CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

DIGITAL SIGNAL PROCESSING

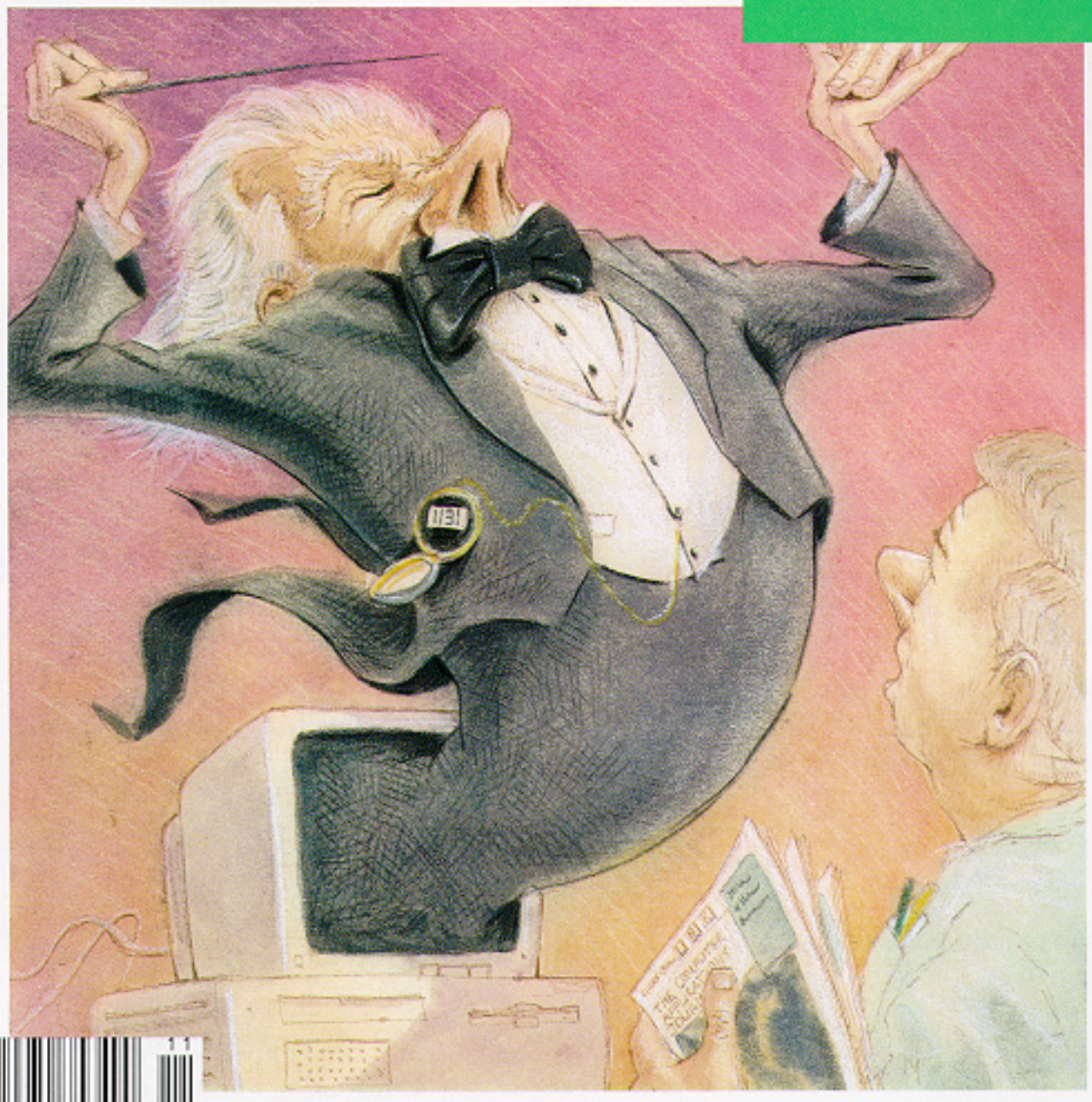
Digital Filtering Primer

Performing DSP Spectral Analysis

Clock-Calendar Chip Survey

Hot Chips VI

November 1994 — Issue #52



0 74470 75349 0 11

$3.95 U.S.
$4.95 Canada

EDITOR'S INK

Taken to the Extreme



Have you ever noticed how some people like to take things to extremes? I know I'm certainly guilty of it at times. The same is sometimes true when it comes to applying digital technology.

Digital electronics have revolutionized many, many aspects of the electronics industry. But, when people fall into a rut, they are often quick to overlook the obvious. For example, when I was reviewing BBS threads for this month's ConnecTime, I came across one in which someone was looking for a highly stable oscillator. The first response from someone suggested he do it digitally, and the discussion took off from there. Quite a ways down the list of replies, someone finally pointed out that a much simpler analog circuit could do the job just fine.

Similarly, we were taken to task by a reader who sent E-mail about an article we ran a few issues ago in which the author stated that a digital filter completely did away with the need for traditional analog filters. Luckily, this month's first article, which presents a primer on digital filtering, corrects the situation. It points out that any digital filter still needs a lowly analog filter on the front end to prevent aliasing when there is a noisy input signal.

This month's theme deals with digital signal processing, and many of the articles preach the gospel pretty thoroughly. However, don't be too quick to throw bits and clocks at a problem when a handful of resistors and op-amps may be just as effective.

Back to 1s and Os, though. Once you're up to speed on digital filters after poring over the first article, it's time to do some full-bore spectral analysis. Our second feature article looks at some of the issues to watch for when applying DSP to such an application.

Next, we look at a novel approach to DSP that attempts to get around some of the shortcomings of the venerable FFT. And, in our last feature, the authors explore some RISC/DSP coding tricks that might help you squeeze that last bit of performance out of a tight processing loop.

In our columns, Ed continues his journey through the protected land, Jeff checks out a huge array of real-time, clock-calendar chips available on the market, Tom gets hot and bothered by the sizzling new graphics and video silicon shown at Hot Chips VI, and John lights a fire under the old 8052 with a new board based on the DS80C520 speed demon,

Ken

CIRCUIT CELLAR INK®

THE COMPUTER 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

NEW PRODUCTS EDITOR
Harv 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) 675-2751. Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S. funds only, via International postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 696, Holmes, PA 19043-9613 or call (600) 269-6301.
POSTMASTER: Please send address changes to The Computer Applications Journal, Circulation Dept. P.O. Box 696, Holmes, PA 19043-9613

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & 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) 671-0549

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 sale and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK*.

Entire Contents copyright © 1994 by Circuit Cellar Incorporated. All rights resewed. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

14 **A Digital Filtering Primer**
by Tom Ulrich

20 **Spectral Analysis/FFTs and Beyond**
by David Prutchi

40 **Introduction to Doremi-DSP**
by Alan Land

50 **Fast-scaling Routine for Floating-point RISC and DSP Processors**
by Michael Smith & Chris Lau

54 ❑ **Firmware Furnace**
Journey to the Protected Land: Base Camp at 1 Megabyte
Ed Nisley

62 ❑ **From the Bench**
Does Anyone Have the Time?/A Comparison of Real-time Clocks
Jeff ***Bachiochi***

68 ❑ **Silicon Update**
Hot Chips VI/Image Compression, 3D, and RISC
Tom Can trell

76 ❑ **Embedded Techniques**
Heavy Duty Hammers/Beef up the 8052 with the DS87C520
John ***Dybowski***

INSIDE ISSUE 52

2 Editor's INK
Ken Davidson
Taken to the Extreme

6 Reader's INK
Letters to the Editor

8 New Product News
edited by Harv Weiner

ConnecTime 83
Excerpts from the Circuit Cellar BBS conducted by Ken Davidson

Steve's Own INK 96
Steve Ciarcia
A Majority Gains Control

Advertiser's Index 81

READER'S INK

Home Automation Information Void?

I agree with the need for home automation as you expressed it in *CAJ 50*. I have been working on it for a few years now. But, I must tell you that one control board does not a system make. I bought two TW523s and worked up software in C for the PC to control my house.

However, there is a big lack of information. I would like to extend the range of the X-10 RF receiver, but can't find the frequency. Being an extra class ham, one more antenna on the roof wouldn't be unsightly. X-10 offers no help at all. Running a ground plane roof antenna would considerably help me control the devices on my IO-acre farm. Think about *farm* control, not house control.

You could put some useful information in your magazine for us hackers-things like the frequency and pulse scheme for the X- 10 remote transceivers and receivers, specs on the infrared to X-10 remotes, tips from people who have solved some problems. For instance, there are people out there who need to know that you can jump the X-10 signal from one wiring side to the next using a 0.1 -µF, 600-V cap across two 1 10-volt phases. There may even be some who would actually pay for such a part in a metal box. Take me for instance, I also put my money where my mouth is. I own a small fortune in X-10 equipment and magazines.

Larry Dalton **(K9LD)**, Memphis, IN

You *are correct that X-10 can be stingy with the information they give out, but there certainly isn't a dearth of it. We've run articles in the past with full specs and schematics for the PL513,TW523, and the IR interface you mention* (CAJ 3, CAJ 5, *and* CAJ 9). *The TW523 data sheet is a gold mine of information about the module and the X-l 0 protocol itself.*

The old "capacitor across the phases" trick has been used for years, but is of questionable safety and only works moderately well. Leviton makes a signal bridge module that consists of a pair of tuned coils back to back that works much better. It is also U.L. listed.

For anyone who missed "Editor's INK" two issues ago, we ran an announcement for "Home Automation and Building Control," a new quarterly special section that will first appear in the January '95 issue of the Computer Applications Journal. *Keep an eye out for it as a prime source of this kind of information.*

One Happy Scavenger

After reading "Steve's Own INK" in *CAJ 48*, I wrote requesting the Term-Mite ST project, and then forgot about it. Much to my surprise, I received a little blue postcard acknowledging my request and notifying me that projects would be shipped soon.

I have to admit, I was a bit skeptical and thought perhaps it was a standard courtesy card sent to anybody requesting a project. When the project arrived, I could not believe I had actually received my first choice.

I understand how things accumulate over the years. I have this ever-increasing collection of manufacturer's data books as well as reference magazines and trade journals such as *Electric Design, EDN, ECN, Byte, Electronics Now, Dr. Dobb's Journal,* and of course, *CAJ*. It's too bad IC data books have to be so thick and that they are generally given away free. My bookcases overfloweth, but I can't bear to part with any books.

I too am a bit of a pack rat when it comes to electronic components. Even though there is little room left at the inn, I did manage to squeeze in my newly acquired project box. I was really happy to see the original prototype board as well as the software EPROMs that you sent me.

Thank you for letting me help you clean out the Circuit Cellar. I am very pleased and feel honored as one of the elite who actually received a project box which is, of course, a unique item in a finite series.

Nicholas Vasil, Bridgeport, CT

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) 87.52199.

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) 871-1988 with your modem (300–14.4k 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

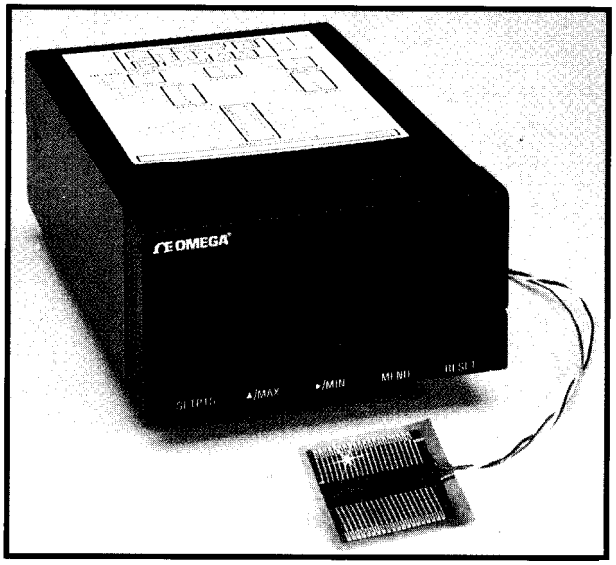
THIN-FILM HEAT-FLUX SENSOR

The **HFS-1** series from Omega is designed for precise measurement of heat loss or gain on any surface material over a temperature range from -201 to +204°C(–330–+400°F). The sensor can be mounted on flat or curved surfaces and employs a butt-bonded junction with a very low thermal profile for efficient reading.

The sensor is available with or without an integral thermocouple for discrete temperature measurement in two different sensitivity ranges. The carrier is a polyimide film which is bonded using a Teflon lamination process.

The sensor functions as a self-generating thermopile transducer with an output that can be read by any direct-reading DC-millivolt meter or recorder. A microvolt meter may be used to obtain maximum resolution.

Prices start at $99.



Omega Engineering
One Omega Dr. • P.O. Box 4047 • Stamford, CT 06907-0047
(203) 359-1660 • Fax: (203) 359-7700 **#500**

DSP DEVELOPMENT SYSTEM

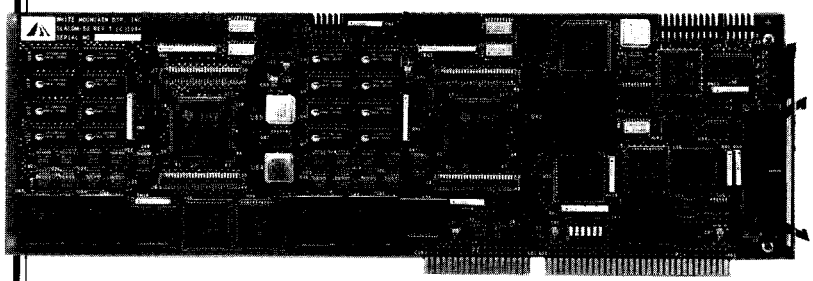
White Mountain DSP has announced the **Slalom-50,** a complete development system for the Texas Instruments TMS320C5*x* family of signal processors. The Slalom-50 incorporates two 57-MHz TMS320-C5 1 DSPs, a full complement of memory, plus daughterboard I/O capability. A TI C and assembly language source-code debugger is included and provides a fully integrated development system to expedite the generation, debugging, and optimization of 'C5*x*-based hardware and software.

The Slalom-50 architecture provides everything from a robust testbed to an end-use platform for C5*x* developers. The two DSP chips are used in a master/slave configuration. Full memory is provided for each DSP with 64 KB x 16 of zero-wait-state memory on each DSP's program and data bus.

A 4-KB x 16 dual-port SRAM provides a seamless data-exchange mechanism between the DSPs via the global-memory feature of the C5*x* family. In addition, the two DSPs are interconnected via the C5*x* TDM (time-division multiplex) bus, which also provides interboard communication. A serial controller chip is interfaced to the master C51 providing both asynchronous and synchronous serial data transmission. I/O can be accomplished via a daughterboard connection providing access to the full 64 KB of I/O space on each TMS320C5 1. Such access supports standard I/O access as well as booting and DMA.

The Slalom-50 can be used in four different ways. As a TMS320C5*x* single- or dual-processor prototyping platform, the Slalom-50 can prototype shared memory, TDM, and serial port interprocessor-communication schemes. The unit can also be used as a single- or multiple-TMS320C5*x* emulator, a software development and algorithm prototyping platform, and an OEM target board for embedded applications.

All systems come complete with a full-size dual-C51 PC/AT card, DOS and Windows versions of the TI C source debugger, Slalom User's Guide, Texas Instruments TMS320C5*x* User's Guide, and C Source Debugger User's Guide. The Slalom-50 sells for $3995.



White Mountain DSP
131 DW Highway, Ste. 433
Nashua, NH 03060-5245
(603) 883-2430
Fax: (603) 882-2655

#501

NEW PRODUCT NEWS

STEPPER MOTOR CONTROLLER

Semix introduces the RC-233 S-Curve Gener-**ate Master,** a stand-alone stepper motor controller featuring S-curve acceleration control for smooth acceleration. It also has I/O controls and an internal pulse generator, and can be operated in open- or closed-loop mode for accurate positioning.

S-curve acceleration and deceleration control has many advantages. It reduces vibration, eliminates the need for damping, and extends the mechanical system's life. It also enables higher frequencies to be reached because it needs less acceleration torque, and when used in servo motor control, it reduces registration time.

The RC-233 also has encoder-input capability, motor-control features, and an internal pulse generator so the user can achieve accurate motor control with inexpensive stepper or servo motors.

