=SRD
6130 S=3 : GOSUB  1000
6140 XBY(CWR)=0 : XBY(KWR)=LSNZ
6150 XBY(CWR)=0  : XBY(CWR)=SWR
6160 XBY(CWR)=0 : XBY(CWR)=SRD
6170 S=0 : GOSUB  1000
6180 XBY(CWR)=0 : XBY(KWR)=MSNZ
6190 XBY(CWR)=0 : XBY(CWR)=SWR
6200 XBY(CWR)=0 : XBY(CWR)=SRD
6210 S=3 : GOSUB  1000
```

(continued)





Listing 1-continued

```
622C XBY(CWR)=0 :  XBY(CWR)=DRD
623C LSNL=(XBY(KRD).AND.OFH)
624C XBY(CWR)=0 :  XBY(CWR)=DWR
6250 XBY(CWR)=0 :  XBY(CWR)=SRD
6260 S=0: GOSUB  1000
6270 XBY(CWR)=0 :  XBY(CWR)=DRD
6280 MSNL=(XBY(KRD).AND.OFH)
6290 L=MSNL*16+LSNL: IF (L=0)  THEN  L=256
6300 XBY(CWR)=0 :  XBY(CWR)=DWR
6310 PRINT $(N),CHR(07EH),CHR(N+20H),CHR(LSNL+20H),CHR(MSNL+20H)
6320 L=L-1
6330 XBY(CWR)=0 :  XBY(CWR)=SRD
6340 S=3: GOSUB  1000
6350 XBY(CWR)=0 :  XBY(CWR)=DRD
6360 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6370 XBY(CWR)=0 :  XBY(CWR)=DWR
6380 XBY(CWR)=0 :  XBY(CWR)=SRD
6390 S=0:GOSUB  1000
6400 XBY(CWR)=0 :  XBY(CWR)=DRD
6410 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6420 XBY(CWR)=0 :  XBY(CWR)=DWR
6430 XBY(CWR)=0 :  XBY(CWR)=SRD
6440 S=3: GOSUB  1000
6450 XBY(CWR)=0 :  XBY(CWR)=DRD
6460 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6470 XBY(CWR)=0 :  XBY(CWR)=DWR
6480 XBY(CWR)=0 :  XBY(CWR)=SRD
6490 S=0: GOSUB  1000
6500 XBY(CWR)=0 :  XBY(CWR)=DRD
6510 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6520 XBY(CWR)=0 :  XBY(CWR)=DWR
6530 XBY(CWR)=0 :  XBY(CWR)=SRD
6540 S=3: GOSUB  1000
6550 XBY(CWR)=0 :  XBY(CWR)=DRD
6560 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6570 XBY(CWR)=0 :  XBY(CWR)=DWR
6580 XBY(CWR)=0 :  XBY(CWR)=SRD
6590 S=0: GOSUB  1000
6600 XBY(CWR)=0 :  XBY(CWR)=DRD
6610 PRINT CHR((XBY(KRD).AND.OFH)+20H),
6620 XBY(CWR)=0 :  XBY(CWR)=DWR
6630 XBY(CWR)=0 :  XBY(CWR)=SRD
6640 S=3: GOSUB  1000
6650 XBY(CWR)=0 :  XBY(CWR)=DRD
6660 PRINT CHR((XBY(KRD).AND.OFH)+20H).
6670 XBY(CWR)=0 :  XBY(CWR)=DWR
6680 XBY(CWR)=0 :  XBY(CWR)=SRD
6690 S=0: GOSUB  1000
6700 XBY(CWR)=0 :  XBY(CWR)=DRD
6710 PRINT CHR((XBY(KRD).AND.OFH)+20H)
6720 XBY(CWR)=0 :  XBY(CWR)=DWR
6730 IF (L<>0) THEN GOT0  6320
6740 PRINT "-END"
6750 GOT0 230
7000 REM DOWNLOADING
7010 PRINT
7020 PRINT "Ready to accept data"
7030 INPUT $(0)
7040 P=l
7050 IF (ASC($(0),P)=13) THEN PRINT "File Error" :  GOT0 230
7060 IF (ASC($(0),P)<>07EH) THEN P=P+1:  GOT0 7050
7070 IF (ASC($(0),P+1)<>45H) THEN GOT0 7100
7080 IF (ASC($(0),P+2)<>4EH) THEN GOT0 7100
7090 IF (ASC($(0),P+3)<>44H) THEN GOT0  7100  ELSE  PRINT  ">":
GOT0 230
7100 N=ASC($(0),P+1)-20H :  LSNL=ASC($(0),P+2)-20H :
      MSNL=ASC($(0),P+3)-20H
7110 L=(MSNL*16)+LSNL:  IF  (L=0)  THEN  L=256
7120 IF ((N<1).OR.(N>(WL*20+20))) THEN PRINT "Word ii invalid"
      GOT0 230
7130 PRINT "Now downloading word number ",N," of length ",L
7140 MSNN=INT(N/16) :  LSNN=N-(MSNN*16)
7150 XBY(CWR)=0 : XBY(CWR)=SRD
7160 S=2 :  GOSUB 1000
7170 XBY(CWR)=0 :  XBY(KWR)=DNL :  XBY(CWR)=SWR
7180 XBY(CWR)=0 :  XBY(CWR)=SRD
```

(continued)

Listing 1-continued

```
7190 S=3: GOSUB  1000
7200 XBY(CWR)=0 : XBY(KWR)=LSNN:  XBY CWR)=SWR
7210 XBY(CWR)=0 :  XBY(CWR)=SRD
7220 S=0: GOSUB  1000
7230 XBY(CWR)=0: XBY(KWR)=MSNN:  XBY CWR)=SWR
7240 XBY(CWR)=0 :  XBY(CWR)=SRD
7250 S=3: GOSUB  1000
7260 L=(L.AND.255) :  MSNL=INT(L/16) : LSNL=L-(MSNL*16)
7270 XBY(CWR)=0: XBY(KWR)=LSNL : XBY CWR)=DWR
7280 XBY(CWR)=0 :  XBY(CWR)=SRD
7290 S=0: GOSUB  1000
7300 XBY(CWR)=0 : XBY(KWR)=MSNL : XBY CWR)=DWR
7310  IF L=O THEN L=256
7320 L=L-1
7330  PRINT ">",
7340  INPUT $(0)
7350 XBY(CWR)=0 : XBY(CWR)=SRD
7360 S=3 : GOSUB  1000
7370 XBY(CWR)=0 : XBY(KWR)=ASC($(0),1 -20H :  XBY(CWR)=DWR
7380 XBY(CWR)=0 : XBY(CWR)=SRD
7390 S=0 : GOSUB  1000
7400 XBY(CWR)=0 : XBY(KWR)=ASC($(0),2 -20H   XBY(CWR)=DWR
7410 XBY(CWR)=0 : XBY(CWR)=SRD
7420 S=3: GOSUB  1000
7430 XBY(CWR)=0 :  XBY(KWR)=ASC($(0),3 -20H   XBY(CWR)=DWR
7440 XBY(CWR)=0 :  XBY(CWR)=SRD
7450 S=0: GOSUB  1000
7460 XBY(CWR)=0 :  XBY(KWR)=ASC($(0),4 -20H   XBY(CWR)=DWR
7470 XBY(CWR)=0 :  XBY(CWR)=SRD
7480 S=3: GOSUB  1000
7490 XBY(CWR)=0 : XBY(KWR)=ASC($(0),5 -20H   XBY(CWR)=DWR
7500 XBY(CWR)=0 : XBY(CWR)=SRD
7510 S=0 : GOSUB  1000
7520 XBY(CWR)=0 : XBY(KWR)=ASC($(0),6 -20H   XBY(CWR)=DWR
7530 XBY(CWR)=0 : XBY(CWR)=SRD
7540 S=3 : GOSUB  1000
7550 XBY(CWR)=0 : XBY(KWR)=ASC($(0),7 -20H   XBY(CWR)=DWR
7560 XBY(CWR)=0  : XBY(CWR)=SRD
7570 S=0 : GOSUB  1000
7580 XBY(CWR)=0 : XBY(KWR)=ASC($(0),8)-20H : XBY(CWR)=DWR
7590  IF (L<>0) THEN GOT0 7320
7600  PRINT ">"
7610 GOT0 7030
8000 REM   SAVES PROGRAM INTO NVRAM A 8000H  (NOT  NECESSARY)
8010 XBY(8010H)=55H
8020 FOR X=200H TO (200H+LEN)
8030 XBY(X+7E11H)=XBY(X)
8040 NEXT X
8050 XBY(8000H)=34H
8060 XBY(8001H)=0FFH
8070 XBY(8002H)=0DCH
8080 XBY(8003H)=7FH
8090 XBY(8004H)=0
```

thought about the data is appropriate here: since it is in nybbles (O-15), why not add 20h to each value to make them printable characters? This avoids control-character unpleasantries.

Now that we know how the data must be formatted to upload properly into the RTC52 and the HM2007, the download portion is just a matter of collecting the data and outputting it in the compatible format. Each word's data is output separately and collected by ProComm into separate .DAT files. You can concatenate any of these words (.DAT files) into a single file for easy handling using any ASCII editor. This also lets you prepare separate files of differing vocabularies.

SPEAKING OF APPLICATIONS

Picture this: Saturday afternoon. The playoffs are on and you've got multiple teams to keep track of. A cold beer in one hand and a 6″ hoagie in the other. You speak, "TV.. .channel.. .O.. 2." The channel changes and you're there as a 30-footer is sunk. You decide to tape the other channel so you won't miss a moment of these last four minutes. You speak again, "VCR.. record." As the front panel indicates life, you think you hear something. Once more you speak, "TV.. .mute." Ring, ring, it's the phone, but it's answered almost immediately by one of the kids. You don't get phone calls anymore; remember, you just live here. "TV.. .mute," you repeat again. Instantly, you're back in the action.

This application uses the RTC52 and the MCIR-Link in an interactive RS-232 mode. The MCIR accepts "SMx" [where x is a number) as a command to send out a prerecorded IR transmission. All the television and VCR functions were trained on the MCIR, and when a spoken word is recognized, the RTC52 outputs the appropriate "SMx" command to the MCIR-Link. A null-modem cable is needed here because both the RTC52 and the MCIR-Link look like DCE devices.

I admit this is not all that practical, but I bet you can come up with a situation where it does make sense. Or maybe like me, you just want to have a little fun.

P.S. HMC, if you're listening, I think the hardware implementation and documentation could be improved, just a bit. But you've probably already ('scuze me] recognized that! ❑

Special thanks to Louis Gerhardy (The Summa Group Limited) and H.C. Lee (Hualon Microelectronics Corp.).

Jeff ***Bachiochi (pronounced*** *"BAH-key-* ***AH-key")*** is an electrical engineer on ***the Computer Applications*** *Journal's* ***engineering*** *staff.* ***His background includes product design and manufacturing. He may be reached at*** *jeff.bachiochi@circellar.com.*

CONTACT

The Summa Group Limited
1 California St., Ste. 1940
San Francisco, CA 94111
(415) 2880390
Fax: (415) 288-0399

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

In the Realm of the Sensors

SILICON UPDATE

Tom Cantrell

In any embedded control or data collection system, the sensors are just as important as the controller. While typically analog, sensor interfaces are slowly making the transition to digital. Check out what's new.

this year's "Sensors Expo" was held at the Disneyland Hotel-a nice place if you don't mind pictures of Mickey Mouse and Goofy instead of the paint-by-the-numbers masterpieces that grace the typical tourist trap.

However, except for the few who qualify as an electronic Michelangelo, a paint-by-the-numbers design mentality is the way to go lest your time-to-market be measured in Sistine Chapel–like terms.

It's an oft-repeated theme of "Silicon Update" that much of the art, albeit black, of scientific and industrial applications revolves around sensor interfacing and analog design antics.

Sure, you can attribute some of my "analogphobia" to a simple case of bit-head bias. Nevertheless, even an objective observer must admit that the process control world is way behind the digital eight ball.

The inertia to do things the "good old way" is strong, but under constant attack by wave after wave of ever more powerful digital ICs. Though reluctant, the data acquisition and process control folks are slowly, but surely, moving into the 1s-and-0s age.

"G" WHIZ

Don't expect the digital takeover to happen overnight. The first tentative steps will include chips that combine analog and digital interfaces to ease the way for wary designers.

Consider the AMP ACH04-08 accelerometer ($40 qty. 1). The device integrates piezoelectric sensors configured to measure acceleration in three axes: Y, Z, and rotation around Z as shown in Figure 1.

While piezo accelerometers are nothing new, the ACH04-08 is innovative because of its integrated interface logic that culminates in—hooray-a digital output mode.

As shown in Figure 2, the three raw inputs are fed to amplifiers with programmable gain which, in turn, feed comparators with programmable threshold. The result is a digital Q (or Q*) output flagging a "shock" defined by the user-specified gain, threshold, and reference voltage: 3-53 g's for the Y- and Z-axis and 125–1660 rad/s^2 rotation about Z. As shown in Figure 3, once awoken from low-power (200 µA)

mV/g, mV/rad/s^2) o r amplified (7.5–100 mV/g, mV/ rad/s^2)

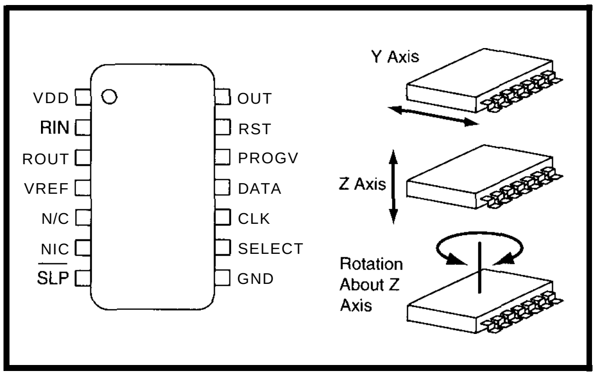


Figure 1—*The AMP ACH04-08* accelerometer measures *acceleration in three axes: Y, Z, and rotation around Z.*

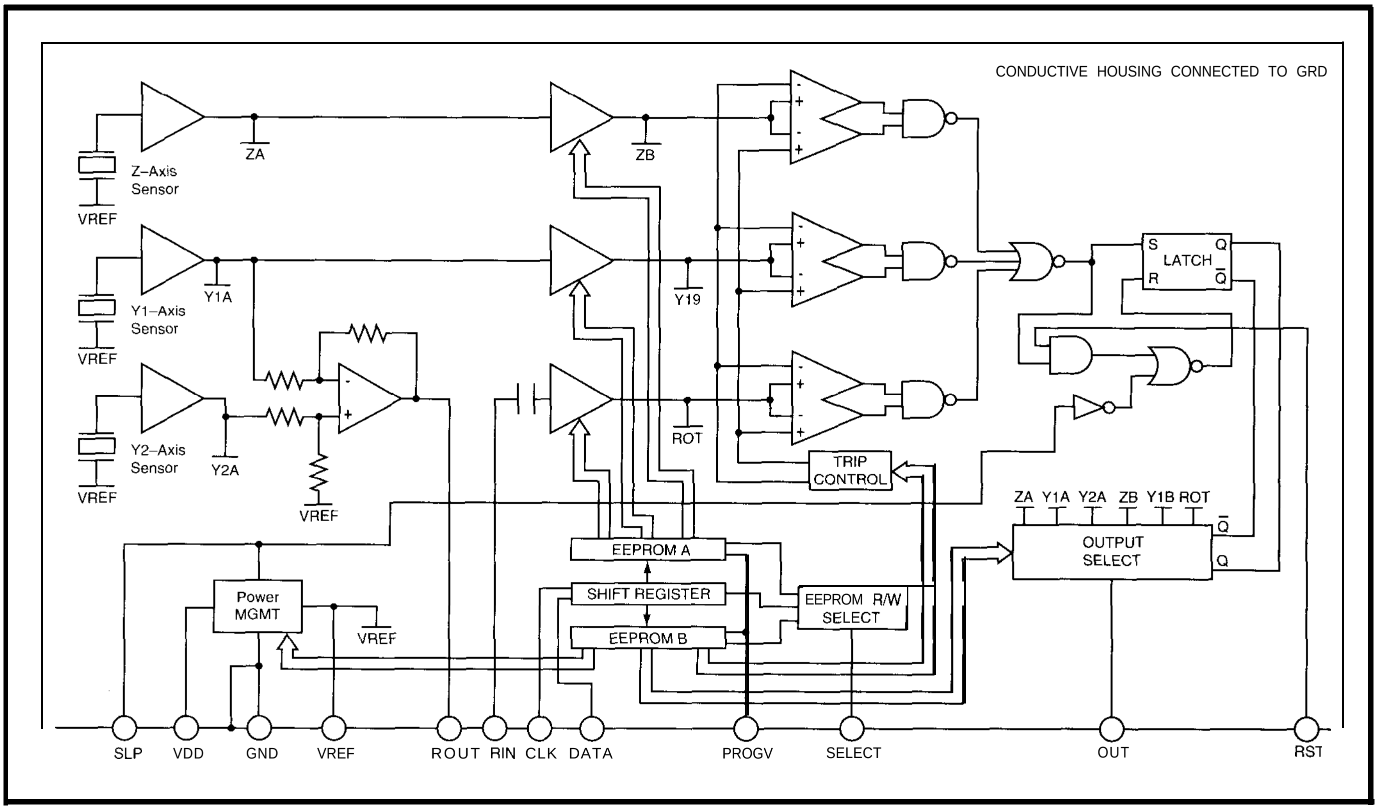


Figure 2—*While piezo* accelerometers are *nothing new, the ACH04-08 is innovative because of ifs integrated interface logic that produces digital outputs.*

The idea of offering both digital and analog outputs is fine, but I question funneling both through the same pin (though note that the rotation output-ROUT-is separately available). For instance, if two pins were provided, it would be possible to connect the digital output to a micro's interrupt request and the analog out to an A/D converter. Then, the micro could ignore minor jiggles, yet still take detailed readings via the ADC in response to a shock interrupt.

At first glance, it seems fairly easy to duplicate two-pin functionality by dynamically reprogramming the OUTPUT SELECT bits and adding a gate or two to route the single output pin. However, beware of gotchas when contemplating such a scheme.

Clocking the data into the on-chip shift register via the CLK and DATA lines takes only a dozen or so microseconds, but the minimum programming time (i.e., subsequent pulse on the PROGV pin) is on the order of 10 ms each for EEPROM A (gain) and EEPROM B (VREF, TRIP, and OUTPUT SELECT). Assuming the idea is to interrupt with the digital output and then quickly sample the analog reading, missing the first, and likely most interesting, 20 ms while making the mode switch would seem problematic for high-speed applications.

The 20-ms reprogramming time that might be "too slow" in an application sense may be "too fast" in terms of the 100k EEPROM write-cycle endurance limit. A worst-case design that did nothing but switch back and forth continuously could die in less than an hour! Yet, a more careful design, perhaps coupled with a little reality-checking software, can obviate the write endurance concern, but be careful not to simply assume (as in ASS-of-U-and-ME) it away.

If you still want to reprogram in *situ,* start digging around in your system for a rather unlikely -16-V (±0.5 V, thank you) PROGV voltage.

Even if you choose to simply use the chip in the clearly intended analog or digital (but not both) fashion, it might still be wise to arrange for reprogramming. I think there is a big, yet largely overlooked, gotcha associated with the ACH04-08 and other

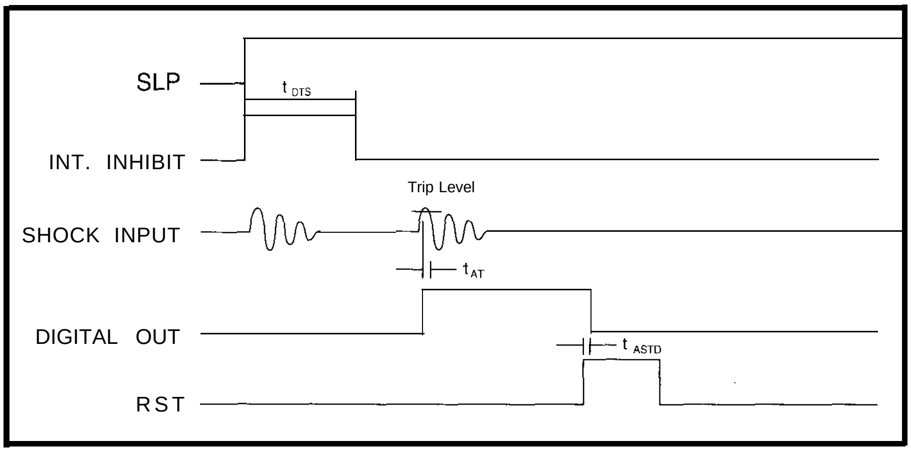
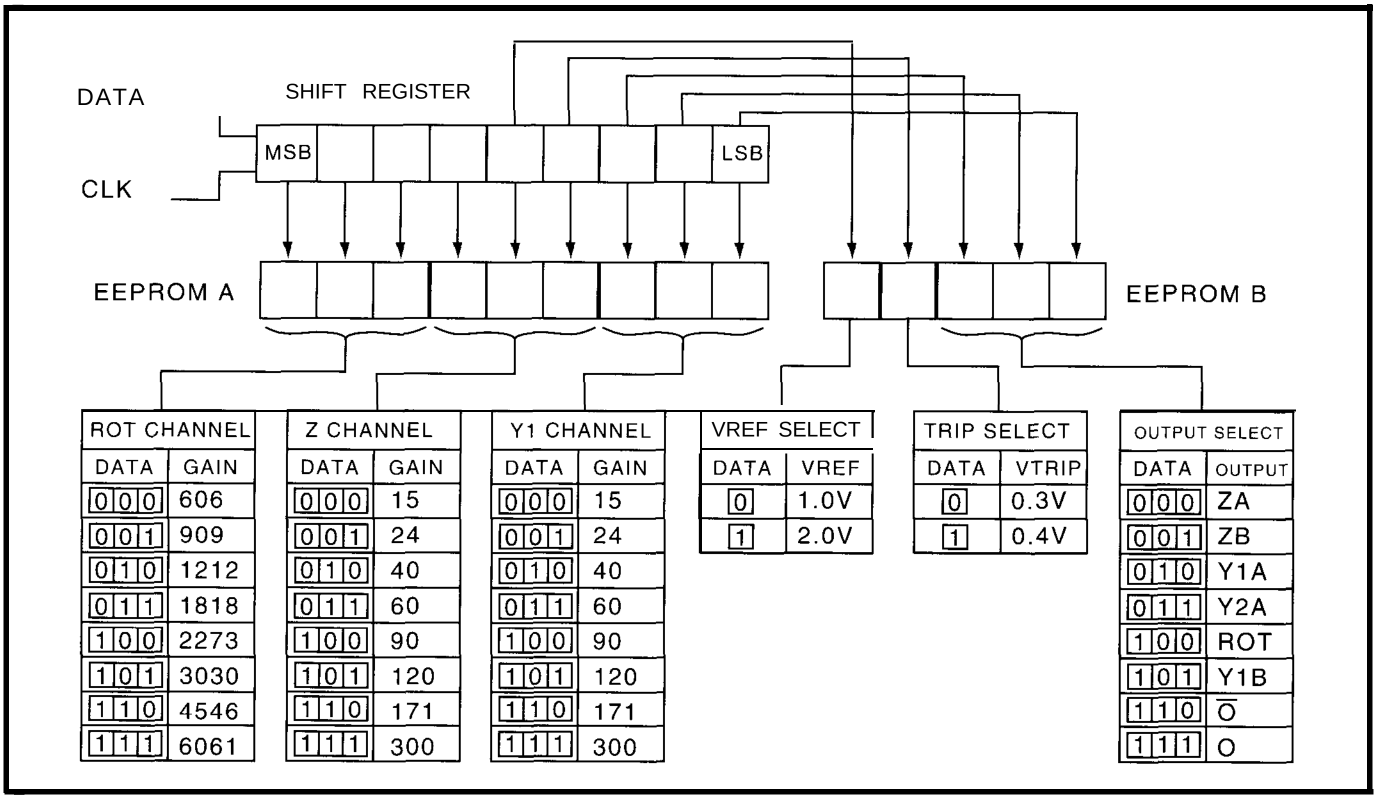


Figure 3—*Once the ACH04-08 is awoken from low-power sleep mode, the output is asserted when a shock is defected and remains latched until c/eared by a subsequent reset input.*



ROT CHANNEL	
DATA	GAIN
000	606
001	909
010	1212
011	1818
100	2273
101	3030
110	4546
111	6061

Z CHANNEL	
DATA	GAIN
000	15
001	24
010	40
011	60
100	90
101	120
110	171
111	300

Y1 CHANNEL	
DATA	GAIN
000	15
001	24
010	40
011	60
100	90
101	120
110	171
111	300

VREF SELECT	
DATA	VREF
0	1.0V
1	2.0V

TRIP SELECT	
DATA	VTRIP
0	0.3V
1	0.4V

OUTPUT SELECT	
DATA	OUTPUT
000	ZA
001	ZB
010	Y1A
011	Y2A
100	ROT
101	Y1B
110	$\overline{O}$
111	O

Figure 4—*The ACH04-08's* user *settings* (gain, *threshold, etc.) are he/d in 14 bifs of EEPROM*

EEPROM-based chips: data retention time.