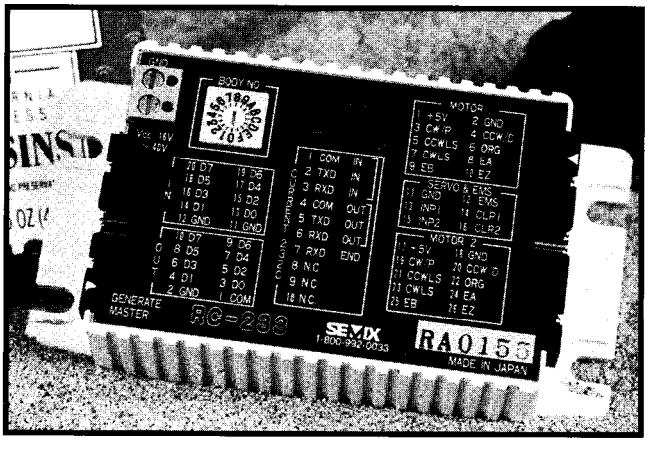
The controller is easily controlled with a personal computer or run as a stand-alone unit. Each controller controls up to two motors alternately, has 16-20 outputs, and high- or low-active configurable inputs.

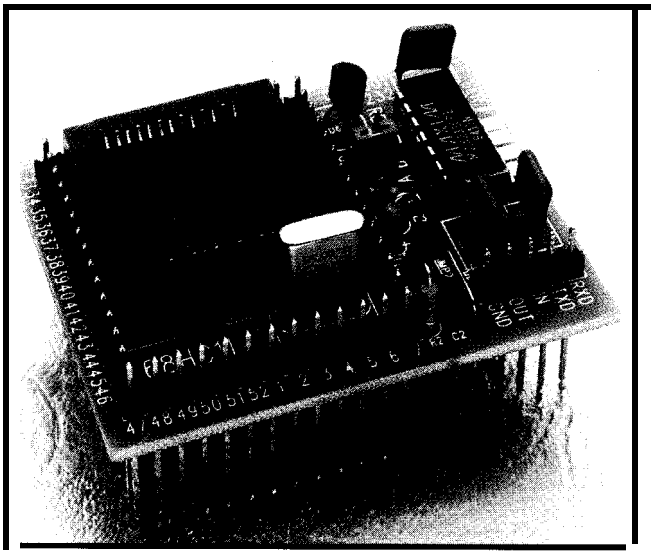
Additional high-performance features such as programmable speed and ramping as well as high-speed counting enable the RC-233 to be used with microstep drivers to achieve low vibration at low speeds.

The RC-233 measures 1.08" x 4.13" x 2.2" and is packaged in a rugged, EMI-shielded, heat- and dust-resistant case. This packaging makes it much more durable and noise resistant than traditional controllers. It can be combined with Semix drivers and stepper motors to make modular, distributed control systems.

Semix, Inc.
4160 Technology Dr.
Fremont, CA 94538
(510) 659-8800
Fax: (510) 659-8444

#502





WIRE-WRAP ACCESSORY

The **Model CGNlOOl** incorporates all the necessary components to begin construction on designs using Motorola's MC68HC11 microcontroller family. The CGNlOOl includes a 52-pin PLCC socket extended to 3-level-length wire-wrap pins on a 0.1" grid. Basic support circuitry for the controller includes a crystal oscillator, pull-up resistors on interrupt lines, reset circuit, mode-selecting jumpers, and power supply bypassing. The upper end of the wire-wrap pins serve as test points, making in-circuit testing and troubleshooting easier from the top side of the board. On this model, all 52 pins on the PLCC socket have a corresponding wire-wrap pin.

The CGNlOOl family is used like an intelligent socket. The developer saves several hours of preliminary construction by inserting the entire assembly into a 0.1" center perf board (as you would with any other wire-wrap socket), then moving on to other elements of the design.

The CGN1001-232 model includes a serial RS-232 level converter, which is built in to provide easy use of the hardware UART on the chip.

The units come fully assembled and prices start at approximately $20.

CGN Technology Innovators
1000 Chula Vista Terr.
Sunnyvale, CA 94086
(408) 720-I 814
Fax: (408) 720-I 814

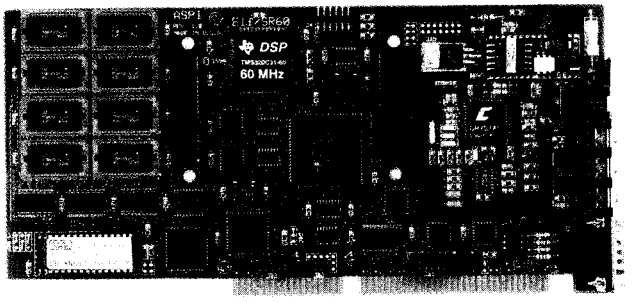
#503

NEW PRODUCT NEWS

LOW-COST, HIGH-PERFORMANCE DSP BOARD

Atlanta Signal Processors has introduced the **Elf/SR60 DSP Platform,** a floating-point DSP add-in card. Applications for the card include digital audio, speech recognition, voice mail, modems, facsimile, as well as image and speech compression and analysis.

Built around the 60-MHz Texas Instruments TMS320C3 1 floating-point DSP, the Elf/SR60 includes 256K words (1 MB) of zero-wait-state static RAM for maximum performance. Full-speed operation of the TMS320C31 equals a peak performance of 60 MFLOPS. The Elf/SR60 DSP Platform also includes a high-quality, stereo, 16-bit ADC and DAC and a telephone line interface.



The Elf/SR60 can be used in stand-alone operation. It features a UART on the AT bus for modem software compatibility and an onboard MIDI interface. The card is equipped with an EPROM monitor for loading, debugging, and running applications.

The Elf/SR60 has two daughterboard connectors that enable it to accept any two of the original Elf add-on boards including a coprocessor board, digital audio interface board, and a SCSI port board. Also available are a development environment (featuring a loader, assembler, C compiler, and C source debugger) and a DSP operating system and host interface software (which allows easy integration into host applications).

The Elf/SR60 DSP Platform sells for $1995 and development systems start at $3795.

Atlanta Signal Processors, Inc.
1375 Peachtree St. NE, Ste. 690 • Atlanta, GA 30309-3115 • (404) 892-7265 • Fax: (404) 892-2512 **#504**

FOR A GOOD REAL-TIME
CALL 800-753-4272

We've sold thousands of Transputer Education Kits for parallel computing, but would you believe the transputer is also terrific as a real-time co-processor for the PC? With its built-in multi-tasking process scheduler (with sub-microsecond task-switching), any number of processes can be made to automatically wake up at predetermined times or upon the sensing of external events. Programming time-outs is a breeze. And using the transputer's 20-megabaud bidirectional serial links (with on-chip DMA and much-easier-to-use-than-a-mm link adapters) you can connect to devices a hundred or more feet away. The Kit conies ready to use, including PC add-in card with a 20-MHz 32-bit T425 transputer, PC interface, and a meg of DRAM. You'll also receive C and Occam compilers and assembler, plus example and demo programs, manuals and schematics. Think about it. You really couldn't have a better time for just

$396

and with a 30-day money-back guarantee, no less!!

Computer System Architects
15 N. 100 E., #100
Provo, Utah 84606
800-753-4272
801-374-2300
Fax 801-374-2306
VISA • Mastercard
AmEx • Discover

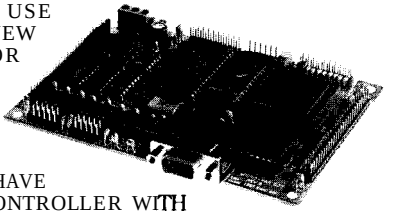


Requires IBM-compatible PC.

#104

SINGLE BOARD COMPUTER

THAT'S RIGHT! $129.95 FOR A FULL FEATURED SINGLE BOARD COMPUTER FROM THE COMPANY THAT'S BEEN BUILDING SBC'S SINCE 1985. THIS BOARD COMES READY TO USE FEATURING THE NEW 80535 PROCESSOR WHICH IS **8051 CODE COMPATIBLE.** ADD A KEYPAD AND AN LCD DISPLAY AND YOU HAVE A STAND ALONE CONTROLLER WITH ANALOG AND DIGITAL I/O. OTHER FEATURES INCLUDE:



- UP TO 24 PROGRAMMABLE DIGITAL I/O LINES
- 8 CHANNELS OF FAST 8/ 10 BIT A/D
- UP TO 4, 16 BIT TIMER/COUNTERS WITH PWM
- UP TO 3 RS232/485 SERIAL PORTS
- BACKLIT CAPABLE LCD INTERFACE
- OPTIONAL 20 KEY KEYPAD & INTERFACE
- 160K OF MEMORY SPACE, 64K INCLUDED
- 805 1 **ASSEMBLER & ROM MONITOR INCLUDED**

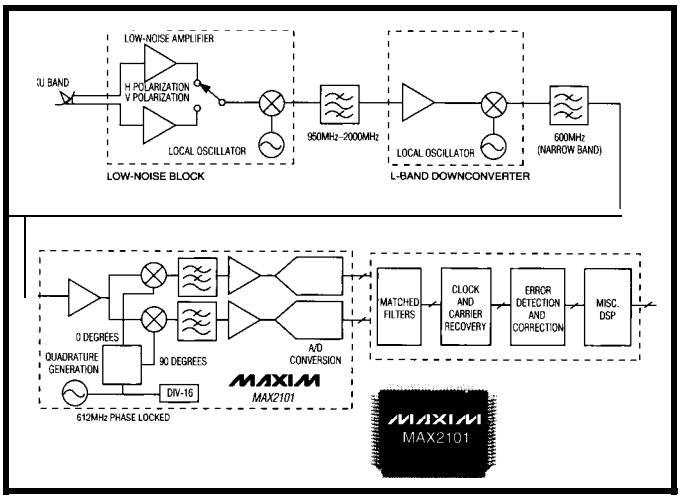
EMAC, inc.
618-529-4525 Fax 4570110 BBS 529-5708
P.O. BOX 2042. CARBONDALE, IL 62962

#105

NEW PRODUCT NEWS

6-BIT QUADRATURE DIGITIZER

Maxim has introduced the **MAX2101,** a 6-bit quadrature digitizer that combines quadrature demodulation with analog-to-digital conversion on a single bipolar silicon die. This unique RF-to-bits function bridges the gap between existing RF downconverters and CMOS DSPs.



The MAX2101 accepts input signals from 400 to 700 MHz and applies adjustable gain, providing up to 40 dB of dynamic range. It also features fully integrated low-pass filters with externally variable bandwidth (1030 MHz), a programmable counter for variable sample rates, and a signal-detection function.

Each baseband can be filtered by an on-chip, 5th-order Butterworth low-pass filter or an external filter. Baseband sample rate is 60 megasamples per second.

The MAX2101's simple receiver subsystem is designed for digital communications systems such as those used in Direct-Broadcast Satellite (DBS), Television Receive-Only (TVRO), and Wireless Local Area Networks (WLAN).

The MAX2101 is available in a 100-pin MQFP package and sells for $17.95 in quantity.

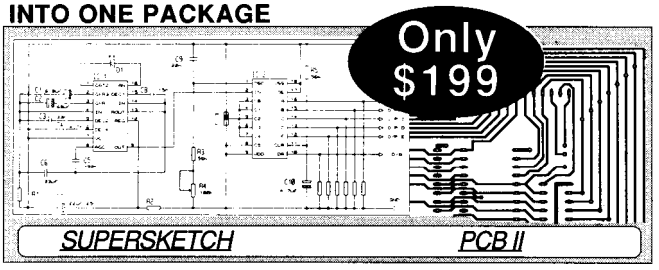
Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) **737-7600**
Fax: (408) 737-7194

#505

CADPAC II

TWO PROGRAMS FOR ONE LOW PRICE!!

SUPERSKETCH & PCB II: INTEGRATED **INTO ONE PACKAGE**



PCB II & SUPERSKETCH features:

- MOUSE DRIVEN *SUPPORTS CGA, EGA, VGA & SVGA,
- OUTPUT TO 9 & 24 PIN PRINTERS, HP LASERJET& HPGL PLOTTERS * OUTPUT TO DTP PACKAGES •
- PCB II ALSO HAS GERBER OUTPUT & VIEWING. •

THE EASIEST TO USE CAD AVAILABLE

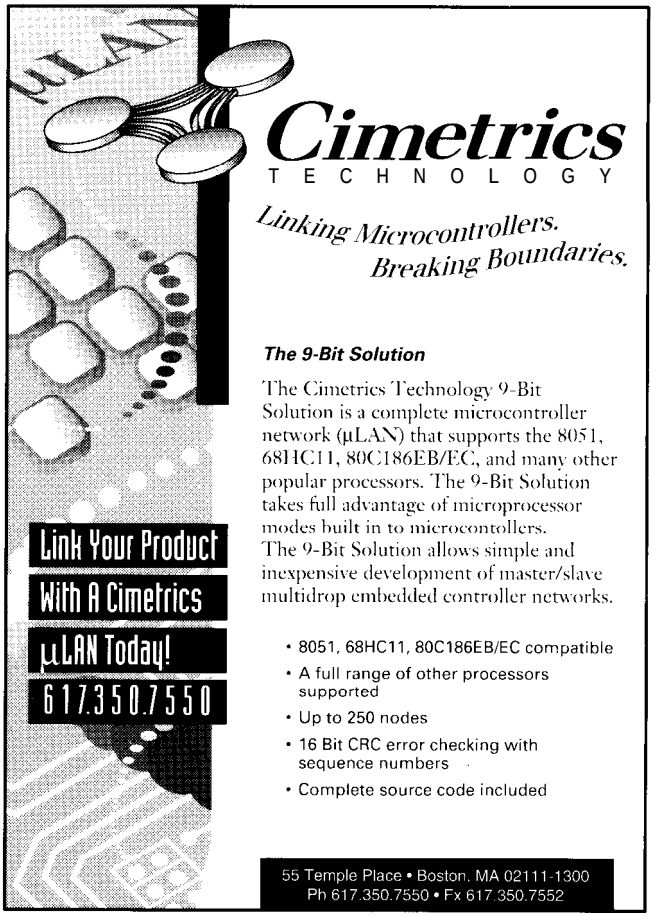
R4 SYSTEMS Inc.
1111 Davis Drive, Suite 30-332
Newmarket, Ontario **L3Y 7V1**
(905) 898-0665
fax (905) 898-0683

Free DEMO Package Write or Call Today

ALL PRICES ARE IN US FUNDS, PLEASE INCLUDE **$7 S/H**

Download DEMO from BBS at (905) 898-0508 (9600/8/N/1)

#106



Cimetrics
TECHNOLOGY

Linking Microcontrollers. Breaking Boundaries.

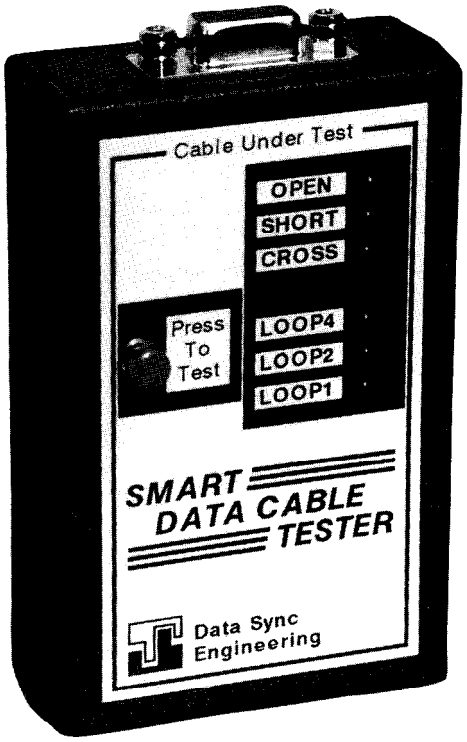
The 9-Bit Solution

The Cimetrics Technology 9-Bit Solution is a complete microcontroller network (µLAN) that supports the 8051, 68HC11, 80C186EB/EC, and many other popular processors. The 9-Bit Solution takes full advantage of microprocessor modes built in to microcontollers. The 9-Bit Solution allows simple and inexpensive development of master/slave multidrop embedded controller networks.

- 8051, 68HC11, 80C186EB/EC compatible
- A full range of other processors supported
- Up to 250 nodes
- 16 Bit CRC error checking with sequence numbers
- Complete source code included

55 Temple Place • Boston, MA 02111-1300
Ph 617.350.7550 • Fx 617.350.7552

NEW PRODUCT NEWS



SMART DATA CABLE TESTER

The Model DCT-1 is a pocket-sized, microprocessor-based cable tester designed to verify the pinout and integrity of new or installed cables having 2-9 conductors. Testing is performed by placing a "diode-configured" terminator at one end of the cable and the DCT-1 at the other. A unique program algorithm tests each conductor for continuity, shorts, and crossed connections. Results are displayed using red and green LEDs. A press-to-test button ensures battery and display operation with automatic power-off when there is no connection.

Useful features include Stop-On-Error, which detects intermittents by freezing the scan on a failed condition, and a Trace-Trap, which places a tone signal on the failed wire to help locate the faulty connection using headphones or a simple LED.

The unit is equipped with a DE9 connector and is supplied with terminators for any end-to-end combination of connections. The unit can be adapted to test coax, twisted-pair, flat-line cord, 10Base-T, Ethernet, Twinax, modular, or any other cable type.

The DCT-1 measures 2.4" x 3.8" x I", weighs less than 5 oz., and is powered from a 9-V alkaline battery. The Model DCT-1 sells for $99.

Data Sync Engineering
40 Trinity St. • Newton, NJ 07860
(201) 383-1355 • Fax: (201) 383-9382

#506

VIRTUAL METERING SYSTEM

Micron Meters has introduced an automatic serial port expander and selector box that provides four extra serial ports for use with any PC in connecting smart meters, controllers, counters, sensors, or transmitters. PortMUX is especially useful for data-acquisition systems using laptops and portable computers. Applications include test and measurement, quality-control-data recording, data communications, as well as multichannel data acquisition and display of virtual meters.

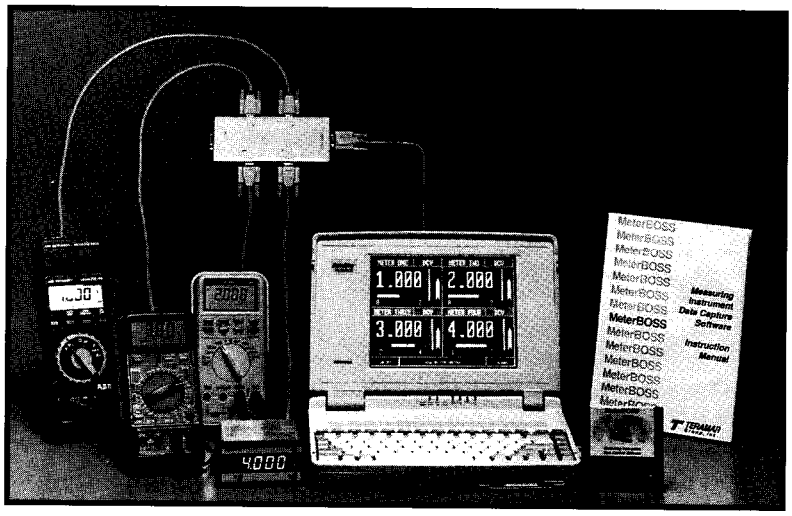
Housed in a compact plastic box 6.5" x 3" x 1", PortMUX has five DE9 connectors, a cable to the PC, and LED indication of ports in use. All ports are self-powered, and enabling software identifies the port each device is connected to. Special features include cascade connection, bidirectional communications, serial error-fault detection, and low-voltage (9 VAC) operation.

A fifth port can be used to connect to mother PortMUX for expansion purposes. Used with **MeterBOSS** software, the PortMUX becomes a field or laboratory data-acquisition system for multiples of four serial measuring devices. Four, eight, or sixteen channels of data can be displayed as virtual meters (including simple math functions such as sum, difference, product, or ratio) or bar graphs and reconfigured from the PC with a data-storage option.

PortMUX sells for $199.00 and the companion MeterBOSS software sells for $99.00 for a single site. Multiple meter versions are available from $249.00.

Micron Meters
4509 Runway St. • Simi Valley, CA 93063
(805) 522-0683 • Fax: (805) 522-I 568

#507

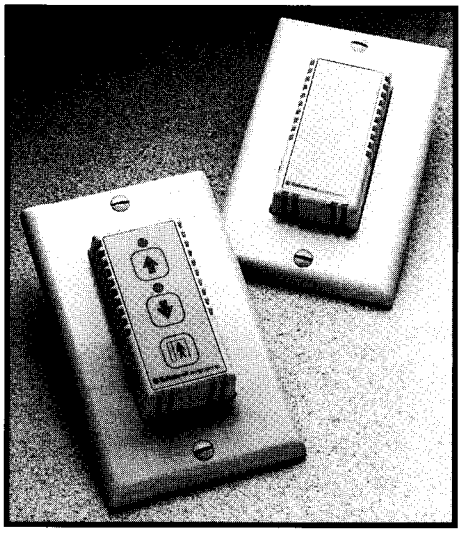


NEW PRODUCT NEWS

ROOM TEMPERATURE SENSOR

The TeleSys temperature modules, designed for use with the TeleSys line of terminal units and unitary controllers, measure ambient zone temperature. The sensors use a 10-kΩ type III thermistor.

The TeleSys sensor module features a unique design that fits into a standard wall switch plate which blends into a room's decor. The sensor comes on a mounting plate which screws directly to a standard, single-gang electrical box, and includes a thermally sealed design to ensure that it measures room temperature and not the air behind the wall.

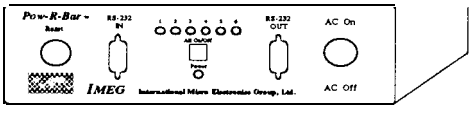


The sensor is available in two versions. One includes a membrane keypad which lets the room occupant adjust temperature setpoints and request after-hours occupancy. Both versions include a communications jack which offers communication to the TeleSys controller using a laptop or notebook computer. Through this, a technician can plug in at the sensor and communicate with a controller which is remotely located. A special RS-232 cable attaches the communications jack on the sensor to a 9-pin, RS-232 port on a computer.

The sensor operating range is from 35 to 125°F and features an accuracy of ±0.36°F. Two 6-position screw terminals on the back of the module accept 22-14 AWG wire.

Teletrol Systems, Inc.
Technology Center
324 Commercial St. • Manchester, NH 03101
(603) 645-6061 • Fax: (603) 645-6174

#508



Pow-R-Bar™ $149.00 (msrp)

is an intelligent, programmable, six outlet power strip which connects to a computer's serial port and operates via RS-232 protocol. Pow-R-Bar™ is the perfect solution for controlling multiple AC outlets.

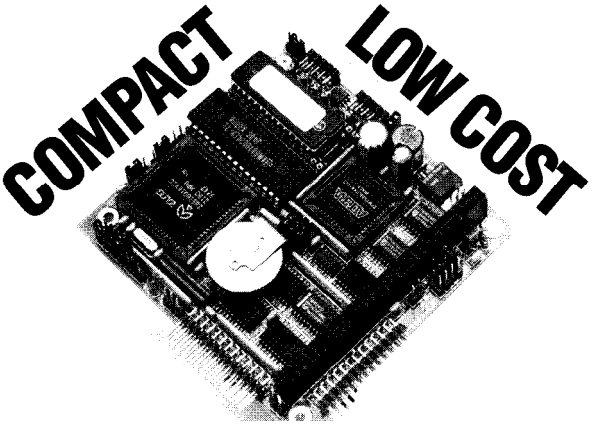
With Pow-R-Bar™ connected to a computer, each of the six AC outlets on the back of Pow-R-Bar™ can be turned on/off from the computer, by typing in a simple command or through custom programming.

Up to 26 Pow-R-Bar™s can be daisy chained together providing up to 156 outlets individually controllable from a single computer. With this system, an entire building can be automated.

IMEG International Micro Electronics Group, Ltd.

155 W. Tiverton Lexington, Kentucky 40503
P.O. Box 25007 Lexington, Kentucky 40524
800-274-8699 606-271-0017 Fax: 606-245-1798

#107



C-Programmable Controllers

Use our controller as the brains of your next control, test or data acquisition project. From **$149** qty one. Features I/O to 400 lines, **ADC, DAC,** RS232/RS485, printer port, battery-backed clock and **RAM,** keypads, LCDs, enclosures and more! Our simple, yet powerful, Dynamic C™ makes programming a snap!

24-Hr AutoFAX: 916.753.0618. Call from *your* FAX. Request catalog 18.

1724 Picasso
Davis, CA 95616
916.757.3737
916.753.5141 FAX

Z-WORLD ENGINEERING

#108

FEATURES

14 A Digital Filtering Primer

20 Spectral Analysis

40 Introduction to Doremi-DSP

50 Fast-scaling Routine for Floating-point RISC and DSP Processors

A Digital Filtering Primer

FEATURE ARTICLE

Tom Ulrich

You'll often find articles written about digital filters, how they work, and how to design them. But, how often do they approach the subject from a practical standpoint? Tom shares some design tips from the trenches.

A common maxim for designing embedded controllers is that a controller is no better than its feedback sensor. For example, if you are trying to control the position of something, a controller can do no better than its sensor's ability to measure a position. You can have the hottest microprocessor or DSP in the world, but if you can't accurately sense what you are trying to control, you will get poor results.

But, what if you are stuck using a sensor that is noisy or a few bits short of resolution? Is there anything you can do!

"Yes! "

The key is to use the processor to enhance the data before using it to control the data. And, the best part is there are simple techniques that enable you to do this even with a low-performance processor.

If you need this kind of information, I invite you to join me on a journey into the world of digital filtering. We'll take a look at how digital filters work, important details to remember when using digital filters, and implementation tips including sample code from real engineering projects.

THE BASICS

The most common digital filtering technique is to simply take a running average of several samples of data. The idea is that rather than just reading the transducer each time you close your control loop, you read it every time you take a piece of data and average it into the previous value using a weighting factor. In the process, the

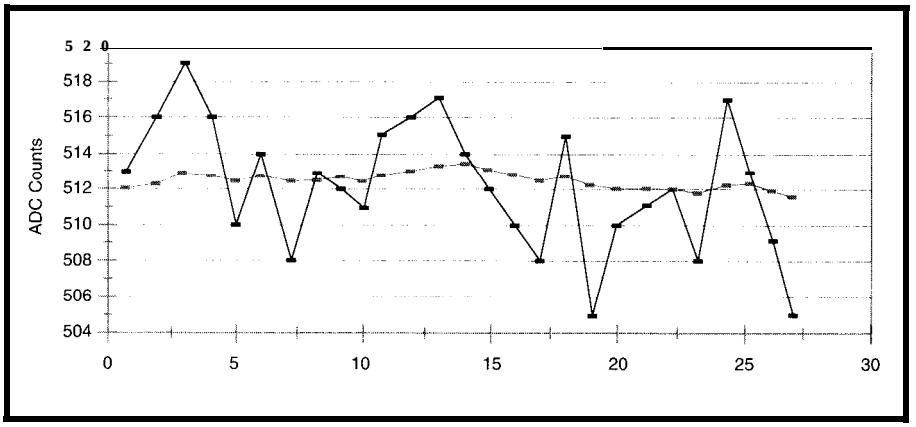


Figure 1 -*A simple digital filter produces the same result as a basic analog RC filter, eliminating high-frequency random noise.*

data becomes less noisy since random errors tend to cancel. Mathematically, this is expressed as:

$$X_{\text{filtered_new}} = K \times X_{\text{filtered old}} + (1 - K) \times X_{\text{raw}}$$

where $X_{filtered_new}$ is the latest filtered value, $X_{filtered_old}$, the previous filtered value, X_{raw}, the value just read from the sensor, and K, the filter constant (this always has a value between 0 and 1 in which 0 represents no filtering and 1 involves total filtering). To minimize the number of multiplication operations, this equation is usually implemented as:

$$X_{\text{filtered new}} = X_{\text{raw}} + K \times (X_{\text{filtered_old}} - X_{\text{raw}})$$

This technique gives a filter with much the same characteristics of a simple RC filter. Figure 1, which shows some raw "noisy" data read in from a sensor and the filtered result, illustrates the effect of this equation.

In looking at Figure 1, you may notice another interesting thing about digital filtering. The filtered signals are fractional values of the analog-to-digital converter's (ADC) codes, which means they are at a higher resolution than the nonfiltered signal. In fact, using digital filtering often gives you the equivalent of one or two additional bits on your ADC! This phenomenon occurs because the filter averages out the white noise on your system.

For example, suppose you have a voltage of 5.05 V on an ADC which was scaled from 0 to 10 V. If the signal was perfect, the ADC would always return a value of 129. However, if there was one bit (about 0.04 V) of white noise on the signal, it usually returns 129 with occasional values of 130 and 128. If the noise is truly white (a fairly good assumption), we would find that the occurrences of the other values would alter the filtered value to be 129.25, a resolution you normally need a 10-bit ADC to obtain.

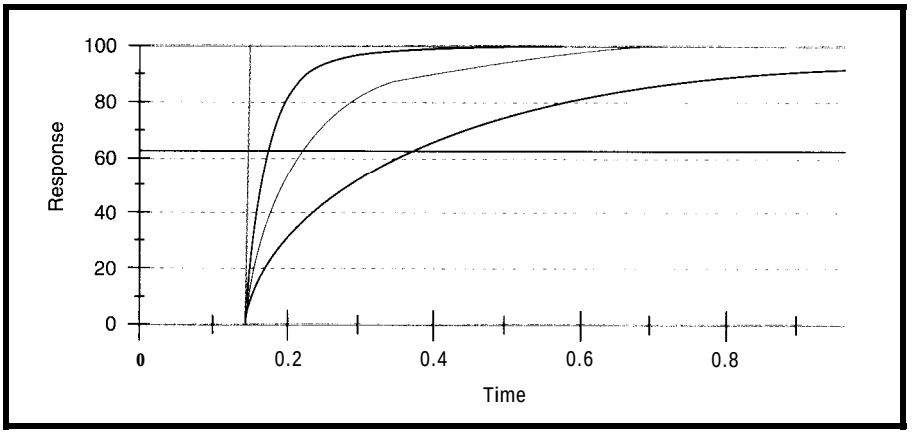


Figure 2—*Digitally filtering a step function with three different filter constants produces slightly different responses.*

To further illustrate this technique, Figure 2 shows the same filtering scheme with three different filter constants on a simple step function. Notice that on this graph, I have drawn a line showing the one-time-constant response ($1 - e^{-1} = 0.63$). Using a spreadsheet to model the filter with a step function is an easy way to determine the time constant. Table 1 shows the time constants corresponding to the three filter constants used in Figure 2.

Figure 3 shows the same filter constants applied to a simple sine wave. This graph clearly shows that a phase lag along with significant attenuation are introduced by digital filtering, as with any type of filtering. Table 2 shows the actual phase lags as determined again from a simple spreadsheet model of the filter and response.

IMPORTANT DETAILS TO REMEMBER

Now that we have looked at how a digital filter works, we need to look at some details that are important to know, but that textbooks usually forget to mention.

- The time constant needs tuning.

When you use a digital filter, the time constant becomes an additional item to tune. For example, if you use digital filtering to clean up data used in a PID servo loop, you will need to tune the filter constant as well as the P, I, and D gains. Furthermore, since the actual time constant of the filter is a function of both the filter constant K and the sample rate time, you need to consider filtering requirements as well as PID requirements.

Although it may appear from first impressions that digital filtering can be more trouble than it is worth, the bottom line is that sometimes you can't get adequate stability without it.

K	Sample Time	Time Constant
10	0.01	0.40 s
25	0.01	0.25 s
45	0.01	0.20 s

Table 1—*The time constant of the filter whose response is shown in Figure 2 decreases as the filter constant increases.*

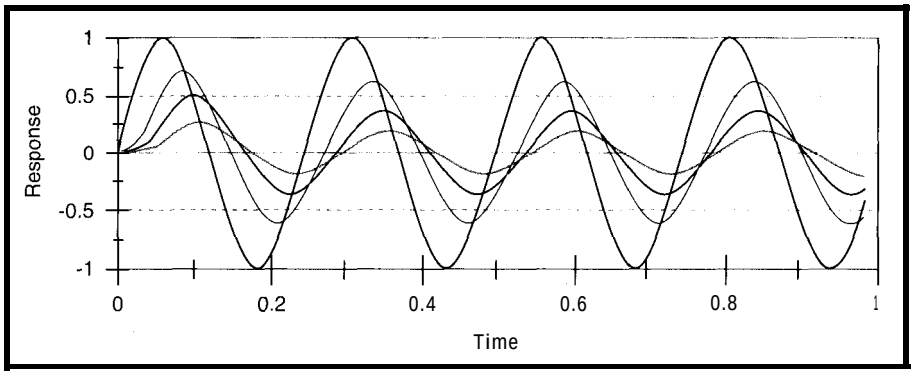


Figure 3—*Digitally* filtering a pure sine wave not only *attenuates the* signal *amplitude,* but also *shifts its phase (like* any ***analog*** *filter* would).

With it, you have to work hard to tune the system, but an acceptable solution is possible.

• Filtering introduces a lag into your control system.

Remembering the lag is especially important if the system dynamics require the use of lead terms such as derivative (or rate) gain. You must be careful not to nullify the advantage of lead terms by using too much filtering. There is a delicate balance that even the most sophisticated control engineers struggle with, but a balance between the two extremes does exist.