Yes, ten years is a long time in the electronics business and, in the faster-paced sectors, it's marginally valid to ASS-U-ME that your gizmo will have long since become a door stop or boat anchor. On the other hand, industrial applications tend to be much more long-lived, and examples abound of machines that outlast their masters.

So, should you decide to ignore the data retention issue, I suggest you mark your calendar. It might be wise to be on vacation or otherwise be "out of the office" 10 years from now when the phone starts ringing.

SONIC TONIC

First-generation ultrasonic transducers were little more than speaker-microphone combinations, leaving the burden of driver modulation, echo detection, and other housekeeping chores to the system designer. Now, reflecting the "smarter is better" trend, transducers are starting to integrate the various bits and pieces-amps, filters, power supply, and so on-needed for a total solution.

Consider the EDP (Electronic Design & Packaging) Sonaswitch 1750 (Photo 1) which measures distances from I" to 30', as is typical for ultrasonic rangers.

What isn't typical is the amount of glue logic integrated in the 1.5" x 6" harsh-environment-tolerant steel and polymer package.

First is a built-in high-efficiency switching power supply that operates from a wide 10–30-VDC (0.5 A max) input range. Though they haven't gone as far as offering digital output, on-board amplification and conditioning does allow them to offer eminently usable 10-bit analog outputs-both O-5 VDC and 4-20 mA. Ah, if only all analog gadgets would so easily mate with micros.

Having accepted the burden of providing meaningful outputs, the 1750 goes for the gusto with built-in temperature calibration, programmable slope (i.e., the polarity of the output can be inverted), and adjustable output filter, easing your micro's processing burden considerably. The settings are programmed via built-in RS-232 port.

Other goodies include push-button inputs and NPN outputs that support calibration (zero/span) and an automatic setpoint monitoring scheme with programmable hysteresis and glitch reduction.

At $450 in singles, the 1750 isn't cheap, but that's usually the case for heavy-duty steel gadgets. On the other hand, it does include DOS upload, download, calibrate, and monitor software.

Besides, if you're in a hurry and aren't making a zillion gizmos, the price is really quite reasonable. Figure it out-unless you're willing to roll your own interface logic and either work very fast or for minimum wage, the $450 may indeed be money well spent.

BIG BROTHER IS DRIVING

While everyone blabs about the "information superhighway," progress is being made on the equally intriguing "superhighway information" front.

What do you get when you combine Caltrans (our state highway bureaucracy), ex-bomb designers from Lawrence Livermore Labs, and a dash of high-tech diodes from HP? It may sound scary, but don't worry-it's only an “Automatic Vehicle Identification” (or “AVI”) system.

As shown in Figure 5, the system consists of a roadside “reader” that, despite its name, supports both reads and writes with a car-mounted “tag” via a 915-MHz (±13 MHz) RF link. The tag uses a “backscatter” approach in which the incoming RF is bounced back to carry the tag’s response. This reduces cost by eliminating the need for an RF generator in the tag. It also improves performance since the response frequency (i.e., that generated by the reader) is known exactly, which wouldn’t be the case if each tag had its own clock subject to variable tolerances and drift. Furthermore, the system is frequency “agile” in the sense that everything will still work if a roadside reader’s frequency must be changed to avoid interference.

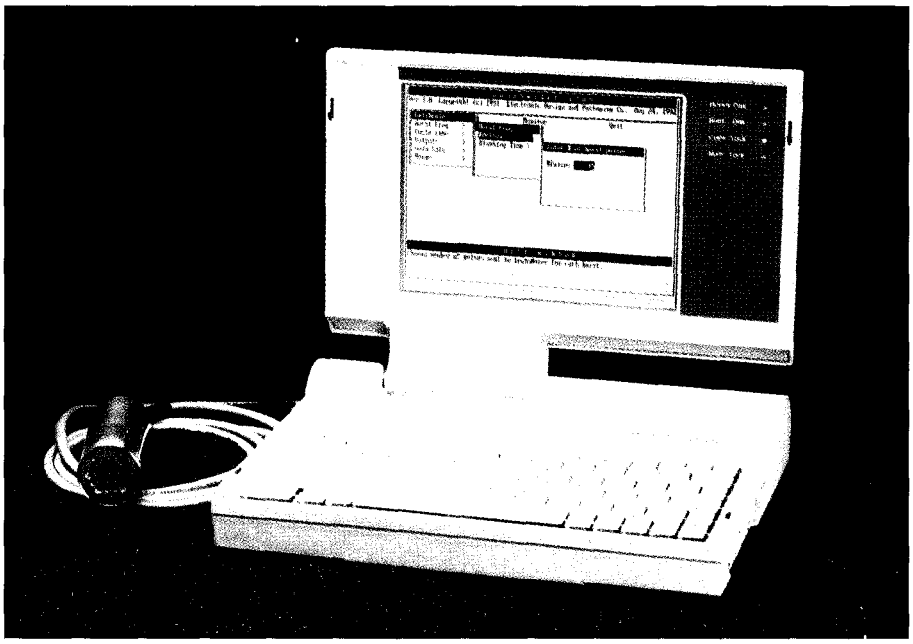


Photo 1 *—The EDP Sonaswitch 1750 measures distances from 1" to 30'.*

Ultimately, it all adds up to a 300k-bps bidirectional link between your wheels and Big Brother’s computers. This can support, for example, five read/write cycles (i.e., error correction) per lane on a four-lane highway, nailing scofflaws traveling at up to 100 MPH through a one-meter RF “trap.”

RF tags are going to be a big deal, not just for AVIs, but for all manner of ID applications up to and including bar code replacement. A detailed discussion of the concept won’t fit here. However, as a citizen and a driver, I will take a few moments to pontificate on the implications of AVI systems.

Ostensibly, AVIs are our friends, making life easier with automatic toll collection, directions to the next restroom, or whatever. What could possibly be wrong with that?

Of course, the road to ruin is always paved with good intentions. To me, the situation is analogous to the government’s attempts to “encourage” us to accept the so-called “Clipper” voice and data encryption technology whose most notable feature is that it gives the feds access to the “keys.” But don’t worry, they say, there are lots of “safeguards” and we really just want to catch “bad guys.” Trust us.

You’ve probably heard about the roadside “candid camera” automatic speeding ticket generators that use radar to clock a car’s speed and snap a picture of any hot rodders. Well, the skeptics and paranoids among you might easily imagine an AVI system put to equally dubious use. Why, the system could not only issue the speeding ticket, but directly debit your Visa card as well-no muss, no fuss.

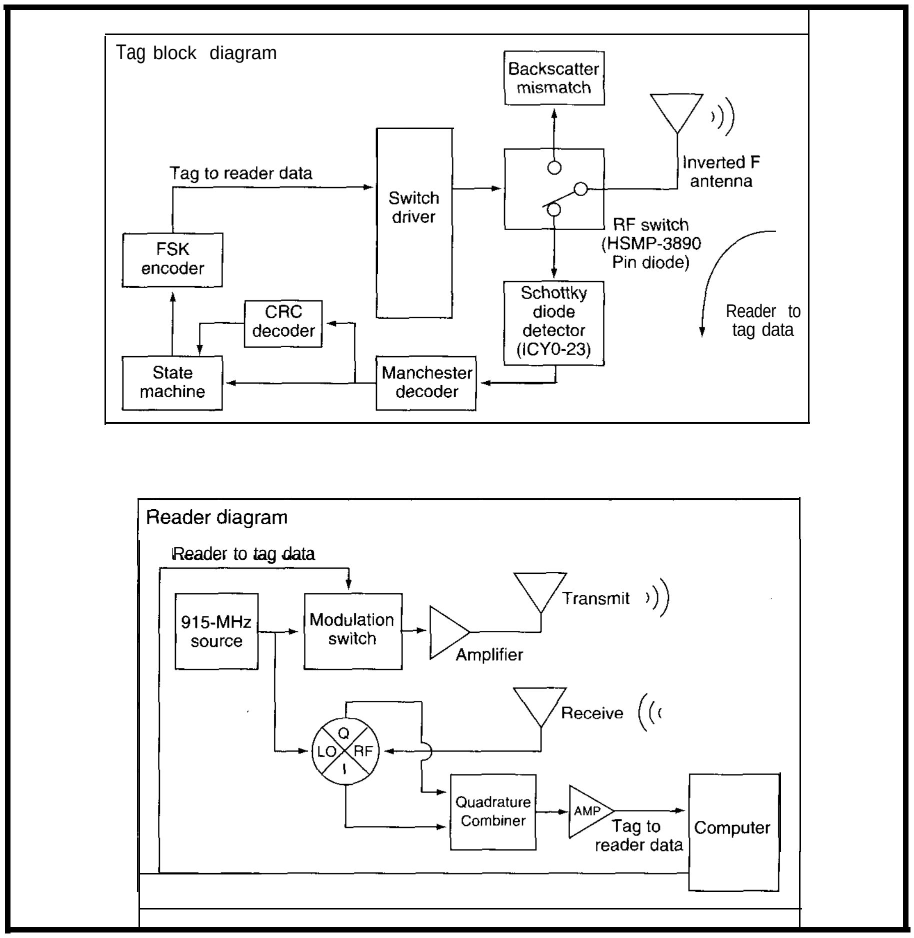


Figure **5**—*In the Automatic Vehicle Identification system being implemented by Caltrans, a roadside “reader” can both read and write car-mounted fags via an RF link.*

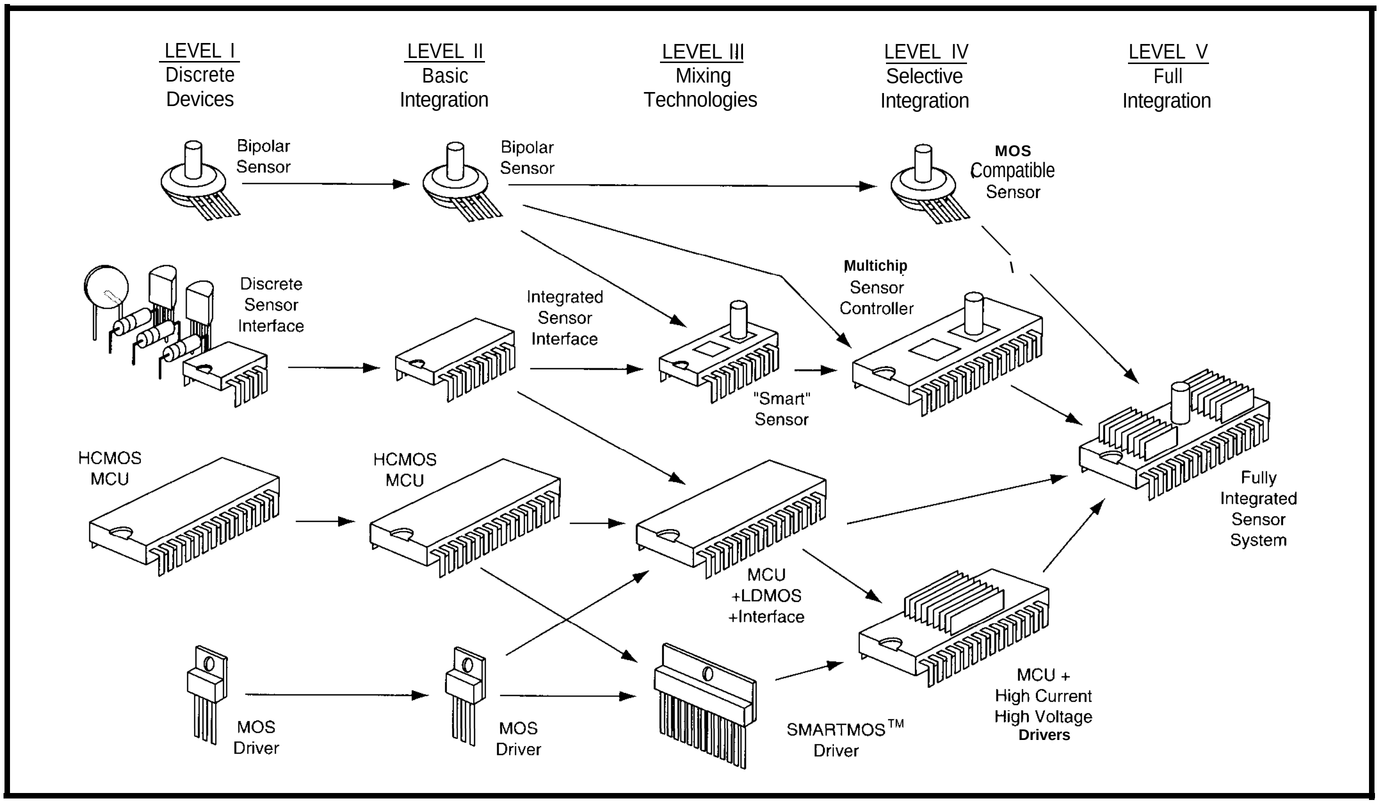


Figure 6—*The* integration path from analog *to digital for sensors can be broken down info five levels from discrete devices to full integration*

OK, OK so you've got to slow down whenever you spot a lurking AVI-they aren't going to be able to blanket every inch of asphalt are they? No, but should they draw the automakers into the unholy alliance, it wouldn't be hard to "log" your bad driving habits to later "tattle" when you do pass an AVI.