When writing the software for Parker's original electrohydrostatic actuator (EHA), I was able to filter both the position and velocity terms without killing the effect of the acceleration gain. In that case, the acceleration term was doubly filtered, but still able to contribute a significant leading effect. It was a difficult tuning task (and I had help from a controls guy), but without it we could not get adequate response from our controller.

• You still need an analog filter.

If you have a signal with noise at a frequency higher than the frequency at which you are sampling the data, you can get a phenomenon called ***aliasing.*** With aliasing, as you sample the higher-frequency data, you can end up reading "beat frequencies," which appear as lower-frequency signals.

For example, suppose we have an unshielded pressure sensor line that is picking up noise from fluorescent lights driven off a 60-Hz AC line. Let's further suppose that we are sampling data at 25 Hz. Here the problem stems from the fact that, at 25 Hz, we are not sampling the whole wave. The filtering is smoothing misrepresentative data points into a fictitious waveform.

The bottom line: anytime you use digital filtering, you must have an analog filter on your signal inputs to filter out higher frequency noise. So, for instance, on the Parker EMC100 digital-programmable motion controller, I used an analog RC antialiasing filter at 600 Hz for a signal that I sampled at 1000 Hz.

K	Sample Time	Phase Lag
10	0.01	72.0"
25	0.01	57.6"
45	0.01	43.2"

Table 2—*Increasing* ***the signal attenuation by decreasing the*** *filter constant also increases* ***the phase lag (as shown in Figure*** *3).*

A question sometimes raised at this point is, "If you always need a hardware filter when using a software filter, why not just forget the software filter and do it all in hardware?"

There are two reasons to not rely solely on hardware. First, implementing a high-frequency antialiasing filter requires only a small (and inexpensive) capacitor and resistor. But, to implement lower-frequency filters, you need much larger (and more expensive) capacitors. Hence, it is usually more cost effective to implement lower-frequency filters in software.

Second, frequently the selection of proper time constants for these filters is a matter of tuning. For different installations, you might want different time constants. With a software filter, adjusting a time constant is no more painful than adjusting a gain. But with a hardware filter, you've got to get out the soldering iron and change capacitors or resistors to make a time-constant change.

• Remember to initialize the filter

A common mistake in implementing a digital filter is failing to properly initialize the running average. Sometimes this mistake arises in the form of simply forgetting to initialize the average at all. Other times, it takes the form of initializing to zero.

The proper approach is to initialize the average to a value near the true value so the filter doesn't have to deal with what is, in effect, a big step function at powerup. A common way to initialize the average is to read the sensor one time at powerup. The

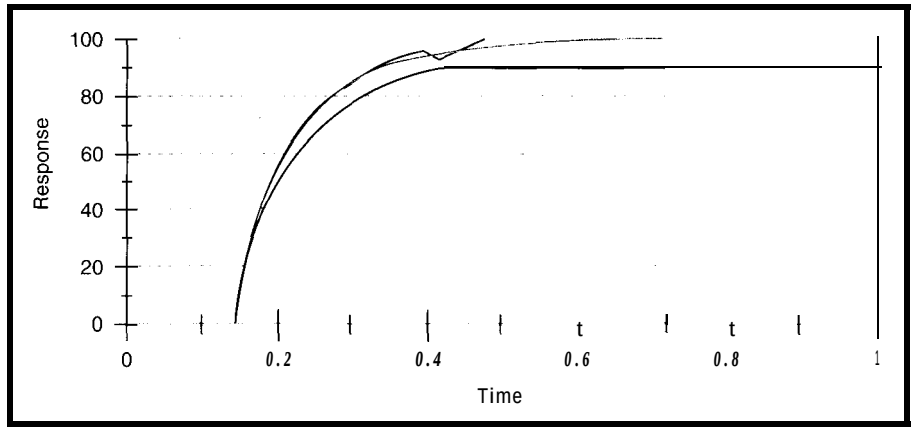


Figure 4—*Filtering* ***a step function using floating-point math in*** *the filter routine* ***produces a*** *nice,* ***smooth*** *output. Using integer math and keeping* ***frackofremainders*** ***results*** *in some bumps,* ***but*** *no DC offset.* ***Using*** integer math *but dropping the remainders produces a DC offset in the output.*

Listing 1—*Implementing a digital filter in* C **requires** *little* **code.**

```
void filter(int *filtered, int raw, unsigned int filtConstant
            unsigned int *low)
{
  long along

  /* convert to long to avoid overflow on multiply */
  along = (* iltered - raw);
  along = (along * filtConstant);

  /* add remainder from last time */
  along += *low;

  /* store remainder for next time through */
   *low = 0x0000ffff & along;

  /* shift right for fractional filter constant */
  along = along >> 16;
  *filtered = raw + along:
```

reading is then used as the value which initializes the average.

A purist may want to initialize the sum to the average of two or three readings, but that is usually not necessary unless your system is extremely noisy. The goal is to get the value nearly right to avoid an extreme response to a step function; the initial value doesn't have to be perfect, just close.

IMPLEMENTATION TIPS

Now that we have looked at how a digital filter works and some important details to remember, we need to look at some implementation tips.

• Don't use floating-point math.

Unless you have the very unusual situation of having an embedded controller with ample horsepower and resources, the last thing you want to do is use floating-point math with this equation. Instead, use fractional-integer math.

You want to represent a noninteger number as some fractional value of either 256 or 65,536. For instance, if you have a 16-bit controller, the natural way to represent the fraction ½ is with the number 32,768. To multiply, you multiply the number by the constant and shift it by 16 when you are all done.

For example, suppose the filter constant is 0.25, our running average is 2000, and the new value is 1900. *K* would equal 65,536 divided by 4 or 16,384. Hence, the equation is:

$$X_{\text{filtered}} = \{(16384 \times (2000 - 1900)) >> 16) + 1900 = 1925$$

Using fractional-integer math rather than floating-point math can easily reduce computation time by an order of magnitude.

• Use an integer and a remainder, rather than a long integer number.

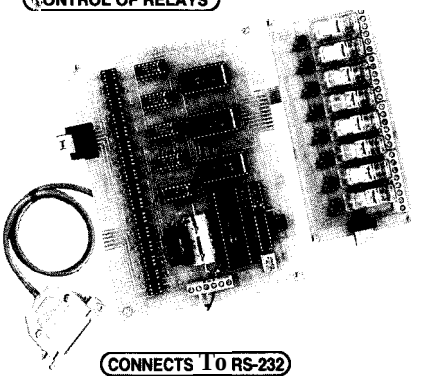
For a 12-bit ADC, you will probably find that using only 16 bits of filtered data will not give you enough resolution and will actually introduce truncation errors in your filtered value. But, if you opt for using a 32-bit word for your filtered data when running on a 16- or 8-bit microprocessor, you greatly increase the processing time needed to do the multiply.

The trick is to hold on to the remainder from the previous pass with the filter. (With a shift operation, the remainder is the part that gets shifted away when you divide by 256 or 65,536 as described above.) Each time you do the multiply, add the remainder from the previous pass and then store the new remainder.

Using a remainder, rather than a longer word length, also offers the advantage of using 16, not 32, bits for subsequent calculations when you use the filtered data in something like a

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

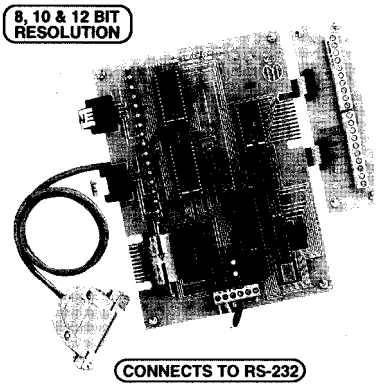


CONNECTS TO RS-232

AR-16 RELAY INTERFACE (16 channel) . . . . . $ 89.95
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards) A variety of relays cards and relays are stocked. Call for more info.
AR-2 RELAY INTERFACE (2 relays, IO amp)....$ 44.95
RD-8 REED RELAY CARD (8relays, 10 VA) . . . . . $ 49.95
RH-8 RELAY CARD (10 amp SPDT, 277 VAC)...$ 69.95

ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER* (16 channel/S bit)..$ 99.95
ADC-8G A/D CONVERTER* (8 channel/10 bit).$124.90
Input voltage, amperage. pressure. energy usage. joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4.000'). Call for info on other AD configurations and 12 bit converters (terminal block and cable sold separately).
ADC-8E TEMPERATURE INTERFACE' (8 ch)..$ 139.95
Includes term. block 8 temp. sensors (-40' to 146' F).
STA-8 DIGITAL INTERFACE'@ channel).....$ 99.95
Input on/off status of relays, switches, HVAC equipment. security devices, smoke detectors, and other devices.
STA-8D TOUCH TONE INTERFACE'.____..... $ 134.90
Allows callers to select control functions from any phone.
PS-4 PORT SELECTOR (4 channels RS-422)....$ 79.95
Converts an RS-232 port into 4 selectable RS-422 ports.
CO-485 (AS-232 to RS-422/RS-485 converter).....$ 44.95

● EXPANDABLE...expand your interlace to control and monitor up to 512 relays, up to 576 digital inputs, up to 128 analog inputs or up to 128 temperature inputs using the PS-4. EX-16, ST-32 & AD-16 expansion cards.

FULL TECHNICAL SUPPORT-provided over the telephone by our staff. Technical reference&disk Including test software & programming examples in Basic, C and assembly are provided with each order.

HIGH RELIABILITY...engineered for continuous 24 hour industrial applications with 10 years of proven performance in the energy management field.

CONNECTS TO RS-232, RS-422 or RS-485...use with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud).

Use our 800 number to order FREE INFORMATION PACKET. Technical information (614) 464.4470.

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

International & Domestic FAX (614) 464-9656
Use for information, technical support& orders.

ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 604
Columbus, Ohio 43215.5438

#109

Listing 2—*By* using a *bit of inline* assembler *(in this case for an 80C196 microcontroller) to* rep/ace some inefficient code generatedby the compiler, the *filter program runs* 70% faster.

```
register int diff;
register long prod;

void filter(int *filtered, int raw, unsigned int filtConstant,
            unsigned int *low)

   diff = (*filtered raw) ;          /* filter temperature */
   asm MUL prod, diff, filtConstant; /* prod=diff * filtConstant */
   prod += *low;              /* add in low word left from last time */
   *low = prod;                     /* store low word for next time */
   asm SHRAL prod, #15;
   asm ADD   prod, raw;
   *filtered = prod;
```

PID algorithm. This technique can easily reduce computation time by a factor of 2-3 times.

Figure 4 shows sample results of this technique. In this figure, you see three filtered results: a result using a floating-point (that's the perfect looking one), a result using a remainder technique (the one with a few bumps, but no DC offset), and a result using integer math without remainders (that's the one with the DC offset).

Listing 1 offers an example of a real implementation of such a filter. The program includes the original C code used to implement a digital filter on the Parker-Hannifin Vapor-Cycle-System Digital Controller for the AH64D Apache Longbow Helicopter.

Note how I take the difference between the filtered and new values, then place the result in a long real number called a 1 ong. This is important because otherwise the C compiler assumes I want an integer result for the subsequent multiply and chops off the high word.

• Use inline assembler when time is tight.

Listing 2 contains the code of Listing 1, except that it is rewritten for increased performance. I used some inline assembler to make the code smarter than that generated by the compiler.

The compiler implemented the multiplication by multiplying a long by a long, which means it did four 16-bit multiplication operations (MSW1 x MSW2, MSW1 x LSW2, LSW1 x MSW2, and LSW1 x LSW2) and then added the products together. In fact, all it needed to do was simply multiply LSW1 x LSW2 with no addition afterwards. By explicitly doing the multiply in assembler, I reduced the execution time of this module by 70%. Further time was saved using registers for some of the intermediate results and by using assembler again to do the shift and final assigns.

In summary, I have tried to present the basics of digital filtering along with some important implementation tricks. With these tools, you have all you need to solve your next noisy sensor problems.

Tom** Ulrich **received B.S. and M.S. degrees in engineering from the University of California at Irvine and is principal engineer in the Gull electronics system division of Parker Hannifin. He has written embedded software for numerous** new products **for both industrial and aerospace** divisions of Parker. He may **be reached at Parker Hannifin, 14300** Alton **Parkway, Irvine, CA 92718.

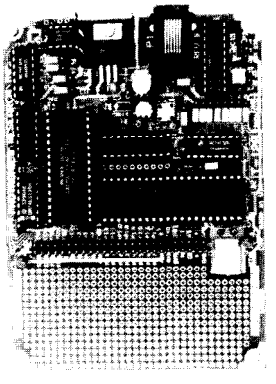
TURBO-128
THE *NEXT GENERATION* EMBEDDED CONTROLLER

**NO DEVELOPMENT TOOLS REQUIRED
READY TO PROGRAM IN BASIC OR ASSEMBLY

Photronics Research introduces the T-128: A True Single Board BASIC Development System. The T-128 is based on Dallas Semiconductor's new 8051-compatible DS80C320. With its 2X clock speed (25MHz) and 3X cycle efficiency, an instruction can execute in 160ns: an 8051 equivalent. speed of 62.5MHz!!! Equally impressive is the T-I 28's high-speed NVRAM interface. Any of the 128K RAM may be Program Development. has never been faster or more convenient, even with the finest EPROM emulator. The T-128 features PORT 0 bias and EA-select for DS87C520 upgrade.

PROCESSOR
•Dallas Semiconductor's DS80C320
•300% more efficient than the 8051
•Three 1 6-bit Timer/bunters
•13 Interrupts (6 Ext, 7 Int)
•A second 16-bit Data Pointer
•384 Bytes of Internal RAM
•Programmable Watchdog
•Brownout Protection
•Power-Fail Reset/Interrupt
•Power-On Reset
•Fully supported by *Franklin C51*

MEMORY
• Entire 128K Memory Map populated with *fast* NVRAM (64K DATA + 64K CODE)
•All memory programmed on-board
•Partitionable as CODE/DATA/OVERLAID
•Code Space is Write-Protectable
•State-of-the-Art Data Protection



Only 5" x 3-3/4" • 500-hole Proto Area • Console/Power connected by a single 4-conductor telephone wire (very convenient)

BASIC-520
•Modified BASIC-52 Interpreter (BASIC-5201 Now Fast Enough for New Applications
•Stack BASIC Pmgrams and Autorun
•CALL ASM Routines for Maximum Speed

I/O
•Three abii Parallel Ports
•Two Full-Duplex RS232 Serial Ports
•Decoded Device I/O Strobes
•50-Pin Bus Connector

UPGRADE
•DS87C520 processor (33MHz)
•Instruction cycle: 121ns 6.25 MIPS
•8051 equivalent: 82.5 MM
•Internal 16K ROM/1K SRAM

Comes Ready to Run with power adapter/cable assembly. Includes utility diskette with DETAILED TECHNICAL MANUAL
$199 in QTY.

109 Camille St. • Amite, LA 70422 • (504) 748-9911 • Tech Support (504) 748-7090 • FAX (504) 748-4242

#110

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

Spectral Analysis

FFTs and Beyond

Digital signal processors are often used for doing spectral analysis of incoming waveforms, but the old standby FFT can have its downside in the real world. Find out what to watch for in your next design.

FEATURE ARTICLE

David Prutchi

The analysis of a signal based on its frequency content is commonly referred to as *spectral* ***analysis.*** Although the mathematical basis for this operation, the Fourier Transform, has been known for many years, it was the introduction of the Fast Fourier Transform (FFT) algorithm which made spectral analysis a practical reality.

Implementing the FFT in personal computers and embedded DSP systems has offered an efficient and economical application of Fourier techniques to a wide variety of measurement and analysis tasks. Moreover, because the FFT has been found to be so valuable in applications such as medical signal processing, radar, and telecommunications, DSP chips are often designed to implement the FFT with the greatest efficiency.

In most instances, the powerful Fourier techniques, used in modems, fax machines, and CT or ultrasound scanners, are hidden from the user, who doesn't have to worry about their mathematical implications. In other cases, however, human interpreters must make diagnostic decisions based on frequency-domain representations of data processed through Fourier transforms.

For example, many digital storage oscilloscopes offer the user the option of converting time-domain signals into the frequency domain through the use of the FFT which runs on an embedded DSP and displays results directly on screen. It is also common for scientists and engineers to write short FFT-based routines to display a spectral representation of experimental data acquired by a personal computer. It is in these cases where the unwary may fall into one of the many traps that the FFTs conceal.

FFT users often forget that real-world signals are seldom periodic, free of noise and distortion, and that signal and noise statistics play an important role in their analysis. Because of these

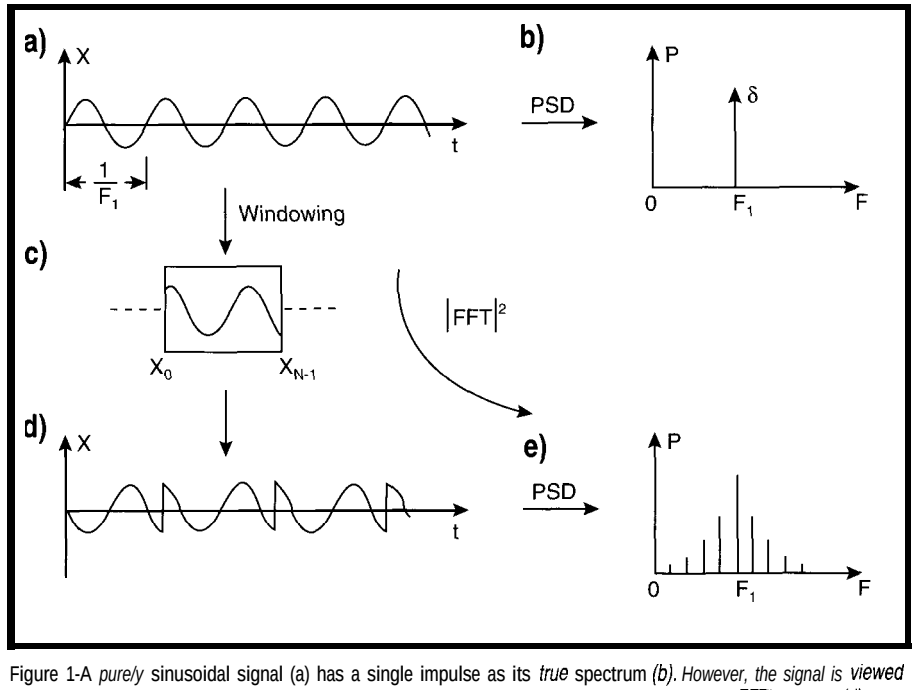


Figure 1-A *pure/y* sinusoidal signal (a) has a single impulse as its *true* spectrum *(b). However, the signal is viewed by FFT* through a finite window *(c),* and it is assumed that this record is repeated beyond the *FFT's window (d). This leads to leakage of the main lobe to sidelobes in the spectral estimate (e).*

"problem" factors, the FFTs and other methods can only provide estimates of the actual spectrum of signals. The results require competent interpretation by the user for correct analysis.

In this article, I will explain the common pitfalls in the use of the FFT and how to avoid them. After exposing some of the inherent problems which make the FFT unsuitable for high-resolution applications, I'll present more powerful spectral estimation methods, which cope with the fundamental shortcomings of the FFT, and describe typical applications for these methods.

FFTs AND THE POWER SPECTRAL DENSITY

Using a typical data acquisition setup, a signal is sampled at a fixed rate of

$$f_s = \left[\frac{\text{samples}}{\text{second}}\right]$$

which yields discrete data samples x_1, $x_1, \ldots, x_{N-1}$. These N samples are then equally spaced by the discrete sampling period $\Delta t[\text{seconds}] = 1/f_s$. The discrete Fourier transform (DFT) represents the time-domain data with N-spaced samples in the frequency domain $X_1, X_1, \ldots, X_{N-1}$ through:

$$X(f) = At \sum_{n=0}^{N-1} x_n e^{-2\pi j f n \Delta t} \qquad (1)$$

where the frequency $f[\text{Hz}]$ is defined over the interval $-1/2\Delta t \le f \le 1/2\Delta t$. The FFT efficiently evaluates this expression at a discrete set of N frequencies spaced equally by $\Delta f[\text{Hz}] = 1/N\Delta t$.

In its most simple form, the energy-spectral-density estimate of the time-domain data is given by the squared modulus of this data's FFT, and the power spectral density (PSD) estimate $P(f)$ at every discrete frequency f is obtained by dividing the latter by the time interval $N\Delta t$:

$$P(f_m) = \frac{|X_m|^2}{N\Delta t}; \; m = 0, 1, \ldots, N-1 \qquad (2)$$

where $f_m[\text{Hz}] = m\Delta f$. In a case which uses real data (this is the norm when sampling from real-world signals], the PSD for negative frequencies is symmetrical to the PSD for positive frequencies, making only half of the PSD useful. However at times, it may be necessary to compute PSD for complex data where relevant results are obtained for both positive and negative frequencies.

Although obtaining the PSD seems to be as simple as computing the FFT and obtaining the square modulus of the results, it must be noted that, because the data set employed to obtain the Fourier transform is a limited record of the actual data series, the PSD obtained is only an estimate of the true PSD. Moreover, as will be seen later, meaningless spectral estimates may be obtained by using Equation (2) without performing some kind of statistical averaging of the PSD.

Window	3 dB Bandwidth [bins]	Scallop Loss [dB]	Highest Sidelobe [dB]
Rectangular $w(n) = \begin{cases} 1 \text{ for } -\frac{N}{2} \le n \le \frac{N}{2} - 1 \\ 0 \text{ for all other} \end{cases}$	0.89	3.92	-13
Trianaular $w(n) = \begin{cases} 1 - \frac{\lvert 2n+1 \rvert}{N} \text{ for } -\frac{N}{2} \le n \le \frac{N}{2} - 1 \\ 0 \text{ for all other} \end{cases}$	1.28	1.82	-27
Hamming $w(n) = \begin{cases} 0.54 + 0.46\cos\left(\frac{2\pi(2n+1)}{2(N-1)}\right) \text{ for } -\frac{N}{2} \le n \le \frac{N}{2} - 1 \\ 0 \text{ for all other} \end{cases}$	1.30	1.78	-43
Hanning $w(n) = \begin{cases} 0.5 + 0.5\cos\left(\frac{2\pi(2n+1)}{2(N-1)}\right) \text{ for } -\frac{N}{2} \le n \le \frac{N}{2} + 1 \\ 0 \text{ for all other} \end{cases}$	1.44	1.42	-32

Table 1—*Typical window functions for use with the FFT in spectral estimation. Windows are N points long and are assumed here to be symmetric around n = 0.*

PITFALLS OF THE FFT

When sampling a continuous signal, information may be lost because no data is available between the sample points. As the sampling rate is increased, a larger portion of the information is made available. According to Nyquist's theorem, to correctly sample a waveform, the sampling rate must be at least twice that of the highest-frequency component of the waveform. Disregarding this rule will result in aliasing-a process in which signal components of frequency higher than half the sampling rate appear as components with a frequency equal to the difference between the actual frequency of the component and the sampling rate.

Because aliased components cannot be distinguished from real signals after sampling, aliasing is not just a minor source of error. It is therefore of extreme importance that antialiasing filters with very high roll-off be used for all serious spectral analysis.

Beyond appropriate sampling practices, the FFT still exposes other inherent traps which can potentially prevent analysis of a signal. The most important problems include leakage and the picket-fence effect.

Leakage is caused by the fact that the FFT works on a short portion of the signal, a phenomenon called *windowing,* because the FFT can only see the portion of the signal that falls within its sampling "window," after which it assumes that windowed data

repeats itself indefinitely. However, as shown in Figure 1, this assumption is only seldom correct. In most cases, the FFT analyzes a distorted version of the signal that contains discontinuities resulting from appending windowed data to their duplicates. In PSD, these discontinuities appear as a leakage of the energy's real frequency components into sidelobes which show up on either side of a peak.

The second problem, called the *picket-fence effect* or scalloping, is inherently related to the discrete nature of the DFT. That is, the DFT calculates the frequency content of a signal at very well-defined discrete points in the frequency domain rather than producing a continuous spectrum. In a perfect system, if a certain component of the signal had a frequency falling between the discrete frequencies computed by the DFT, this component would not appear in the estimated PSD.

To visualize this problem, suppose that an ideal signal is sampled at a rate of 2048 Hz and processed through a 256-point FFT. There would be a spectral channel every 4 Hz (at DC, 4 Hz, 8 Hz, 12 Hz, etc.). Suppose now that the signal being analyzed is a pure sinusoidal with a frequency of 10 Hz. In a perfect system, this signal would not appear in the PSD because it falls between two discrete frequency channels-much like the case of a picket fence which allows us to see detail in the scene behind it only if the details happen to fall within a slot between the boards.

In reality, however, because the FFT produces slightly overlapping "bins" of finite bandwidth, components with frequencies that fall between the theoretical discrete lines

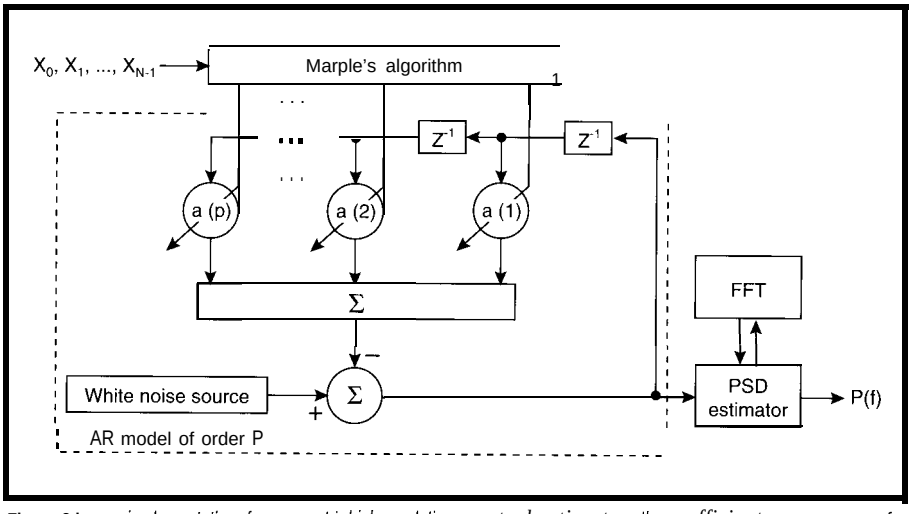
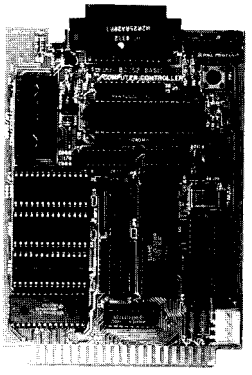


Figure 2-h one *implementation of a parametric high-resolution spectral estimator, the coefficients* $a_1, a_2, \ldots, a_p$ *of an AR filter are determined from input data through Marple's algorithm. The transfer function of the filter is then evaluated efficiently by FFT, resulting in a high-resolution estimate of the input data's PSD.*

BCC52
BASIC -52
COMPUTER/CONTROLLER



The BCC52 controller continues to be Micromint's best selling single-board computer. Its cost-effective architecture needs only a power supply and terminal to become a complete development system or single-board solution in an end-use system. The BCC52 is programmable in BASIC-52, (a *fast,* full floating point interpreted BASIC), or assembly language.

The BCC52 contains five RAM/ROM sockets, an "intelligent" 27641128 EPROM programmer, three b-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.

PROCESSOR
- 80C52 8-bit CMOS processor w/BASIC-52
- Three 16-bit counter/timers
- Six interrupts
- Much more!

INPUT/OUTPUT
- Console RS232–autobaud detect
- Line printer RS-232
- Three 8-bit parallel ports
- EXPANDABLE!

*Compatible with 12 BCC expansion boards

MEMORY
- 48K RAM/ROM, expandable
- Five an-board memory sockets
- Either 8K or 16K EPROM

TO ORDER CALL 1-800-635-3355
TEL: (203) 871-6170
FAX: (203) 872-2204

BCC52	Controller board with BASIC-52 and 8K RAM	$189.00 Single Qty
BCC52C	Low-power CMOS version of the BCC52	**$199.00**
BCC52I	-40°C to +85°C industrial temperature version	**$294.00**
BCC52CX	Low-power CMOS, expanded BCC52 w/32K RAM	$259.00

CALL FOR OEM PRICING

MICROMINT, INC. 4 Park Street, Vernon, CT 06066
In Europe (44) 0285-658122 • In Canada: (514) 336-9426 • In Australia (3) 467.7194
Distributor Inquiries Welcome!

#119

CONCEPT TO MARKET

Professional Computer Services
Specializing in
System Design and Software

- 8051 and family
- PIC16C5X and PIC16C71/84
- 386 + /Windows 3.1
- C, C++, BASIC, ASM
- Real-Time Embedded Control
- Data Acquisition, Automation
- Communications, etc.

Satisfaction Guaranteed
MYRIAD DEVELOPMENT Co.
9220 West Tennessee Ave.
Lakewood, CO 80226
(303) 692-3836

are distributed among adjacent bins, but at reduced magnitudes. This attenuation is the actual picket-fence or scalloping error. Both of these problems are somewhat corrected by the use of an appropriate window.

So far, all samples presented to the FFT have been considered equal, which means that a weight of one has been implicitly applied to all samples. The samples outside of the FFT's scope are not considered, and thus their effective weight is zero, resulting in a rectangular-shaped window. This ultimately leads to the discontinuities that cause leakage.

A number of windows have been devised which reduce the amplitude of the samples at the edges of the window while increasing the relevance of samples towards its center. By doing so, these windows reduce the discontinuity to zero, thus lowering the amplitude of the sidelobes that surround a peak in the PSD. In addition, the use of a nonrectangular window increases the bandwidth of each bin, which results in a decreased scalloping error.

Some typical window functions and their characteristics are presented in Table 1. In essence, these functions produce N weights $w_0, w_1, , w_{N-1}$ which are "weighted" (multiplied) one-to-one with their corresponding data samples $x_0, x_1, , x_{N-1}$ before subjecting them to the FFT:

$$X(f) = At \sum_{n=0}^{N-1} w_n x_n e^{-2\pi j f n \Delta t} \qquad (3)$$

Reduced resolution is the price paid for a reduction in leakage and scalloping through the use of a nonrectangular window. In fact, if it is necessary to view two closely spaced peaks, the rectangular window's narrow main lobe lets the user obtain analysis results, which report the existence of these closely spaced components. Any of the other windows would end up fusing these two peaks into a single smooth crest.

The use of a rectangular window is also appropriate for the analysis of transients. In these cases, a zero signal usually precedes and succeeds the transient. Thus, if the FFT is forced to look at the complete data record for the transient, no artificial discontinuities are introduced, and full resolution can be obtained without leakage.

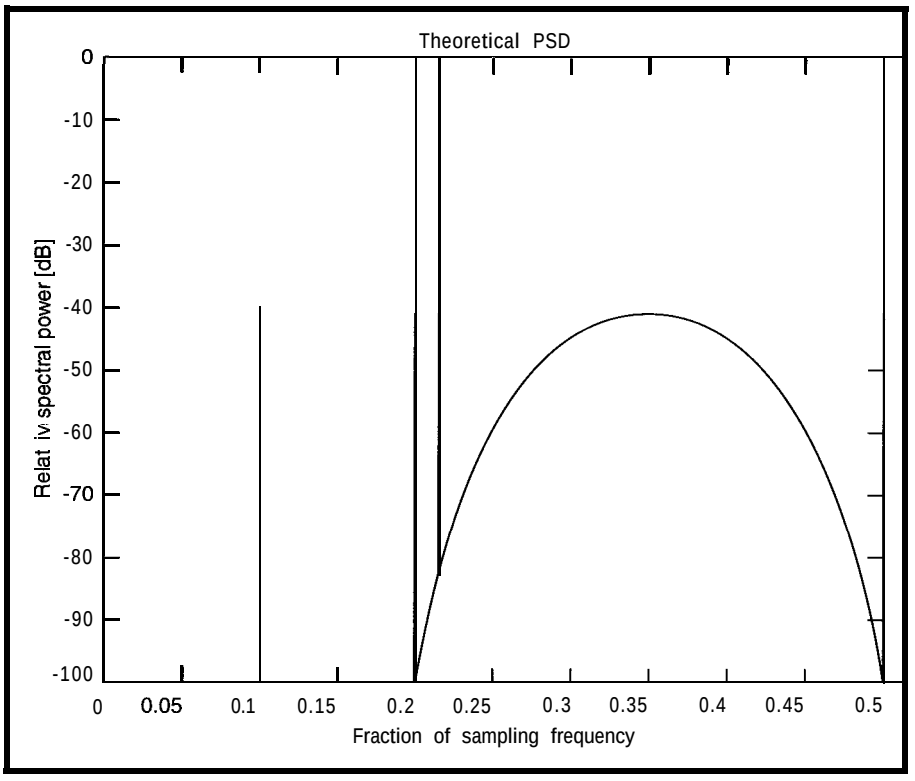


Figure **3a**—*Positive* frequency theoretical *spectrum* of *Marple's 64-point* complex data test set. This *spectrum* includes features *that* are *well* suited for evaluating spectral estimators.