However, an AVI-based "Officer Speed" scheme suffers at the hands of our justice system since it turns out all you have to do is say you happened to loan your car to "some person whose name escapes me." Yes, the good old "cars don't speed, drivers do" defense to the rescue.

If the system were only one-way (i.e., tag data to reader), that would be the end of it. But a two-way system allows our leaders to take active countermeasures. Maybe the same "timeout" strategy used for obnoxious kids is in order-if the AVI detects your transgression, it can tell your car to shut off, giving you time to contemplate the error of your ways.

Nobody knows how all this will turn out, but I do predict that once we let Big Brother into the driver's seat, he'll be there for good.

OF SINKS, KITCHEN, AND HEAT

Having started down the slippery slope of combining analog sensors with digital intelligence, the question is, where it will all end?

According to Motorola, the integration path can be broken down into five levels from discrete devices (Level I) to full integration (Level V) as shown in Figure 6.

For many, many years we've used and abused Level I techniques. Only recently have Level II-III options, such as those mentioned in this article, emerged. Level IV-V devices remain on the horizon for now.

Putting everything but the kitchen sink on a chip raises a couple of issues. Basically, what makes money and what makes sense?

From the manufacturer's point of view, assuming integrating all the goodies is possible, the concern boils down to keeping die size near that which provides optimal yields.

Most of you may be aware that die cost doesn't necessarily scale linearly with size. A die with twice the area may cost more than twice the smaller device. In technospeak, the phenomenon is referred to as the "iso-defect curve" that relates die size to yield. The curve is nonlinear because wafers are characterized by a certain defect density, no matter how many die they may contain. Thus, it's easy to see that a larger die has a higher probability of running into a defect than a smaller one. In the above example, the big die might fail, but one of the two smaller ones will make it.

If die size is small enough, the iso-defect issue becomes fairly moot. Given, for example, 10 defects per wafer, whether there are 200 or 300 die per wafer matters relatively little. Boost die size into the 50 or 100 die per wafer range, though, and things start to get-pardon the pun—dicey.

From my (a user's) point of view, I'm all for ultimate integration since it gives me the freedom to choose the design-time versus unit cost tradeoff. Naturally, I'll use the availability of lower-cost, less-integrated alternatives to push the supplier to cut the highly integrated unit's price. Why should I care if it's hard for the supplier to make money on the "overintegrated" unit? (Of course, I'd also be the first to whine should they decide to quit supplying the "unprofitable" part).

Otherwise, my only concern about full integration is the migration of high-voltage and high-temperature functions on-chip at levels IV and V. I'm sure the chip designers will be careful to guard against IC meltdown, while power consumption and thermal management remain issues at the system level, whatever the number of chips inside.

Nevertheless, I've never really felt comfortable with chips that are too hot to touch, so talk of 40-W ICs tends to make me a little nervous.

It seems likely that power, voltage, and heat will remain a dividing line with chips falling into one of two camps. Since both camps believe the sky's the limit with integration, many systems may ultimately devolve to two chips. One chip will be a big ugly metal thing with lots of cooling fins or perhaps even liquid or thermocouple "active" cooling-dumb, but able to handle the juice needed to make something happen in the real world. The other will be a high-pin-count, low-voltage (3 V down to perhaps 1 V), plastic-packaged Cray-on-a-chip that does all the thinking. ❑

Tom Cantrell has been an engineer in Silicon Valley for more than ten years working on chip, board, and systems design and marketing. He can be reached at (510) 657-0264 or by fax at (510) 657-5441.

CONTACT

Hewlett-Packard
Communication Components Division
350 West Trimble Rd.
San Jose, CA 95131-1096
(408) 4354303
Fax: (408) 4354303

LLNL Transportation Program, L-644
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550
(510) 423-4497
Fax: (510) 423-9649

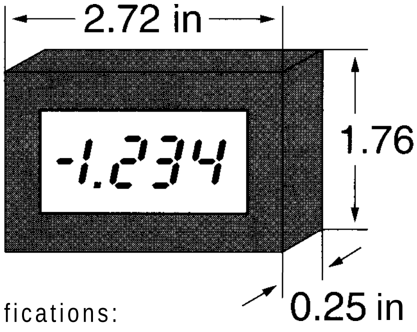
Motorola Semiconductor Products Sector MD 56-102
3102 North 56th St.
Phoenix, AZ 85018-6606
(602) 952-4103
Fax: (602) 952-4067

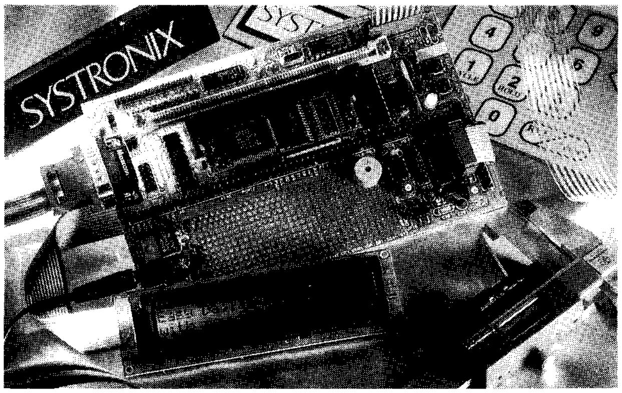
AMP, Inc.
Piezo Film Sensors
P.O. Box 799
Valley Forge, PA 19482
(215) 666-3500
Fax: (215) 666-3509

EDP
37666 Amrhein
Livonia, MI 48 150
(313) 591-9176
Fax: (313) 591-7852

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful



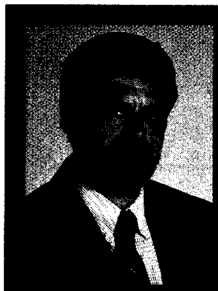


Speed Demon in 8031's Clothing

Exploring the DS80C320 Processor

EMBEDDED TECHNIQUES

John Dybowski



How do you increase the performance of an existing 8031 circuit by just changing the processor? Add a Dallas Semiconductor 80C320. Find out about other timing-related issues that also come into play.

The proclamation of the death of 8-bit processors could be viewed as a classic example of a high-profile blunder. Maybe it seemed like a safe call at the time, but whoever made this statement had little understanding of the dynamics at work in the embedded arena. Then again, if you're of a more cynical bent, you might suspect that there lies a more sinister motive behind what they were telling you.

As an example, I still remember in amusement how hard Intel tried to convince everyone that their 8096 architecture was the logical successor to the 805 1. They even went so far as to provide a code translator that would, presumably, take your 805 1 application and translate it into 8096 code. The fact that the two chips had memory and I/O models that were completely different and that the 8096 maintained no continuity with anything that came before it apparently didn't deter them. Most people saw this as a ridiculous marketing ploy intended to sell them a more expensive processor than they needed or wanted. It didn't work.

The basic 803 1 is still a viable candidate for new designs, and new-and-improved derivatives based on this fundamental architecture promise to safeguard your investments well into the future. Obviously, this is what it's all about. With the massive investment in development equipment, substantial firmware libraries, and an enormous working knowledge of the 8031 many engineers possess, it would be very difficult, indeed, to move away from any such "standard" architecture. This is especially true if the architecture is perfectly adequate for the task at hand. But for every 8031 user that is satisfied with the basic 8031 feature set, there's someone who needs a little more performance or additional

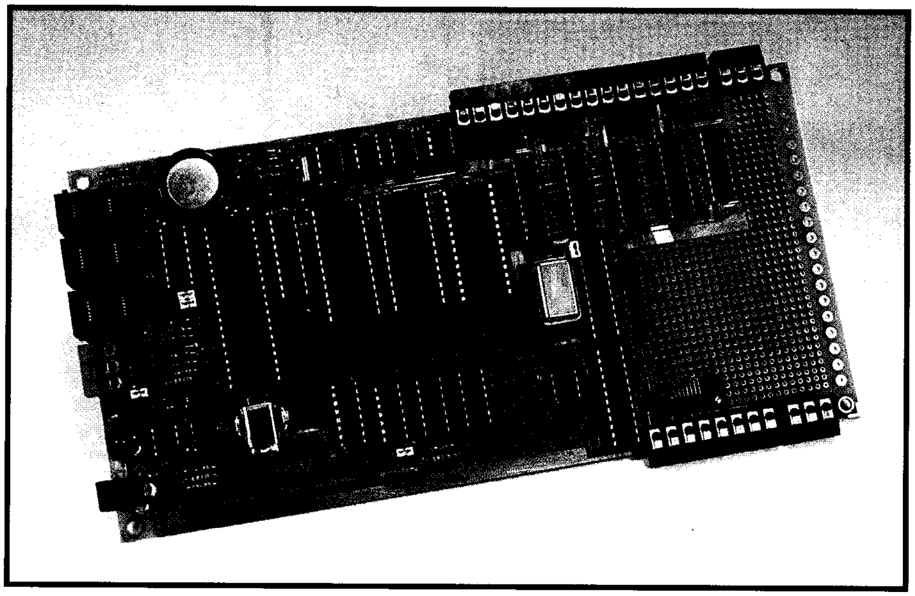


Photo I--The ec.32 *high-speed processor comes with an extensive set of PC-hostedsoffware tools that includes an 80C320 simulator and monitor/debugger, and 80C320 cross-assembler, and a C cross-compiler.*

functionality. These needs don't go unfulfilled for long since there's always some manufacturer willing to up the processing ante.