As you see, there is no single window which outperforms all others in every respect, and it is safe to say

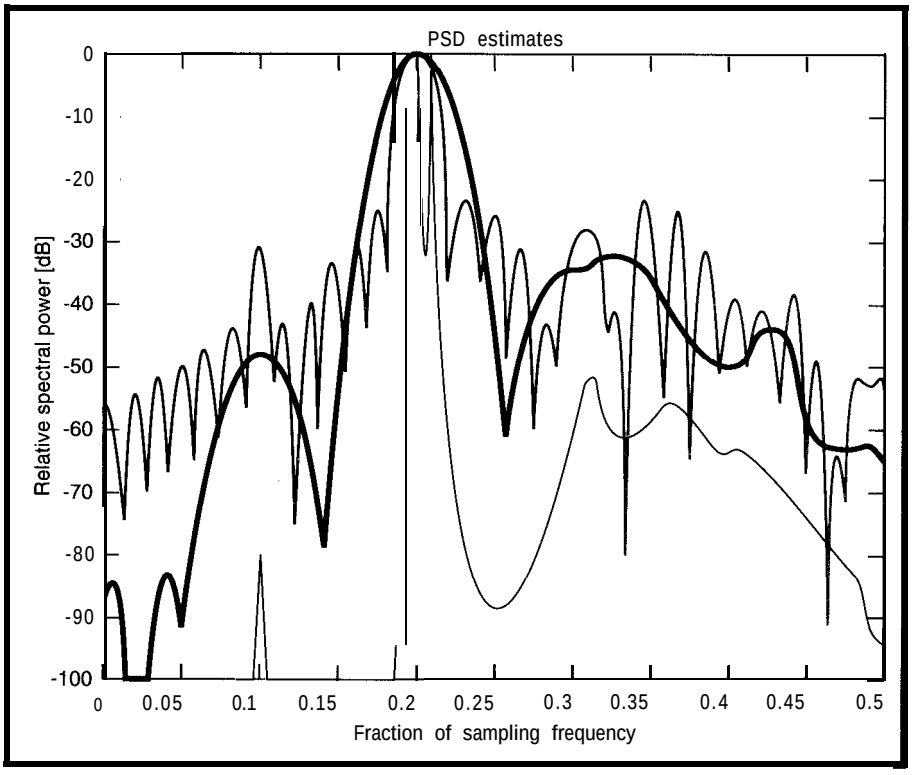


Figure **3b**—*Unlike the* very well-established features of *the theoretical* spectrum shown in Figure *3a,* spectral estimates distort *the PSD according to their inherent assumptions. The spectrum is* estimated using *three* different *methods: i) zero-padded FFT periodogram (green line), ii) Welch's method (black line, N = 32, S = 16), and iii) Marple's method (red line, p = 15).*

that selecting the appropriate window for a specific application is more of an art than an exact science.

Another solution comes in handy when the signal rides on a relatively high DC level or on a strong sinusoidal signal. In these cases, it is advisable to remove these components from the data before the PSD is estimated. Without taking this precaution, the biasing and strong sidelobes produced could easily obscure weaker components. Whenever physically expected, the DC component of a signal can usually be removed by subtracting the sampled data mean,

$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x_n ,$$

from each data sample to produce- a "purely AC" data sequence $x_0 - \bar{x}, x_1 - \bar{x}, \ldots . x_{N-1} - \bar{x}$.

ZERO-PADDING THE FFT

An interesting property of the FFT is that simply adding zeros after a windowed-data-samples sequence x_0, $x_1, \ldots, x_{N-1}$ to create a longer record x_0,

Listing I--This program *estimates the power spectral* distribution *of a complex data* sequence using *three different approaches: fhe zero-padded FFT, the Welch periodogram method, and Marple's autoregressive method. The subroutines listed were adapted from those presented in SL Marple's Digital Spectral Analysis with Applications, Englewood Cliffs, NJ: Prentice-Hall, 1987. The subroutines were translated to run under QuickBasic 4.5.*

```
DEFDBL A-          'Use double precision

'  COMPLEX  ARRAYS
REM $DYNAMIC:
DIM ar(1),ai(1),dr(1),di(1),rr(1),ri(1),cr(1),ci(1),xr(1),xi(1)
DIM w(1),xfftr(1),xffti(1),psd(1),win(1),xsegr(1),xsegi(1)

CLS
INPUT "Please input name of data file :",filename$
INPUT "Is data complex [y/n]?", complex8
INPUT "Sampling period [seconds] ? ", t
INPUT "Periodogram number of samples per segment ? ", nsampl
INPUT "Periodogram number of samples shift ? ", nshiftl
INPUT "Auto-regressive model order ? ", ip

'Determine length of data record
n = 0
OPEN filename$ FOR INPUT AS #1
WHILE NOT EOF(1)
    n = n + 1
    IF (complex$ = "y" OR complex$ = "Y") THEN
      INPUT #1, xr(1), xi(1)
    ELSE
      INPUT #1, xr(1)
    END IF
WEND
CLOSE #1
```

(continued)

STOP LOOK LISTEN!

Odds are that some time during the day you will stop for a traffic signal, look at a message display or listen to a recorded announcement controlled by a Micromint RTC180. We've shipped thousands of RTC180s to OEMs. Check out why they chose the RTC180 by calling us for a data sheet and price list now.



MICROMINT

MICROMINT, INC.
4 **Park Street, Vernon, CT 06066**
(203) 871-6170•Fax (203) 872-2204

in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194 • Distributor Inquiries Welcome

#119

Listing 1-continued

```
REDIM xr(n), xi(n)    'Redimension data array
' Read data into array
OPEN filenames FOR INPUT AS #1
FOR k = 1 TO n
    IF (complex$ = "y" OR complex$ = "Y") THEN
      INPUT #1, xr(k), xi(k)
    ELSE
      INPUT #1, xr(k)
    END IF
NEXT k
CLOSE #1

' Draw display screen on EGA mode 640x350 with 16 colors
SCREEN 9, 0
CLS
LINE (45, 300)-(562, 300), 3, , &H8888
LINE (45, 250)-(562, 250), 3, , &H8888
LINE (45, 200)-(562, 200), 3, , &H8888
LINE (45, 150)-(562, 150), 3, , &H8888
LINE (45, 100)-(562, 100), 3, , &H8888
LINE (45, 50)-(562, 50), 3, , &H8888
LINE (45, 300)-(45, 50), 3, , &H8888
LINE (562, 300)-(562, 50), 3, , &H8888
LINE ((562-45)/2+45, 300)-((562-45)/2+45, 50), 5, &H8888
LOCATE 22, 2: PRINT "-100";
LOCATE 18, 2: PRINT "-80";
LOCATE 15, 2: PRINT "-60";
LOCATE 11, 2: PRINT "-40";
LOCATE 8, 2: PRINT "-20";
LOCATE 4, 2: PRINT "0 dB";
LOCATE 3, 28: PRINT "RELATIVE POWER SPECTRUM":
IF (complex8 = "y" OR complex$ = "Y") THEN
    npsd = 512
    LOCATE 23, 6: PRINT "-0.5";
    LOCATE 23, 39: PRINT "0";
ELSE
    npsd = 1024
    LOCATE 23, 6: PRINT "0";
    LOCATE 23, 37: PRINT "0.25"
END IF
LOCATE 23.71: PRINT"0.5";
LOCATE 24,18: PRINT"FRACTION OF SAMPLING FRED, Fs="; 1/t; " [Hz]";
LOCATE 1,4: PRINT"PSD ESTIMATORS :";
COLOR 11: PRINT"Zero-Padded FFT ";
COLOR 12: PRINT"Welch's Method ";
COLOR 10: PRINT"Marple's Method ";

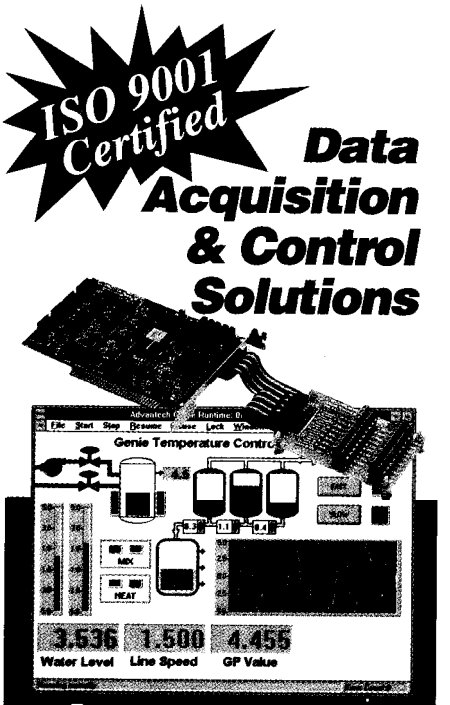
' Compute zero-padded FFT
nshift = 1
nsamp = n              'Set the periodogram for a single segment, &
wintype = 1            'Use rectangular window, in order to
GOSUB periodogram      'Compute zero-padded FFT through periodogram
col = 11: GOSUB plot 'Plot results in light blue

' Estimate the PSD through Welch's averaged periodogram method
nshift = nshiftl       'Periodogram num of sample shift between segs
nsamp = nsampl         'Periodogram num of samples per segment
wintype = 0            'Apply Hamming window
GOSUB periodogram      'Estimate PSD through Welch's method
col = 12: GOSUB plot 'Plot results in light red

' Estimate the PSD through Marple's method
GOSUB marplepsd        'Estimate PSD through Marple's AR method
col = 10: GOSUB plot 'Plot results in light green
GOTO progend

marplepsd:
' Subroutine to estimate the power spectral distribution of a
' data sequence by Marple's method. This subroutine first solves
' Marple's equations for the estimation of complex autoregressive
' coefficients from complex data. Then, it evaluates the transfer
```

(continued)



Temperature Measurement & Control Solution Package – $895

Key Features:

- For J, K, S, T, B, R, E thermocouples
- 8 thermocouple channels, expandable to 32 channels
- 0.1° C resolution with 12-bit A/D
- Up to 250 samples per second
- 16 D/I and 16 D/O channels
- Powerful Windows SCADA software
- Icon based, no programming required

- ❑ General Purpose DAS Card $295
  8 A/D, 1 D/A, 16/16 DIO, wiring kit
- ❑ Multifunction DAS Card $395
  16 A/D, 2 D/A, 32 DIO, counter
- ❑ High Performance DAS Card $595
  100KHz, programmable gain
- ❑ 24 Ch Digital I/O Card $100
- ❑ 24 Ch Digital Input w/Interrupt ...$180
- ❑ 32 Ch Digital I/O Card $170
- ❑ 32 Ch Digital Input w/Interrupt ...$235
- ❑ 144 Ch Digital I/O Card $295
- ❑ 8 Relay & 8 Digital Input Card ...$210
- ❑ High Speed Data Streaming Pkg. $595
  100KHz data streaming to disk
- ❑ PC Strip Chart Recorder Pkg. ...$695
  16 Channel recording at 15KHz
- ❑ Remote DA&C Starter Kit $495
  RS-485 based, up to 4000 feet

FREE!

232-page Solution Guide for quality minded, budget conscious Engineers

1-800-800-6889



Industrial & Lab Automation with PCs

ADVANTECH®

750 East Arques Ave., Sunnyvale, CA 94086
Tel: (408) 245-6678 FAX: (408) 245-8268

#112

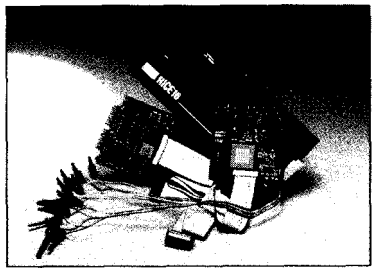
PIC16C5x/16Cxx Real-time Emulators

Introducing RICE16 and RICExx-Juniors, real-time in-circuit emulators for the PIC16C5x and PIC16Cxx family microcontrollers: affordable, feature-filled development systems from **$599** *

*Suggested Retail for U.S. only

RICE16 Features:

- Real-time Emulation to 20MHz for 16C5x and 10MHz for 16Cxx
- PC-Hosted via Parallel Port
- Support all oscillator types
- 8K Program Memory
- 8K by 24-bit real-time Trace Buffer
- Source Level Debugging
- Unlimited Breakpoints
- External Trigger Break with either "AND/Of?' with Breakpoints
- Trigger Outputs on any Address Range
- 12 External Logic Probes
- User-Selectable Internal Clock from 40 frequencies or External Clock
- Single Step, Multiple Step, To Cursor, Step over Call, Return to Caller, etc.
- On-line Assembler for patch instruction
- Easy-to-use windowed software



Emulators for **16C71/84/64** available now!

- Support 16C71, 16C84 and 16C64 with Optional Probe Cards
- Comes Complete with TASM16 Macro Assembler, Emulation Software, Power Adapter, Parallel Adapter Cable and User's Guide
- 30-day Money Back Guarantee
- Made in the U.S.A.

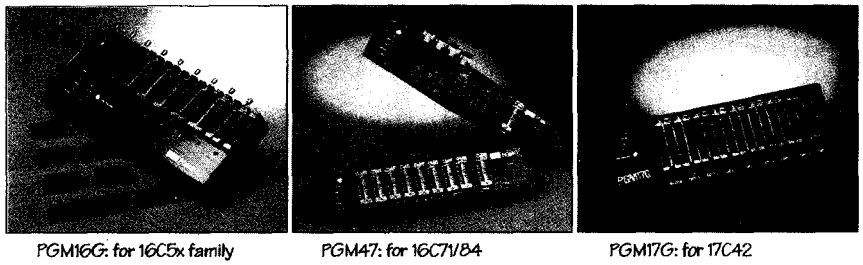
RICE-xx Junior series

RICE-xx "Junior" series emulators support PIC16C5x family, PIC16C71, PIC16C84 or PIC16C64. They offer the same real-time features of RICE16 with the respective probe cards less real-time trace capture. Price starts at $599.

PIC Gang Programmers

Advanced Transdata Corp. also offers PRODUCTION QUALITY gang programmers for the different PIC microcontrollers.

• Stand-alone COW mode from a master device ■ PC-hosted mode for single unit programming ■ High throughput. Checksum verification on master device ■ Code protection ■ Verify at 4.5V and 5.5V ■ Each program cycle includes blank check, program and verify eight devices ■ Price5 start at **$599**



PGM16G: for 16C5x family PGM47: for 16C71/84 PGM17G: for 17C42

Call (214) 980-2960 today for our new catalog.

For RICE16.ZIP and other product demos, call our BBS at (214) 980-0067.

TransdAtA Advanced **Transdata** Corporation Tel **(214) 980-2960**
14330 Midway Road, Suite 120. Dallas, Texas 75244 Fax (214) 980-2937

#113

$x_1, \ldots, x_{N-1}, 0, \ldots, 0$ before performing the FFT causes the FFT to interpolate transform values between the N original transform values. This process, called zero padding, is often mistakenly thought of as a trick to improve the inherent resolution of the FFT. Zero padding, however, provides a much smoother PSD and helps annul ambiguities regarding the power and location of peaks that may be scalloped by the nonzero-padded FFT.

CLASSICAL METHODS

As mentioned before, a common mistake is to assume that the solution to Equation (2), the so-called *periodogram,* is a reliable estimate of PSD. Actual proof of this is beyond the scope of this article. But, it has been demonstrated that regardless of how large N is (the number of available data samples), the statistical variance of the estimated periodogram spectrum does not tend to zero. This statistical inconsistency is responsible for the lack of reliability of the periodogram as a spectral estimator.

The solution to this problem is simple, however. If a number of periodograms are computed for different segments of a data record, their average results in a PSD estimate with good statistical consistency. Based on this, Welch [1] proposed a simple method to determine the average of a number of periodograms computed by overlapping segments of the available data record.

Welch's PSD estimate $\hat{P}(f)$ of M data samples is the average of K periodograms $P(f)$ of N points each:

$$\hat{P}(f) = \frac{1}{K} \sum_{i=1}^{K} P(f) \qquad (4)$$

where $P(f)$'s are obtained by applying Equation (2) on appropriately weighted data.

It is obvious that, if the original M-point data record is divided into segments of N points each, with a shift of s samples between adjacent segments, the number of periodograms that can be averaged is:

$$K = \text{Integer}\left(\frac{M-N}{S+1}\right) \qquad (5)$$

Listing I-continued

```
' function of the estimated AR system by using the FFT.

' Input Parameters:
      n      Number of data samples (integer)
      ip     Order of linear prediction model (integer)
      xr,xi  Array of complex data samps xr(1),xi(1) xr(n),xi(n)
      npsd   Power spectral distribution length

' Intermediate Parameters:
      p      Real linear prediction variance at order ip
      ar,ai  Array of complex linear prediction coefficients

' Output Parameters:
      psd    Array containing real power spectral distribution,
               with a maximum power of psdmax

REDIM ar(ip), ai(ip), dr(ip + 1), di(ip + 1), rr(ip) ri(ip)
REDIM cr(ip + 1), ci(ip + 1)

r1 = 0
FOR k = 2 TO n 1
    r1 = r1 + 2 * (xr(k) ^ 2 + xi(k) ^ 2)
NEXT k
r2 = xr(1) ^ 2 + xi(1) ^ 2
r3 = xr(n) ^ 2 + xi(n) ^ 2
r4 = 1 / (r1 + 2 * (r2 + r3))
p = r1 + r2 + r3
delta = 1 r2 * r4
gamma = 1 r3 * r4
lambdar = r4 * (xr(1) * xr(n) xi(1) * xi(n))
lambdai = -r4 * (xr(1) * xi(n) xr(n) * xi(1))
cr(1) = xr(n) * r4: ci(1) = xi(n) * r4
dr(1) = r4 * xr(1): di(1) = -r4 * xi(1)
m = O
IF ip = 0 THEN
  p = (.5 * r1 + r2 + r3) / n
  LOCATE 1, 1: BEEP: PRINT "ERROR: Zero AR model order '        "
  GOT0 progend
END IF

'   Main loop of Marple's Modified Covariance algorithm
marpleloop:
m = m + 1
savelr = 0
saveli = 0
FOR k = m + 1 TO n
    savelr = savelr + xr(k) * xr(k m) + xi(k) * xi(k m
    saveli = saveli xr(k) * xi(k m) + xi(k) * xr(k m
NEXT k
savelr = 2 * savelr: saveli = 2 * saveli
rr(m) = savelr: ri(m) = -saveli
thetar = xr(n)*dr(1)-xi(n)*di(1): thetai = xr(n)*di(1)+dr(1)*xi(n)
psir = xr(n)*cr(1)-xi(n)*ci(1): psii = xr(n)*ci(1)+cr(1)*xi(n)
xir = xr(1)*dr(1)+xi(1)*di(1): xii = xr(1)*di(1)-xi(1)*dr(1)

IF m <> 1 THEN
  FOR k = 1 TO m 1
      thetar = thetar + xr(n-k) * dr(k+1) xi(n-k) * di(k+1
      thetai = thetai + xr(n-k) * di(k+1) + xi(n-k) * dr(k+1
      psir = psir + xr(n-k) * cr(k+1) xi(n-k) * ci(k+1)
      psii = psii + xr(n-k) * ci(k+1) + xi(n-k) * cr(k+1)
      xir = xir + xr(k+1) * dr(k+1) + xi(k+1) * di(k+1)
      xii = xii + xr(k+1) * di(k+1) xi(k+1) * dr(k+1)
      rr(k) = rr(k)-(xr(n+1-m)*xr(n+1-m+k)+xi(n+1-m)*xi(n+1-m+k))
      ri(k) = ri(k)-(-xr(n+ 1-m)*xi(n+1-m+k)+xi(n+1-m)*xr(n+1-m+k
      rr(k) = rr(k)-(xr(m)*xr(m-k)+xi(m)*xi(m-k))
      ri(k) = ri(k)-(xr(m)*xi(m-k)-xi(m)*xr(m-k))
      savelr = savelr+rr(k)*ar(m-k)+ri(k)*ai(m-k)
      saveli = saveli+rr(k)*ai(m-k)-ri(k)*ar(m-k)
  NEXT k
END IF
```

(continued)

HIGH-RESOLUTION METHODS

The main limitation of FFT-based methods is restricted spectral resolution. The highest inherent spectral resolution (in Hz) possible with the FFT is approximately equal to the reciprocal of the time interval (in seconds) over which data for the FFT is acquired. This limitation, which is further complicated by leakage and the picket-fence effect, is most noticeable when analyzing short data records.

It is important to note that short data records not only result because of the lack of data (such as when sampling a short transient at a rate barely enough to satisfy Nyquist's criterion), but also from data sampled from a process which slowly varies with time.

For example, by analyzing the vibrations picked up from an oil-well drill, the operator can monitor the buildup of resonance in the long pipe that carries the torque to the drill bit, avoiding costly damages to the equipment [2]. Although a continuous signal from the vibration transducers is available for sampling, the vibrations on the drill assembly change rapidly, resulting in a limited number of data samples which represent each state of the drill bit. It is here where high-resolution estimates would be desirable, even though the data available is limited.

A number of so-called *high-resolution spectral estimators* have been proposed. These alternative methods do not assume, as the FFT does, that the signal outside of the observation window is merely a periodic replica of that observed. Instead, for instance, the parametric estimator relies on the selection of a model, which suitably represents the process generating the signal, to capture the true characteristics of the data outside of the window. By determining the model's parameters, the theoretical PSD, implied by the model, can be calculated and should represent the signal's PSD.

Many signals encountered in real-world applications are well approximated by a rational transfer function model. For example, human speech can be characterized by the resonances

of the vocal tract that generates it. In turn, these resonances are well represented by the poles of a digital filter. Parameters for the filter can be estimated so that the filter could turn a white-noise input into a signal of interest. From the filter's transfer function, we could easily estimate the PSD of the signal.

Various kinds of filter structures exist and are often classified according to the type of transfer function they implement. An all-pole filter is called an ***autoregressive*** (AR) model, an all-zero filter is a moving-average (MA] model, and the general case of a pole-zero filter is called an *autoregressive-**moving-average*** (ARMA) model. With the last example, the model best suited for speech is then an AR model.

Although high-resolution estimators have been implemented for all these models, AR-based estimators are the most popular because many computationally efficient algorithms are available. A well-behaved set of equations to determine the AR parameters with a computationally efficient algorithm has been introduced by Marple [3].

In the model of Figure 2, the AR-filter coefficients a,, $a_1, ., a_p$ are estimated by Marple's algorithm based on the input data samples $x_0, x_1, .,$ x_{N1}. The model assumes that a white-noise source drives the filter in which the output is regressed through a chain of delay elements z^{-1} from which p taps feed the AR coefficients. The system's transfer function can then be computed efficiently through the FFT, resulting in an estimate of the signal's PSD.

The performance of Marple's estimator is startling. Figure 3a presents three spectral estimates obtained from a short 64-point complex-test-data set suggested by Marple. Estimates obtained through the zero-padded FFT periodogram, Welch's averaged periodogram, and Marple's method can be compared to the theoretical spectrum of Figure 3b. Only positive-frequency PSD estimates are shown for clarity.

Notice that the closely spaced components cannot be resolved by either of the classical methods, but

Listing 1-continued

```
clr = -savelr / p: cli = -saveli / p
ar(m) = clr: ai(m) = cli
p = p *  (1  clr  ^ 2  cli  ^ 2)
IF m <> 1 THEN
   FOR k = 1 TO m / 2
       mk = m - k
       savelr = ar(k): saveli = ai(k)
       auxr = savelr + clr * ar(mk) + cli * ai(mk)
       ai(k) = saveli  clr  *  ai(mk) + cli  *  ar(mk)
       ar(k) = auxr
       IF k <> mk THEN
          ar(mk) = ar(mk)+ clr  *  savelr + cli * saveli
          ai(mk) = ai(mk)  clr  *  saveli + cli * savelr
       END IF
   NEXT k
END IF
IF m = ip THEN
   p  =  .5  *  p  /  (n m)
   GOT0 arpsd
END IF
'    Time update of C,D vectors and GAMMA,DELTA,LAMBDA scalars
rl = 1 / (delta  *  gamma  lambdar  ^ 2 lambdai ^ 2)
clr = (thetar * lambdar + thetai * lambdai + psir * delta) * rl
cli = (-thetar * lambdai + thetai * lambdar + psii * delta) * rl
c2r = (psir * lambdar psii * lambdai + thetar * gamma) * rl
c2i = (psir * lambdai + psii * lambdar + thetai * gamma) * rl
c3r = (xir *  lambdar  + xii * lambdai + thetar * delta) * rl
c3i = (-xir * lambdai + xii * lambdar + thetai * delta) * rl
c4r = (thetar *  lambdar  thetai * lambdai + xir * gamma) * rl
c4i = (thetar * lambdai + thetai *  lambdar  + xii  *  gamma) * rl
FOR  k  =  1  TO  (m 1) / 2  +  1
    mk = m + 1 - k
    savelr = cr(k): saveli = -ci(k)
    save2r = dr(k): save2i = -di(k)
    save3r = cr(mk): save3i = -ci(mk)
    save4r = dr(mk): save4i = -di(mk)
    cr(k) = cr(k)+clr*save3r-cli*save3i+c2r*save4r-c2i*save4i
    ci(k) = ci(k)+clr*save3i+cli*save3r+c2r*save4i+c2i*save4r
    dr(k) = dr(k)+c3r*save3r-c3i*save3i+c4r*save4r-c4i*save4i
    di(k) = di(k)+c3r*save3i+c3i*save3r+c4r*save4i+c4i*save4r
    IF k <> mk THEN
       cr(mk) = cr(mk)+clr*savelr-cli*saveli+c2r*save2r-c2i*save2i
       ci(mk) = ci(mk)+cli*savelr+clr*saveli+c2r*save2i+c2i*save2r
       dr(mk) = dr(mk)+c3r*savelr-c3i*saveli+c4r*save2r-c4i*save2i
       di(mk) = di(mk)+c3r*saveli+c3i*savelr+c4r*save2i+c4i*save2r
    END  IF
NEXT  k
r2 = psir ^ 2 + psii ^ 2
r3 = thetar ^ 2 + thetai ^ 2
r4 = xir ^ 2 + xii ^ 2
auxr = psir * lambdar psii * lambdai
auxi = psir * lambdai + psii * lambdar
aux2r = auxr * thetar + auxi * thetai
r5 = gamma (r2 * delta + r3 * gamma + 2 * aux2r) * rl
auxr = thetar * lambdar thetai * lambdai
auxi = thetar * lambdai + thetai *  lambdar
aux2r = auxr * xir + auxi * xii
r2 = delta (r3 * delta + r4 * gamma + 2 * aux2r) * rl
gamma = r5
delta = r2
lambdar = lambdar+c3r*psir+c3i*psii+c4r*thetar+c4i*thetai
lambdai = lambdai-c3r*psii+c3i*psir-c4r*thetai+c4i*thetar
IF p <= 0 THEN GOT0 arpsderr
IF (delta<=0 OR delta>1 OR gamma<=0 OR gamma>1) THEN GOT0 arpsderr
rl = 1 / p
r2 = 1 / (delta  *  gamma  lambdar  ^ 2 lambdai ^ 2)
efr = xr(m + 1): efi = xi(m + 1)
ebr  =  xr(n m): ebi  =  xi(n m)
FOR k = 1 TO m
   efr = efr + ar(k) *  xr(m  +  1  k) ai(k)  *  xi(m  +  1  k)
   efi  =  efi  +  ar(k) * xi(m + 1 k) + ai(k) * xr(m + 1 k)
```

(continued)

Listing 1-continued

```
    ebr = ebr + ar(k) * xr(n m + k) + ai(k) * xi(n m + k)
    ebi = ebi + ar(k) * xi(n m + k) ai(k) * xr(n m + k)
NEXT k
clr = ebr * rl: cli = ebi * rl
c2r = efr * rl: c2i = -efi * rl
c3r = (ebr * delta + efr * lambdar efi * lambdai) * r2
c3i = (-ebi * delta + efr * lambdai + efi * lambdar) * r2
auxr = ebr * lambdar ebi * lambdai
auxi = ebr * lambdai + ebi * lambdar
c4r = (efr * qamma + auxr) * r2: c4i = (efi * qamma auxi) * r
FOR k = m TO 1 STEP -1
    savelr = ar(k): saveli = ai(k)
    ar(k) = savelr+c3r*cr(k)-c3i*ci(k)+c4r*dr(k)-c4i*di(k)
    ai(k) = saveli+c3r*ci(k)+c3i*cr(k)+c4r*di(k)+c4i*dr(k)
    cr(k + 1) = cr(k) + clr * savelr cli * saveli
    ci(k + 1) = ci(k) + clr * saveli + cli * savelr
    dr(k + 1) = dr(k) + c2r * savelr c2i * saveli
    di(k + 1) = di(k) + c2r * saveli + c2i * savelr
NEXT k
cr(1) = clr: ci(1) = cli
dr(1) = c2r: di(1) = c2i
r3 = ebr ^ 2 + ebi ^ 2
r4 = efr ^ 2 + efi ^ 2
auxr = efr * ebr efi * ebi: auxi = efr * ebi + efi * ebr
aux2r = auxr * lambdar auxi * lambdai
p = p (r3 * delta + r4 * gamma + 2 * aux2r) * r2
delta = delta r4 * rl
gamma = gamma r3 * rl
auxr = efr * ebr efi * ebi: auxi = efr * ebi + efi * ebr
lambdar = lambdar + auxr * rl: lambdai = lambdai auxi * r
IF (p>0 AND delta>0 AND delta<=1 AND gamma>0 AND gamma<=1)
    THEN GOTO marpleloop

arpsderr:
LOCATE 1, 6: BEEP
PRINT"ERROR: Numerical ill-conditioning detected for model order>":
       m - 1;
GOTO progend

arpsd:    'Evaluate the AR model S transfer function
nfft = npsd
REDIM xfftr(nfft), xffti(nfft), psd(npsd)
xfftr(1) = 1: xffti(1) = 0
FOR k = 1 TO ip
    xfftr(k + 1) = ar(k): xffti k + 1) = ai(k)
NEXT k
FOR k = ip + 2 TO npsd
   xfftr(k) = 0: xffti(k) = 0    'Zero-oad UD to nosd
NEXT k
GOSUB fft
psdmax = 0
FOR k = 1 TO npsd
   psd(k) = p * t / (xfftr(k) ^ 2 + xffti(k) ^ 2)
   IF psd(k) > psdmax THEN psdmax = psd(k)
NEXT k
RETURN

fft:
' Subroutine to compute the complex FFT of a complex data series.

' Input Parameters:
     nfft        FFT size
     t           Sample interval in seconds
     xfftr,xffti Array of nfft complex data samples
                    xfftr(1),xffti(1) to xfftr(nfft),xffti(nfft)

' Output Parameters:
      xfftr,xffti nfft complex transform values replace original
                  data samples indexed from k=1 to k=nfft,
                  representing the frequencies (k-1)/nfft*t [Hz]
```

(continued)

they appear clearly separated in the estimate produced by Marple's method. You may also notice that Marple's estimate is "peaky" even for the smooth continuous spectral components at the far right of the PSD.

The reason for this peakiness is that a purely autoregressive filter generates a spectrum based on pure resonances. Only through the use of a moving-average could these resonances be damped to produce a perfectly smooth spectrum in regions where this is necessary. Although this limitation of AR-based estimators would lead to errors in the actual amplitudes of the PSD components, it is very well suited for the high-resolution detection of periodicities in the signal.

A price must be paid for the increase in resolution and, just as you may suspect, the computational burden of these high-resolution methods far exceed that of a simple FFT. In addition, like the selection of an appropriate window for the classical estimators, the rules for selecting an appropriate model, parameter estimation method, and model order are all but cast in stone.

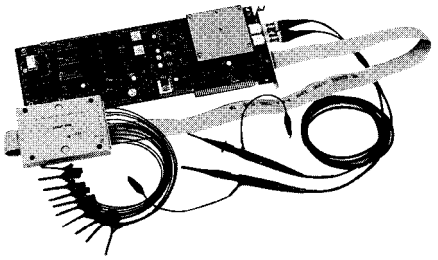
IMPLEMENTING SPECTRAL ANALYSIS ALGORITHMS

Program Spectrum.bas presented in Listing 1 demonstrates the implementation of the spectral estimation methods discussed. The program was written in QuickBASIC 4.5, but should run with little trouble under any other BASIC compiler on an IBM PC-compatible with EGA/VGA graphics. However, BASIC does not support complex-number arithmetic, so explicit operations have been used in which variable names with the suffix *r* represent the real portion of that variable, while those with the suffix *i* represent the imaginary portion of the same.