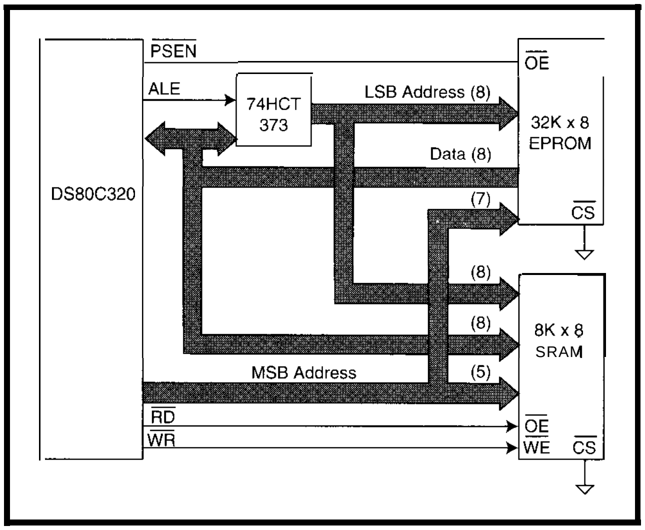


Figure 1—*In* a typical *80C320* setup where, for *simplicity's sake, the EPROM chip select is tied active, the EPROM access time is primarily determined by its address access* time parameter.

Truly impressive 803 1-based derivatives are being continually developed that provide specialized built-in peripheral sets that are well suited for a number of specific applications. As far as performance goes, most manufactures simply boost the basic clock rate, in some cases, beyond 40 MHz. Although this is one way of getting additional performance, it would make sense to first fix the execution core before throwing faster clock cycles at the problem. And if you don't think the 8031 is broken, how do you explain that it takes 12 oscillator clocks to choke out a machine cycle? And what about those instructions that contain clocks that do nothing?

This is where the story takes a bizarre twist. It turns out that the processor I'm about to tell you about, though based on the S-bit 803 1, attains a level of performance that allows it to give some of the 16-bit machines on the market a run for their money. And it shows that given enough raw processing power, you can overcome such seemingly insurmountable deficiencies as a meager instruction set.

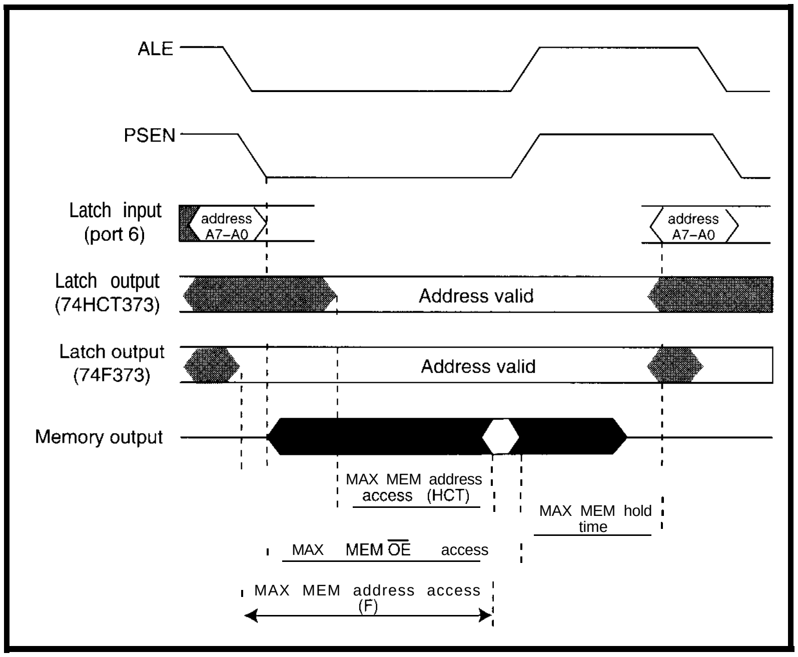


Figure 2-One *alternative that allows the use of lower-cost EPROMs is to speed up the discrete logic path by using an f-series part in p/ace of HCT. Jhe faster parts give fhe EPROM extra* time during a program *fetch cycle.*

GO FASTER

Dallas Semiconductor's new 80C320 addresses a number of performance-related issues plus adds a number of new features to the familiar 803 1. Actually, it's based on the 8032, which contains 256 bytes of internal RAM and a third timer. The new, high-speed 80C320 operates all the way from DC to 25 MHz. More importantly, the machine cycle, which is a processor's basic unit of timing, now consumes only four oscillator clocks instead of twelve as with a conventional 803 1. In addition, wasted cycles have been removed, streamlining certain instructions.

All instructions run faster, but some realize more of a performance gain than others. Typical applications should see a 2.5 times speed improvement using the same code and same crystal. Stepping it up a bit, at 25 MHz, a single-cycle instruction executes in as little as 160 ns. This is where architectural refinement pays off since it results in an apparent execution speed of 62.5 MHz, which works out to about 6 MIPS. Admittedly, these 6 MIPS are made up of tiny, little instructions, so take it as a relative, not an absolute, figure. Surprisingly, with the multitude of derivatives that had been introduced since the 8031's inception, it's taken this long to get any real architectural progress. If you've been avoiding using a language compiler on the 803 1 because of performance concerns, your case has just gotten a lot weaker.

Other new features include dual data pointers, a second full-duplex serial port, built-in power-on reset, watchdog timer, power-fail interrupt, and a total of 13 interrupt sources (six external).

IT'S ALL IN THE TIMING

By now, experienced engineers may be wondering what these performance gains are going to cost them. The cost I'm talking about goes beyond the price premium that the 80C320 exacts and involves system-wide concerns. Obviously, there are going to be timing-related questions since the 80C320 runs faster than a standard 803 1, even at a given crystal frequency. And to really tap its potential, many people will run it at maximum speed. Cranking up the clock rate implies tradeoffs involving logic families, memory and peripheral costs, and power consumption. When considering a new 80C320 design, one of the first things

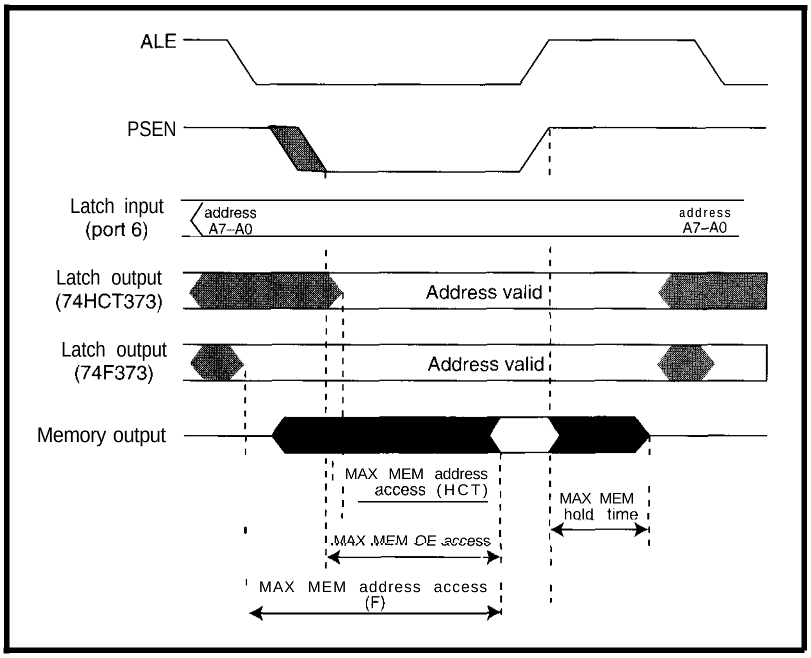


Figure 3—*Similar to Figure 2, the data memory read timing also benefits from using F-series logic in critical paths.*

you'll need to know is the boundary frequency at which faster-and more expensive-memory components are required, and the point at which a truly low-power design becomes difficult to attain.

Existing applications can be enhanced by using the 80C320 as a drop-in replacement for the 803 1. Since the instruction execution time is faster, there is less time available to transfer data to and from memory. Although it's true that, for a given clock speed, there is less time for memory access, the problem may turn out to be negligible for many systems running at 12 MHz or less. For example, a standard 8031 running at 12 MHz has an address access time (neglecting any delays in the support circuitry) of 300 ns. The corresponding figure for an 80C320 running at the same frequency is 230 ns. Take this with a grain of salt, however, since we really can't afford to neglect any delays in the support circuitry, as I'll demonstrate in short order.

Shifting our attention to the upper frequency extreme, things become more constrained. Since a memory's read timing usually proves to be more restrictive than its write timing, I'll look at an instruction fetch cycle first. For simplicity, let's consider a hypothetical bus configuration where the EPROM has its chip select pin permanently enabled (see Figure 1). In this arrangement, the access time is primarily determined by the EPROM's address access time parameter, t_{AA}. The limiting factor in this timing path is the time it takes the low-order address to propagate through the transparent address latch.

At 25 MHz, an instruction must be read into the processor within 100 ns from the time at which the address is emitted. This is dictated by the 80C320 timing parameter t_{AVIVI}, which is defined as $3t_{clcl}-20$, where t_{clcl} is the oscillator clock period. With just 100 ns available for the data transfer, you've got to take a hard look at the propagation delays in your support circuitry. Naturally, these delays are dependent on the logic family you choose. Using a CMOS 74HCT373 would introduce a maximum propagation delay of 45 ns. This amounts to a fairly substantial proportion of the overall time available for access and, consequently, requires the use of a fast 55-ns EPROM. Although you can get 55-ns EPROMs, they're not common, which means they're costly.

An alternate approach is to use a faster technology for the address latch, A safe bet would be a 74F373 with a worst-case propagation delay of 8 ns. This results in a address access requirement of 92 ns that is much easier to satisfy while staying on budget. Figure 2 is a simplified timing diagram of a program fetch cycle.

When beginning a new design, I always like to start by analyzing the timing relationship between the processor and all the memory and I/O components on the bus. With most conventional 8031 designs, this results in a somewhat academic activity since seldom do I encounter any surprises. To anyone considering using the 80C320, particularly if the system is to run at higher speeds, I would urge you to look to the 80C320 and all of the memory, peripheral, and logic timing in some detail. It's alarming to see how some engineers have grown complacent about timing issues just because they've been working with "safe" processors for a long time.

There are many different logic families available that are suitable for use in high-speed systems. Some are more conducive to low-power operation than others. Regardless of the choice, faster operation unquestionably means higher power consumption. There are tricks you can use to keep the power consumption in check, such as running the processor intermittently using idle or stop mode, for instance. The effectiveness of such tricks is very much dependent on the technology you select for your glue circuitry. Unfortunately, you compromise your potential power savings once you leave the domain of a full CMOS design.

An attempt to delimit where the performance/power boundaries fall indicates that an economical, low-power design can be rendered in CMOS if you keep the oscillator frequency under 16 MHz. This implies a relatively conventional implementation using a 120-ns EPROM. If you're willing to opt for a 90-ns EPROM, then the limit for full CMOS design falls at about 18 MHz. Once you exceed this range, you leave the realm of what can economically be attained using conventional HCT logic.

ACCOMMODATING TIMING

The 80C320 references its program storage at a fixed rate. This rate is based exclusively on the oscillator frequency and, as a result, imposes a fixed timing constraint on the system program memory. The only way to slow down the program memory accesses is to use a lower crystal frequency. Unlike program memory, when performing external data accesses, the 80C320 has a special feature that allows the application to control the access speed via a variable-speed MO V X instruction. The 80C320 can perform a MOV X in as little as two instruction cycles. This interval can be extended, if needed, all the way to nine cycles. Although it's true that fast RAM is easier to come by than fast PROM, the problem is that a lot of memory-mapped peripherals are not fast. Even if a fast system is required, the peripherals usually don't have to be fast, and fast RAM might not be necessary either unless the processor spends a significant portion of its time accessing data memory.

The processor can be instructed to stretch its read or write strobe by specifying a stretch value of between one and seven. The use of stretch cycles (or wait states), which widen the read or write strobe, gives the memory or peripheral more time to respond. These stretch cycles can be dynamically controlled by firmware if slower devices are being accessed. The stretch value is selected using the bits CKCON.2–0 in the Clock Control SFR (special function register) located at 8Eh. On power-up, the 80C320 defaults to a one stretch cycle for systems with slow data memories.

The timing requirements for external data memory are similar to those for the system PROM except that stretch cycles can be introduced to accommodate slower devices. For example, an HCT-based system running at 12 MHz would typically require a 170-ns RAM with zero stretch cycles, but could run with a 200-ns device with one stretch cycle. At 16 MHz, still in HCT CMOS, the corresponding figures would be 120 ns and 200 ns, respectively. A system using FAST logic operating at 25 MHz would require an 80-ns RAM with zero stretch cycles, but could get by with a 170-ns part with one stretch cycle. Figures 3 and 4 illustrate the basic data memory read and write timings.

At this point, some readers might be wondering what fringe effect these stretch cycles have on the access cycle. In other words, what happens to the setup and hold times? These are certainly valid concerns since you can extend the read or write strobes indefinitely, but if you are unable to meet the setup and hold times of your particular device, you're out of luck. And the setup and hold times get tight at 25 MHz, which could render a lot of your favorite peripheral chips useless; that is, unless you could buy yourself a little extra time.

Early 80C320 data sheets gave little clarification on these important timing aspects, but more recent documentation reveals that the introduction of stretch cycles does, in fact, adjust the setup and hold times as hoped. The term t_{MCS} represents the time interval added for each stretch cycle. Generally speaking, the value of t_{MCS} is increased by $4t_{clcl}$ (remember t_{clcl} is the clock period) for each additional stretch cycle that is called out. Note, however, that the first stretch cycle does not follow the expected pattern of adding four clocks to the strobe. This first stretch, in fact, uses one clock to create additional setup time and one clock to create additional hold time. The other two clocks are dropped into the middle of the read or write strobe. Subsequent stretch values have no further effect on the setup and hold times and instead are entirely used to extend the strobe.

Considering the particular timing parameters that are important in rendering a high-speed computing system economically, Dallas did their homework on this one. Conversely, it should be pretty apparent that the guys who have been pushing the clock rate of the original 803 1 core to inordinate extremes haven't done us any favors for anything other than basic single-chip systems.

Another timing-related area where the 80C320 accommodates the programmer is in the number of oscillator clocks used to advance the on-chip timers. Although the 80C320 is capable of using four clocks per timer tick, the default value is twelve, just like the 8031 and 8032. This allows you to calculate your baud rate divisors and other timer parameters in the conventional manner you're accustomed to. If you need higher timer resolutions, the speed of any of the timers can individually be adjusted higher. The control bits are contained within CKCON (the Clock Control Register) located at 8Eh. CKCON.5 controls the speed of timer 2, CKCON.4 controls timer 1, and CKCON.3 controls timer 0.

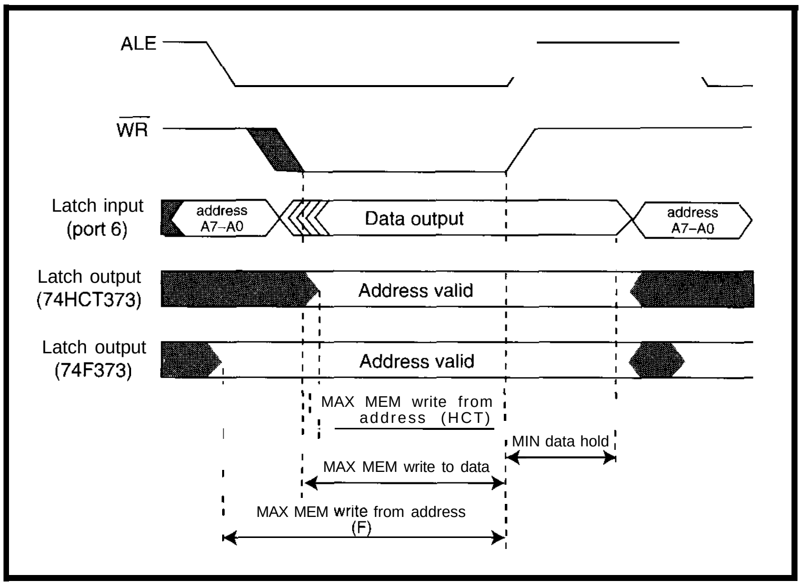


Figure 4-/n addition to *program fetch and data memory read cycles, the data memory write timing is simplified by using fasf discrete parts*

80C320 TEST FLIGHT

A processing engine as advanced as the 80C320 requires a test chassis equipped with matching capabilities. We've already established the superiority of the 80C320 architecture as a drop-in replacement for an 8031 in a given application. At the other extreme of the performance band, the 80C320 can be cranked all the way up to 25 MHz. But a fast processor buys you nothing unless you have a suitable peripheral set to do some useful work.

Since the 80C320 will, doubtless, find its way into applications such as process control and data acquisition, it follows that including the support functions necessary to satisfy these applications goes a long way toward showcasing the processor's capabilities in a realistic manner. To satisfy this need, Mid-Tech Computing Devices has embarked on a joint development effort with Dunfield Development Systems. The results are the ec.32 high-speed processor (Photo 1) and an extensive set of PC-hosted software tools that include an 80C320 simulator and monitor/debugger, an 80C320 cross-assembler, and C cross-compiler. Rest assured that I will complete my overview of the 80C320 and cover the ec.32's hardware and firmware in detail in upcoming columns, but since I'm almost out of space, let me leave you with an overview of the ec.32's principal features.

THE EC.32 HIGH-SPEED PROCESSOR

The ec.32 includes an 80C320 processor running at 25 MHz that supports the primary analog/digital peripherals directly on its high-speed bus. The digital I/O section includes 16 TTL inputs; 8 TTL outputs; and 8 high-voltage, high-current Darlington outputs. Analog I/O is supported with a 4-channel ADC and a 4-channel DAC. These are high-performance 8-bit devices that operate over a voltage span of O-2.5 V. The ADC performs a conversion in 3.6 us and has the capability to sample all four channels simultaneously. The DAC offers a 6-us settling time and can update all of the outputs simultaneously.

The system has 32K of EPROM, 32K of data RAM, and 32K of program RAM. All RAMs are backed up using a high-density capacitor power source. Possessing some of the desirable attributes of a battery, this "supercap" exhibits no wear-out mechanism and doesn't require any maintenance. The program RAM permits users to download executable programs from a host computer and allows the resident monitor to set breakpoints and to perform program modifications from the console. Access to the monitor is normally via the ec.32's secondary RS-232 port, leaving the primary serial port (RS-232 or RS-485) free for applications.

The secondary peripheral set is supported on an I^{2}C bus. Locally, this bus connects a real-time clock, 256 bytes of nonvolatile RAM, 5 12 bytes of E^{2}PROM, and a fully programmable interval timer that is set up for use as an interrupt source. The I^{2}C bus is carried through to a connector where external peripherals can be attached.

Finally, an efficient switch-mode power supply provides 5 V to the system and tolerates an input range of 8.5-28 VDC. Unlike a simple pass stage, this supply doesn't double as a heater. And the wide input range works well in a centrally powered configuration. ❑

John Dybowski is an engineer involved in the design and manufacture of hardware and software for industrial data collection and communications equipment. He may be reached at john.dybowski@circellar.com.

SOURCES

For elements of the project, contact

Mid-Tech Computing Devices
P.O. Box 218
Stafford Springs, CT 06075-02 18
(203) 684-2442

Individual chips are available from

Pure Unobtainium
13109 Old Creedmoor Rd.
Raleigh, NC 276 13
Phone/fax: (919) 676-4525

422 Very Useful
423 Moderately Useful
424 Not Useful

CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) 871-I 988-Four incoming lines
Internet E-mail: **sysop@circellar.com**

Reliability testing can be the bane of any electronic design. Getting the circuit to work is one thing, but keeping it working in the real world can be quite another. In our first thread this month, we look at some design techniques that can be applied to your next circuit to allow it to withstand transient voltage tests.

Next, we move over the world of telephones with a short discussion about how to do call progress monitoring. While the human ear can do it effortlessly, it's not always that easy to do it electronically.

Switching to the software side of things, converting between infix and postfix notation is something most any compiler must do, but knowing how to do it is a prerequisite to writing that compiler. We discuss some tips.

Finally, while Tom Cantrell talks about automatic vehicle identification in his "Silicon Update" column this month, what about having the computer take complete control of the car, allowing the driver to actually fake a nap? Our last discussion looks brief/y at what's been proposed for making such a system a reality.

High-voltage protection

Msg#:42055
From: GARY OLMSTEAD To: ALL USERS

I am developing a system that will (probably) be required to pass UL508 Standard for Industrial Control Equipment. I am concerned about passing the Transient Voltage Test described in paragraph 59. It says that a single 5-kV, 50-µs pulse will be applied.

I assume that it may, or will, be applied to any pin that comes to the outside world (although the standard doesn't say anything about this). I have protected the analog inputs with an MOV/resistor/zener combination, but I also have a 4–20-mA output and an RS-485 communications circuit. I can put an MOV across them, but the current-limiting resistor can't be used.

How do you protect these points?

Msg#:43400
From: GEORGE NOVACEK To: GARY OLMSTEAD

A piece of cake. Well, not really, but it's not as difficult once you have done it the first time and understand the mechanics. I am not familiar with UL508 specifications, so you'll have to let me know what it is. The "5 kV/50 µs" means nothing unless I know the impedance of the transient generator.

There is one word of caution, however. Most designers make the mistake of designing their circuit from the functional standpoint first and only then start worrying about EM1 and/or transient protection. You have to do it the other way around. You start with the protection circuits you will need (of course it is not a completely isolated process; you keep the functionality in mind) and only when you have it do you design the function. You may have to go back and forth a few times, but I can guarantee that if treated as an afterthought, no protection will work very well in tough environments.

Msg#:44650
From: DAN HOPPING To: GEORGE NOVACEK

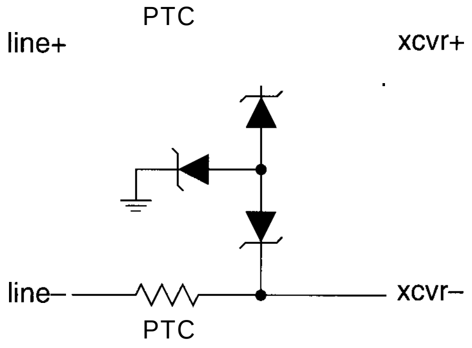
Well put, George.

I want to put a large "!" at the end of George's statement. I have been involved with medical equipment hardware design for years and there were major problems with *every* piece of equipment that left the component protection (and isolation) as an afterthought. On the other hand, when it was designed in at the beginning (i.e., the rest of the circuit is built up around the required protection scheme) the protection and isolation testing required at the end of the project *always* went VERY smoothly. Like the guy in the oil commercial says, "You can pay me now.. .or you can *PAY ME* later!" All you youngsters who claim to "want to learn" ought to tape George's statement to your design bench.

Msg#:44654
From: JOHN **HARTMAN** To: GARY OLMSTEAD

Well, we use 25-ohm PTCs as the series resistors on our RS-485 inputs, which are then clamped with back-to-back zeners as follows:

CONNECTIME



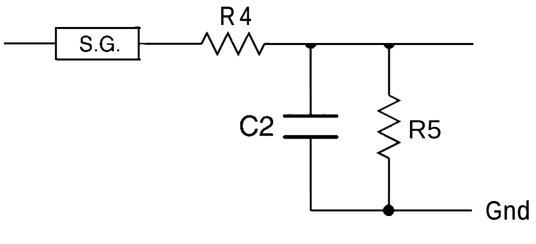
Msg#:44750
From: GARY OLMSTEAD To: JOHN **HARTMAN**

Yes, I asked Raychem about using PTCs in that application. They said their PTCs are only rated to 600 V and they won't survive 5 kV. Raychem specializes in the telecom market and only has to worry about UL1489.

Msg#:44749
From: GARY OLMSTEAD To: GEORGE NOVACEK

Well, it really doesn't matter too much which part of the circuit is designed first: the input just feeds an ADC, and the output is either a DAC feeding a 4–20-mA output or an RS-485 link. Either way, starting over isn't much of a problem.

Anyway, on to UL508. I'm really looking forward to being able to send graphics to a BBS; it would be a lot easier... (Say, maybe there's a product idea there. :-)) Here goes:



where:

- the stuff to the left of the input is a 120-V–to–5-kV converter. It reduces, more or less, to a 10-M resistor to ground in parallel with a 0.1 -µF cap.
- S.G. is a spark gap ("employing boiler electrodes," whatever they are.. .)
- R4 is 12 ohms, wirewound, wound to reduce reactance.
- C2 is 0.07 µF (yes, 0.07: 2 x 0.01 + 0.05).
- R5 is 350 ohms (300 watts!) I know you don't need to know that, but stage one of the conversion is a neon sign transformer. I've seen neon sign transformers, but never one that made anybody think of 300-W resistors. (Of course, maybe they just need it for the voltage rating.)
- The output pulse is described as a "single 1.2 by 50 microseconds fullwave impulse with a crest value of 5.0 kV."

Well, that's it.

Msg#:46339
From: GEORGE NOVACEK To: GARY OLMSTEAD

That is a fairly standard transient generator. R4C2 determines the rise time and R5C2 the fall time of the pulse. Also, R4 determines the maximum current you can zap your circuit with. In this case, with 12-ohm resistor, the maximum current will be roughly 417 A. Nothing to sneer at, but no big deal.

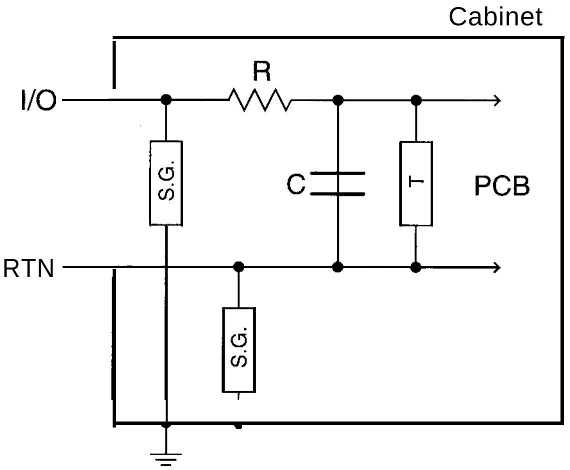
There are generally two types of tests: pin injection and bulk injection. In pin injection the generator is grounded, as is your enclosure, and then the generator discharges into each pin of your connector (or wire, if you have just fly leads). This is usually referred to as a *damage tolerance test,* since you want to make sure nothing gets fried.

In the bulk injection test the entire harness from your box passes through a current transformer. When the generator is discharged into the transformer, the entire harness swings, injecting common mode transient into your system. This is generally referred to as a *functional upset test.* You inject all kinds of pulses and single and multiple bursts, examining what the system does. You want to make sure that if the transients upset it, the upset is benign and you also want to know how the system recovers from it.

Testing for damage tolerance between two pins is highly unusual. If, for example, you have an RS-485 line, it is a twisted pair. You do not test it by sticking a 5-kV pulse between the two pins. In a lightning strike, for instance, both lines are close enough to be at the same potential. Sure, you can have a short on one line, but then you are getting into a higher level of protection. Now, you are not dealing with the environment, but with failures. You need to consider, however, what is connected to your equipment. You can have the best protection in the world just to be blown out of the water if the other guy did not do his job right.

Your simplest protection of the circuit is to have either a plastic enclosure or, if it is metal, then floating the electronics inside it. The test is performed by applying the transient between the ground-usually a large copper plate to which your equipment is fastened-and every connector pin or lead coming out of your box. If the electronics are floating, then all you have to worry about is the dielectric strength between the guts and the case. For the most part it is a question of spacing if you have a metal envelope.

You also want to make sure that the parasitic capacitance between the electronics and the enclosure will not allow high-voltage spikes to build on the lines. This is easily prevented by capacitors, which should be on the inputs for ESD protection anyway (you can buy ICs, such as RS-232 line drivers with the ESD already on chip). A typical line (input or output) would look like this:



Here, R would be a small resistor, say 25 ohms on an RS-485 line. On high-impedance inputs you can go a lot higher (the higher the better). C should be as large as possible, typically 0.1 µF on output lines. Again, you have to make sure you are not limiting your system bandwidth. The RTN (return) line should go through without a resistor if it is actually your common potential of the electronics. If not, such as with a differential pair {RS-485}, both lines should have the RC elements, tied internally to RTN.

I would also use spark gaps (SG) between each line and the cabinet. This will ensure that the potential between the cabinet and the electronics will never exceed the flash-over level. I use CP Clare's dual spark gaps. They fire at about 200 V and the smallest ones (about 0.5" dia x 1") will easily take a 5-kA surge. In addition, the dual gaps, when used on differential lines, will both fire at the same time, thus preventing differential voltage buildup which could damage the IC. They cost around $4 in quantities of 10. Unlike MOVs, they do not deteriorate with use.

You will always have a residual spike due to finite turn-on time of the surge suppressors. This is where the RC comes in. It is also a good idea to put a device between two differential lines, be it inputs or outputs, to limit the differential voltage between them. It is shown as T (for transzorb), but can be many other devices. In the isolated cabinet scheme you do not need to use these devices between single input or output lines and RTN unless you want to protect the line from overvoltage. The RC is already taking care of those. Also, keep in mind that in this scheme, any current due to transient injection flowing between your electronics and the chassis ground is through the spark gap (once fired, the voltage drop will be about 20 volts) and some parasitic capacitance between the line and the chassis. Consequently, the current will be a narrow, low-energy spike which any quarter-watt resistor in place of R will handle.

It is unusual to require that this (isolated cabinet) system take a hit or accept very high voltage between two lines as opposed to a line and the chassis. If this is a requirement, then you will have to protect the lines the same as if the case were on the same potential as the common potential of the electronics (internal ground).

To limit the voltage between two differential lines (device T above), you have many options. For communications lines, the best choice is usually a bipolar transzorb. It depends on the differential voltage you can allow and the maximum current which would flow through the clamp. For many current inputs where voltage difference would be minimal, two parallel signal or rectifier diodes limiting the voltage to 0.55 V will do. Or you can use back-to-back zeners, although I prefer bipolar transzorbs, as they are faster and take higher surge.

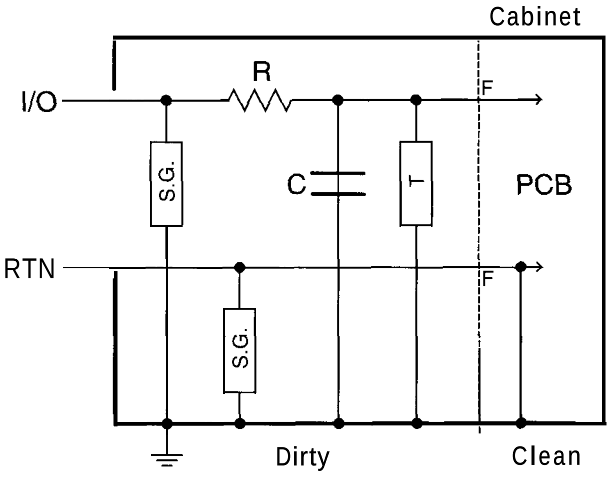
If the leakage is critical, using a reverse-biased P-N junction of a PNP or NPN transistor (small-signal transistors are best; leave the collector unconnected) gives you a beautiful "zener" with an extremely sharp knee zeners can only dream about at just few microamperes of current. Leakage below that knee is hardly worth considering. They will dissipate about the same as 500-mW zeners and their zener voltage will vary between about 5 and 10 V, depending on the device (voltage of a device is very consistent).

LEDs in forward bias also make excellent zeners. Their voltage will again depend on the type, starting from about 1 V to 3 V, with red LEDs at the low end, followed by green and yellow the highest. I used them once as a voltage reference when I had only 2 µA to work with and no commercial zener could do the job.

MOSFETs are also great. You connect drain and gate together to the line, source to the return. With many

MOSFETs having a gate threshold voltage around 3 V, you'll have a nice, often powerful (depends on the FET) clamp. And it will be bidirectional. The leakage will be determined by the I_{dsoff}.

I can't think of one commercial application where you would need to ground your electronics to the chassis (cabinet). You generally have no other choice when you are up against some onerous emission and susceptibility requirements, which make it necessary to run your interface with the world through feedthrough low-pass filters. Their unfortunate shortcoming is they are rated for only about 50 V. That means you have to cut the transients first, then go through the filter into your electronics. The filters are usually in a pi configuration, with capacitors grounded to the chassis. You could have an isolated metal box within another box, but this is expensive and heavy. The solution is in using a dual cavity, but then the electronics ground must be connected to the chassis ground. (If it was not, you'd be getting some nice signal feedback through these capacitors.)



As you can see, the RTN is now internally connected to the chassis. Spark gaps will take care of any voltage over 200 V and, once fired, clamp it to 20 V. In the second stage, the transzorb T with resistor R cut the spike to, say, 5 V on the RS-485 lines. On some differential lines you may want to use three transzorbs: each line to ground and then one between the lines.

Call progress detection

Msg#:44803
From: JAY SISSOM To: ALL USERS

I am looking for an easy way to monitor what is happening on a phone line. I need to know if the remote phone is ringing, if the line is busy, if they picked up the phone/hung up the phone, and so forth. My first thought was to purchase a cheap modem, but I need to send audio to the phone line and record from the phone line.

Does anyone know of a chip that does it all (and works with an approved DAA}?

Msg#:45165
From: BEN MEHLMAN To: JAY SISSOM

What you are talking about is called "Call Progress Detection" and there are chips out there that do it. Since I have no experience with any of them I won't attempt to say more on that score, except that you might want to check out Cermetek, Harris, Teltone, and Motorola among others.

Another possible option for you is a software-driven solution. You could couple the phone line (through some simple shaping circuitry) to the serial port on a computer (or to a pin on your microcontroller). But what I'm suggesting is not to attempt to measure the frequency as some have done, but rather to look at the timing of silence versus sound on the line. Each type of signal has its own distinctive rhythm. This is how it was done for call progress detection on the old Applecat modem. This really cuts your hardware and software down to a minimum.

Msg#:45455
From: JAY SISSOM To: BEN MEHLMAN

Thanks for the message. That's a good idea. I never thought about measuring silences. I'll have to play with it to see if I can make that work!

Msg#:45221
From: RICHARD NEWMAN To: JAY SISSOM

Dialogic makes boards for PCs that can do everything you want and more, however they are quite expensive: about $250 per port and usually have 2 or 4 ports.

Is this in your price range?

If you want to build one yourself, it's easy to do. You can get a DTMF transceiver in a single package with call progress detection included. You must validate the call progress with software to derive specific signals (busy, ringback, sit, voice, answer machine).

The DAA should have a duplexer so as to improve your transceiver characteristics. (Being able to hear the DTMF being sent to you while you are sending speech back to them.) The duplexer could be as simple as a transformer which would already be UL type approved.

Digital speech recording and playback is simple to do also. You need to make some decisions about fidelity versus cost and storage space. If you need great fidelity, you're probably better off with a DSP. You can do ADPCM encoding/decoding and reduce your bit rate considerably and then do compression to reduce storage requirements even more. All of this in one DSP.

If your going to be satisfied with average fidelity and you don't need to use a standard audio file format then using a CVSD type part is cheap and easy. You can record one second of speech in 1500 bytes of storage and the audio quality compares with a telephone line on a long distance call.

All of the software to drive this could be written in C, and one piece of software could drive multiple cards in one PC. It's probable that one card could have several interfaces on it also.

Your price range for a single-port card, if you built it yourself, would probably be about $40 including a PC prototype card.

Msg#:45456

From: JAY SISSOM To: RICHARD NEWMAN

Thanks for the message. Basically, I want to use a SoundBlaster board to do the recording and playing. I am writing the user interface in Windows and do not want to do low-level calls if I don't have to. There are multimedia drivers for the SB card, so I would like to use them. I want to build a board that I can control with the serial port to do the call monitoring. The $40 solution you were talking about sounds good. Thanks for your help!

Msg#:46086

From: BEN MEHLMAN To: JAY SISSOM

If you must use the SoundBlaster, you'll definitely need an external DAA which you can take on or off hook via a control line on the serial port or parallel port. You can detect call progress through silence detection and you can generate DTMF for dialing out fairly easily. The only thing that's not totally straightforward is detecting DTMF in software. But I question whether you are using the right tool for the job.

Someone mentioned Dialogic boards which do everything you want including the DAA. I have one and it's very nice, but expensive. There are cheaper boards if you only need to handle one line (the Dialogic does four). A friend has a $199 voice mail board (including software!) that works very nicely. I believe they sell an SDK for it for $150. There may be others that come with an SDK or, better yet, just a documented interface, or perhaps a TSR with a documented interface.

Infix to postfix conversion

Msg#:42727

From: SCOTT CHRISTENSEN To: ALL USERS

I am looking for source in C to convert from infix to postfix. I envision input and output are strings. For example:

In: 4 + 5 / (2 + 1)
Out: 4 5 + 2 1 + /

Thanks.

Msg#:43024

From: DAVE TWEED To: SCOTT CHRISTENSEN

Ah, that brings back memories of college and the compiler class. Every compiler has to do that conversion and it was a basic exercise that we all had to write.

I don't have source code for you, but I can tell you that you basically need to set up a stack to hold the operators as you scan them from the input string, and then a "precedence table" tells you when to take things from the stack and send them to the output string. The operands (numbers or variables) are always passed through.

Be careful: the example you give seems to use "calculator precedence," which is really no precedence at all-all operators are equally important. Another translation of your input string, using algebraic precedence in which multiplication and division are done before addition and subtraction, would be:

4 + 5 / (2 + 1) —> 4 5 2 1 + / +

This is part of the problem with infix notation-the precedence rules need to be established separately. With postfix (or prefix) notation, it is explicit, and you never need parentheses to make it clear.

Let's say you have a scanner that gives you one token (operand or operator) at a time. As you scan the input string left to right, you look at each token in turn and apply the following rules:

1. If the current token is an operand, append it to the output string.

CONNECTIME

2. If the current token is an operator, check the operator stack:

2a. If it is empty, put the operator on the stack.

2b. If the last operator on the stack has equal or higher precedence than the current token, pop operators from the stack and append them to the output string until you find one with lower precedence or the stack is empty. Put the current token on the stack.

3. If the current token is an open parenthesis, push it on the stack unconditionally.

4. If the current token is a closed parenthesis, pop operators from the stack and append them to the output until you find the open parenthesis, then throw away both parentheses.

5. If there are no more input tokens, pop any remaining operators from the stack and append them to the output.

Note that rules 3 and 4 can be mostly implemented by treating parentheses as operators and setting up the precedence table so that the open parenthesis is the highest possible precedence and the closed parenthesis is the lowest. You still need to make sure you throw them away rather than put them into the output string. The precedence table will look something like one of these:

Algebraic		Calculator	
Highest:	(	Highest:	(
	^		+ – * / ^
	/ *	Lowest:	)
Lowest:	)		

This should get you going. The C code is left as an exercise for the reader :-). I did mine in PDP-11 assembly language.

Msg#:43068
From: SCOTT CHRISTENSEN To: DAVE TWEED

Thanks, Dave. I started working on this yesterday using a data structures book that talks about this. The thing that may be a problem is multiple occurrences of operators in a row. For example, 6 */5 = ?. This would of course be illegal, but 6 • -5 = ? would be legal. Differentiating this is the fun part.

Msg#:43346
From: DAVE TWEED To: SCOTT CHRISTENSEN

Yes, I wondered whether that was going to be an issue for you. It's really two problems: The scanner should deal with literal numbers with leading sign digits, since the sign really is part of the number (like -123). The other problem is really the "unary operator" problem, in which unary "–" (as in "y + –x") is really a one-operand prefix-type operator, just like functions such as "sin x," "logy," and so forth.

The infix-to-postfix algorithm is really only for binary (two-operand) operators; it is assumed that a parser has already validated the string and dealt with the unary "–," usually by changing it into some other symbol analogous to "change sign" (which is how most calculators deal with the issue in the first place). So, your example would look something like this:

Input: 6 * – 5
After parsing: 6 • CHS 5
After postfix conversion: 6 5 CHS •

All of the unary prefix operators have a precedence between "(" and the binary operators; this causes them to be transferred to the output string at the right time, while still allowing things like:

sin (x + y)

to be turned into

x y + sin

Just to cover all of the bases, if you should have any postfix operators in the input string, like factorial ("!"), the infix-to-postfix conversion should simply copy them straight to the output when they occur in the input. So,

3 + IO!

would become

3 IO! +

and

(3 + 10)!

would become

3 10 + !

Msg#:43591
From: LEE STOLLER To: SCOTT CHRISTENSEN

Allen Holub has written a book, "Compiler Design in C." I haven't read it, but from other books he has written, it ought to be good. The man writes clearly with good examples. Try to see if you can borrow it from someone or somewhere. It ought to have just what you need.

CONNECTIME

Msg#:40108
From: SCOTT COLSON To: ALL USERS

A friend of mine has been asked to research Intelligent Highway Vehicle Systems (IHVS). She has no technical background and asked me for a quick explanation of how the system works. I have no idea how IHVS works so I was wondering if anyone here might be able to direct me to a source of information. I believe IHVS is used in those automated toll booths but that's all I know about it. I would appreciate any info.

Msg#:42093
From: GEORGE NOVACEK To: SCOTT COLSON

We looked into it some time ago. With the recession in the aerospace industry and the defense cuts, the high-tech companies were looking at what to do next. I'm not really cognizant of the subject as it exists today. Undoubtedly a lot of things happened since I looked into it. Basically, freeways and throughways would be set up in such a way that once the automobile gets on the ramp, a computer takes over the control of the car. The passengers can literally go to sleep until the car comes to the predestined exit, where the driver is alarmed and takes over from the computer.

Of course, two-way communications must exist between the car and the system. Inductive control with a cable buried in the road was considered as well as something similar to the cellular phone. Car spacing was to be either controlled centrally by splitting the road into small segments, as well as giving vehicles their own intelligence by installing ranging devices on them. On the top level, one had a computer system capable of routing thousands of vehicles for their destinations at shortest, fastest, and most efficient way.

To maintain the safety (this is where the aircraft companies could score), almost everything on the vehicles was to be at least dual redundant, fail safe, with detected faults forwarded through communications to the nearest service depot, so that spare parts and maintenance crews would be waiting by the time the vehicle arrived.

The big issue, of course, was software reliability (this is a misnomer, but is being used], with the minimum 10^{-10} probability of failure (bugs]. All of the system software was to be done in Ada according to avionic criticality level A. The main computer was to be at least triple redundant, with each of the three computers using a different processor with software written by different teams to prevent any possibility of a common mode error. As you can see, a bug could result in quite a carnage.

I don't know where the project stands now. Some of the first work was to be done in California and many aerospace companies were interested. A lot of interest dried up when it was indicated that this was not a government-funded project, but that each company would do their R&D for their nickel and hope to capitalize on it in the future from commercial success. This might work when the times are good. When companies struggle, there is little money left for pie-in-the-sky speculation. This is where I was pulled out of it.

Hope it helps.

Msg#:43335
From: BOB PADDOCK To: SCOTT COLSON

If this is the same as Intelligent Vehicle Highway System (IVHS) you might want to check out the June 1994 issue of Ward's *Auto* World for "IVHS: A smart way to go: Conference growth indicates blossoming industry" by Tim Keenan.

We invite you call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps. For information on obtaining article software through the Internet, send E-mail to info@circellar.com.

ARTICLE SOFTWARE

Software for the articles in this and past issues of *The Computer Applications Journal* may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only $12.

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (203) 8752199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

IRS

425 Very Useful 426 Moderately Useful 427 Not Useful

STEVE'S OWN INK

Time to Move On

I just spent the last hour and a half on the Circuit Cellar BBS answering electronic mail. Coincident with the physical posting of the magazine, I asked Ken to put last month's "Ciarcia Junk" editorial on the BBS as well. Barely a day later, I've already got a dozen requests. Everybody is requesting a specific project box, but ultimately they petition to receive virtually anything. The present requests range in scale from the earliest switching regulator project to the 64-processor Mandelbrot Generator, with no two appeals for the same project.

The fact that the first request was for the Mandelbrot Generator brought up the question of whether some projects were too expensive to give away. If I remember correctly, it cost close to $12,000 to build. Of course today, with $150 486-25 motherboards having equivalent power, the Mandelbrot box would sell by the pound. Obsolete hardware is just that, obsolete. OK, 60 Ibs into the UPS box.

Processing these requests has generated some vivid recollections. While the hardware in these projects is significant because it often marked a technological milestone, the fact that these projects were published so others could physically share the experience is what makes them truly significant. If having a circuit kludge or noteworthy manuscript page tacked to your computer room wall helps you remember that, so be it. I'm glad to help.

Ensuing technical events will probably be less dramatic, but no less important. We have witnessed a major evolution in computer systems. Among the half dozen or so technical revolutions brewing, be prepared for similar advances in home automation and building control technology. Of course, some people will be dragged kicking and screaming into this new world because our misinformed media usually depicts home automation as something to do with expensive entertainment and stereo systems. How wrong they are.

If I had said the words "energy management" instead, the whole world of political correctness opens up. Add functional embellishments like "security enforcement" and "lighting manager" and people want to know more.

The problem with political correctness in technical nomenclature is that such terminology beats around the bush. Functional interrelationships are lost. If you have not realized the energy benefit of your air conditioner turning on two hours before you get home rather than having it run all day because I call the $10 control module that accomplishes the task a home control device rather than an energy management facilitator, I apologize. Political correctness is not my bag.

I recently came across a good videotape that may help many fence sitters better understand an electronically enhanced home. Presented so that a novice can learn and a professional won't get bored, "Living With An Intelligent Home" is a good introduction describing current applications using today's technology. This VHS tape retails for $24.95 plus $5 shipping, but is being offered to *Computer Applications Journal* subscribers for $17.95 plus $4 shipping (in U.S.). Call, fax, or write us and be sure to include the subscriber number from your *Computer* Applications Journal mailing label.

Understanding new technology always benefits those who recognize and do something with it first. Next month I'll give you an opportunity to do both.

Steve