After being defined by the user, the program reads a file containing the N-data-point sequence to be analyzed. The data can be either a single column of (plain ASCII) samples or two columns, one containing the real and the other, the imaginary parts of complex data samples. The program

PC-Based Instruments
200 MSa/s DIGITAL OSCILLOSCOPE

HUGE BUFFER
FAST SAMPLING
SCOPE AND LOGIC ANALYZER
C LIBRARY W/SOURCE AVAILABLE
POWERFUL FRONT PANEL SOFTWARE



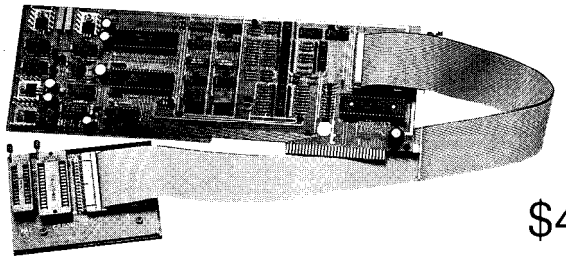
DSO Channels
2 Ch. up to 100 MSa/s
or
1 Ch. at 200 MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

Logic Analyzer Channels
8 Ch. up to 100 MHz
4K or 64K Samples/Ch
Cross Trigger with DSO

$1799 - DSO-28204 (4K)
$2285 - DSO-28264 (64K)

Universal Device Programmer

'AL
GAL
EPROM
EEPROM
-LASH
MICRO
PIC
etc..

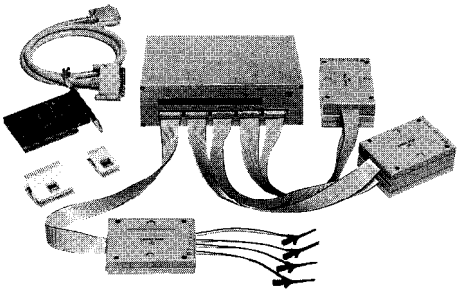


$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

- up to 128 Channels
- up to 400 MHz
- up to 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



$799 - LA12100 (100 MHz, 24 Ch)
$1299 - LA32200 (200 MHz, 32 Ch)
$1899 - LA32400 (400 MHz, 32 Ch)
$2750 - LA64400 (400 MHz, 64 Ch)

Call (201) 808-8990

369 Link Passaic Instruments Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786

#114

will estimate the spectrum of the input data using three methods:

1) A single periodogram of the data record is obtained by zero padding the data up to ***npsd*** data points (***npsd = 5*** 12 for complex and 1024 for real input data) from which the squared modulus of the FFT is computed. A rectangular window is assumed.
2) Welch's method with a Hamming window is applied using the number of samples per periodogram and the shift specified by the user.
3) Marple's method is used to estimate the PSD of the data using an AR model with model order given by the user.

Prior to its display in the output screen, PSD is normalized relative to its maximum, and transformed to decibels. For complex input data, both the positive and negative frequency sides of the spectrum are plotted. Otherwise, only the positive frequency spectrum is presented.

Because of screen resolution limitations, the number of computed PSD points for display has been limited to 512. If a larger PSD record is required, however, ***npsd*** can be increased to any desired power of 2, and a file can be opened to receive the PSD-estimate results.

A few simple demonstrations can be set up to compare the performance of the methods. First, you may generate a data file for a signal consisting of a single sinusoid at $\frac{1}{4}f_s$ with white noise added to it using the program in Listing 2.

You may vary the signal-to-noise ratio by changing the value of the noise component's coefficient. As well, the frequency of the sinusoidal component may be changed by altering the denominator of the sine argument. Of course, from Nyquist's theorem, a denominator smaller than two produces an aliased signal (you may want to experiment with the effect that this has on the PSD estimate).

In addition, the resolving power of the estimators can be compared by using a signal containing two closely separated sinusoidal components. This

Listing I-continued

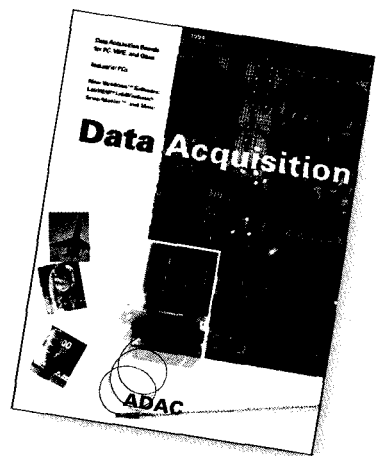
```
REDIM wr(nfft) AS DOUBLE, wi(nfft) AS DOUBLE
' Set up complex exponential table for FFT
nexp = 1
nt = 2 ^ nexp
WHILE nt < nfft
   nexp = nexp + 1
   nt = 2 ^ nexp
WEND
IF nt <> nfft THEN
   LOCATE 1, 4: BEEP: PRINT "Error!  nfft is not a Dower of 2    "
   GOT0 progend
END IF
s = 8 * ATN(1) / (nt)
clr = COS(s): cli = -SIN(s)
c2r = 1: c2i = 0
' Compute complex exponential array
FOR k = 1 TO nt
   wr(k) = c2r: wi(k) = c2i
   auxr = c2r * clr c2i * cli: c2i = c2r * cli + c2i * clr
   c2r = auxr
NEXT k

' Main FFT routine
mm = 1
11 = nfft
FOR k = 1 TO nexp
   nn = 11 / 2
   jj = mm + 1
   FOR i = 1 TO nfft STEP 11
      kk = i + nn
      c1r = xfftr(i) + xfftr(kk): cli = xffti(i) + xffti(kk)
      xfftr(kk)=xfftr(i)-xfftr(kk): xffti(kk)=xffti(i)-xffti(kk)
      xfftr(i) = clr: xffti(i) = cli
   NEXT i
   IF nn <> 1 THEN
      FOR j = 2 TO nn
         c2r = wr(jj): c2i = wi(jj)
         FOR i = j TO nfft STEP 11
            kk = i + nn
            clr = xfftr(i)+xfftr(kk): cli = xffti(i)+xffti(kk)
            auxr = xfftr(i)-xfftr(kk): auxi = xffti(i)-xffti(kk)
            xfftr(kk) = auxr * c2r auxi * c2i
            xffti(kk) = auxr * c2i + auxi * c2r
            xfftr(i) = clr: xffti(i) = cli
         NEXT i
         jj = jj + mm
      NEXT j
      11 = nn
      mm = mm * 2
   END IF
NEXT k
nv2 = nfft / 2
nml = nfft 1
j = 1
FOR i = 1 TO nml
   IF i < j THEN
      clr = xfftr(j): cli = xffti(j)
      xfftr(j) = xfftr(i): xffti(j) = xffti(i)
      xfftr(i) = clr: xffti(i)= cli
   END IF
k = nv2
WHILE k < j
   j = j - k
   k = k / 2
WEND
   j = j + k
NEXT i
FOR i = 1 TO nfft
   xfftr(i) = xfftr(i) * t: xffti(i) = xffti(i) * t
NEXT i
RETURN
```

(continued)

NEW Data Acquisition Catalog

Covers expanded low cost line.



FREE!

1994 120 page catalog for PC, VME, and Qbus data acquisition. Plus informative application notes regarding anti-alias filtering, signal conditioning, and more.

NEW Software:
LabVIEW®, LabWindows®, Snap-Master™, and more

NEW Low Cost I/O Boards

NEW Industrial PCs

NEW Isolated Analog and Digital Industrial I/O

New from the inventors of plug-in data acquisition.

Call, fax, or mail for your free copy today.

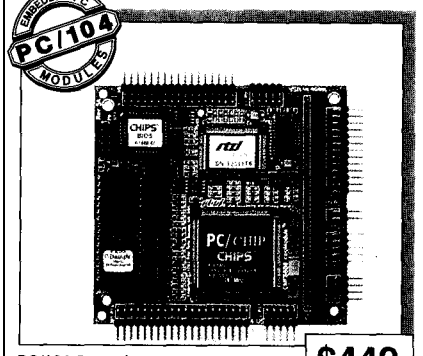
ADAC

American Data Acquisition Corporation
70 Tower Office Park, Woburn, MA 01801
Phone: (800) 648-6589 Fax: (617) 938-6553

#115

Replace Four Conventional PC/I 04 Modules with One SuperXT™ CMF8680 cpuModule™

Embedded PC/XT Controller with intelligent Power Management



PC/104 Compliant
Actual Size: 3.6 x 3.8 x 0.6"

$449 100 pcs.

- PC/XT compatibility with 286 emulation
- 14 MHz, 16-bit 8086 CPU
- +5V only; 1.6W at 14.3 MHz, 1 W at 7.2 MHz
- Intelligent sleep modes, 0.1 W in Suspend
- ROM-DOS and RTD enhanced BIOS
- Compatible with MS-DOS & real-time operating systems
- 1M bootable Solid State Disk & free software
- 4K-bit configuration EEPROM (2K for user)
- 2M on-board DRAM
- IDE &floppy interfaces
- CGA CRT/LCD controller
- Two RS-232 ports, one RS-485 port
- Parallel, XT keyboard & speaker ports
- Optional X-Y keypad scanning/PCMCIA interface
- Watchdog timer & real-time clock

Expand This Or Any PC/I 04 System with the

CM106 Super VGA Controller utilityModule™

- Mono/color STN & TFT flat panel support
- Simultaneous CRT & LCD operation
- Resolution to 1024 x 768 pixels
- Displays up to 256 colors

$223 100 pcs.

Speed Product Development with the

DS8680 Development System

Your DS8680 includes the CMF8680, CM102 keypad scanning/PCMCIA, CM1 04 with 1.8 85MB hard drive, CM106 SVGA controller & DM5406 12-bit, 100 kHz dataModule™ in an enclosure with external power supply, 3.5" floppy, keyboard, keypad, TB50 terminal board, SIGNAL*VIEW™, SIGNAL*MATH™, MS-DOS, SSD software & rtdLinx™ for just $2950.

For more information on our PC/104 and ISA bus products, call today.

rtd®

Real Time Devices USA

200 Innovation Blvd. • P.O. Box 906
State College, PA 16803 USA
(814) 234-8087 / Fax: (814) 234-5218

RTD Europa • RTD Scandinavia

Real Time Devices is a founder of the PC/104 Consortium

#117

Listing *I-continued*

```
periodogram:
'    Subroutine to compute averaged periodogram over nseg segments.
' Input Parameters:
      n        Number of data samples
      nshift   Number of samples shift between segments
      nsamp    Number of samples per segment (must be even)
      t        Sample interval in seconds
      xr,xi    Array of complex samples xr(1),xi(1) to xr(n),xi(n)
      wintype   Window type 0 = Hamming, other = rectangular

' Output Parameters:
      nseg    Number of segments averaged
      psd     Array containing real power spectral distribution,
                with a maximum power of psdmax

REDIM win(nsamp), xsegr(npsd), xsegi(npsd), psd(npsd),
      xfftr(npsd), xffti(npsd)
pi2 = 8 * ATN(1)

' Compute window
FOR k = 1 TO nsamp
   IF wintype = 0 THEN
      ' Hamming window
      win(k) = 0.538 + 0.462 * COS(pi2*(-0.5+(k-1)/(nsamp-1)))
   ELSE
      win(k) = 1   'Rectangular window
   END IF
NEXT k

' Compute Welch's averaged periodogram applying window
nseg  = (n nsamp) / nshift + 1
FOR k = 1 TO nseg
   FOR j = 1 TO nsamp
      index = j + (k 1) * nshif
      xsegr(j) = xr(index) * win(j)
      xsegi(j) = xi(index) * win(j)
   NEXT j
   FOR j = nsamp + 1 TO npsd
      xsegr(j) = 0: xsegi(j) = 0     'Zero-pad up to npsd
   NEXT j
   nfft = npsd
   FOR j = 1 TO nfft
      xfftr(j) = xsegr(j): xffti(j) = xsegi(j)
   NEXT j
   GOSUB fft
   FOR j = 1 TO npsd
      IF k = 1 THEN
         psd(j) = xfftr(j) ^ 2 + xffti(j) ^ 2
      ELSE
         psd(j) = psd(j) + xfftr(j) ^ 2 + xffti(j) ^ 2
      END IF
   NEXT j
NEXT k
psdmax = 0
FOR k = 1 TO npsd
   psd(k) = psd(k) / (nseg *  nsamp)
   IF psd(k) > psdmax THEN psdmax = psd(k)
NEXT k
RETURN

plot:
' Plot results on graph using color col, assuming npsd = 512
' (complex) or npsd = 1024 (real)
FOR k = 1 TO npsd
   psd(k) = 20*LOG(psd(k)/psdmax)/LOG(10) 'Normalize & xform data
   IF psd(k) < -100 THEN psd(k) = -100     'Clip at -100 dB
NEXT k
IF (complex8 = "y" OR complex$ = "Y") THEN
   ' Plot PSD for positive frequencies
   FOR k = 2 TO 256
      LINE(44+k+256,50-2.5*psd(k-1))-(45+k+256,50-2.5*psd(k)),col
```

(continued)

Listing 1—*continued*

```
    NEXT k
    ' Plot PSD for negative frequencies
    FOR k = 256 TO 512
       LINE(44+k-256,50-2.5*psd(k-1))-(45+k-256,50-2.5*psd(k)),col
    NEXT k
ELSE
    ' Plot PSD only for positive frequencies
    FOR k = 2 TO 512
       LINE (44+k, 50-2.5*psd(k-1))-(45+k, 50-2.5*psd(k)), col
    NEXT k
END IF
RETURN

progend:
' Wait for any key to be pressed to exit program
WHILE INKEY$ = "": WEND

END
```

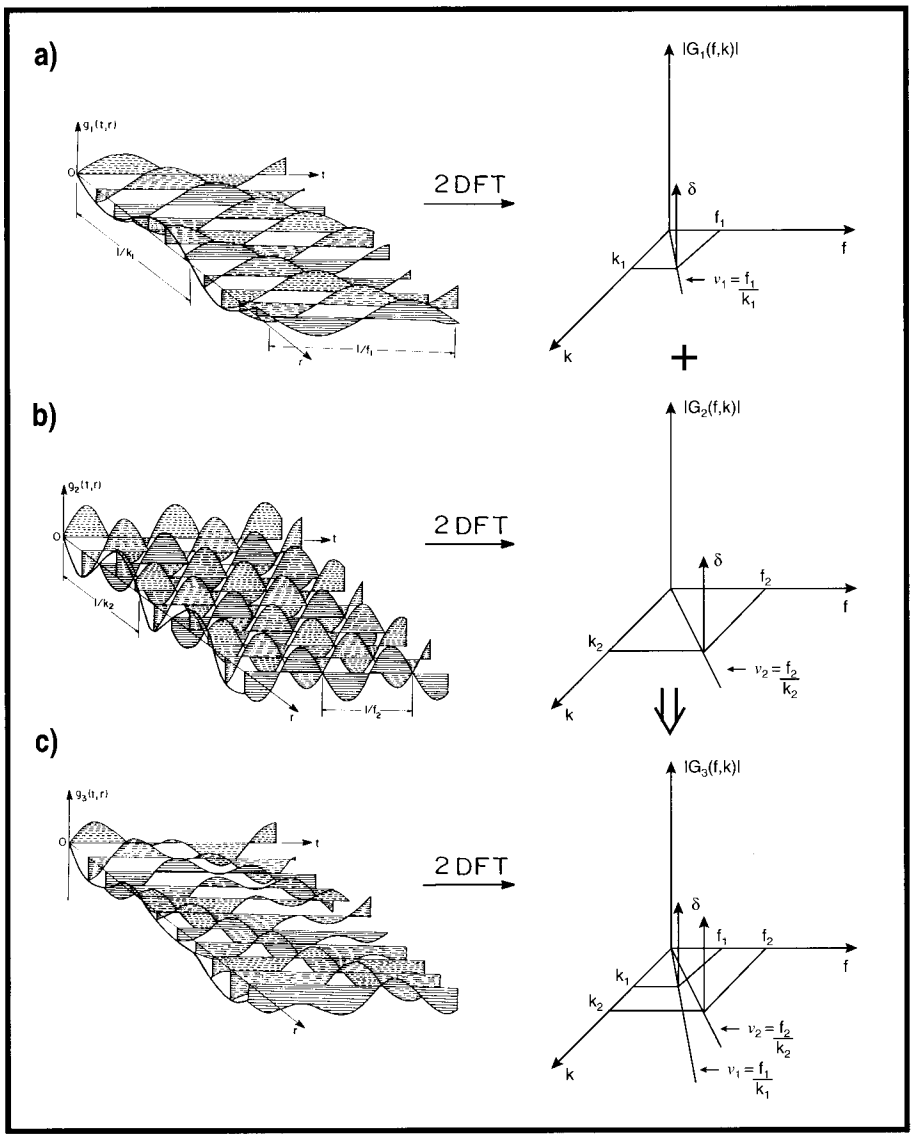


Figure 4—*Frequency-wavenumber spectra of planewaves offers (a) spatiotemporal (left) and frequency-wavenumber (right) representations of a sinusoidal wave of frequency f_1 traveling at propagation velocity v_1, (b) sinusoidal wave of frequency f_2 traveling at propagation velocity v_2, (c) a sum of the above. The two-dimensional PSDs clearly show the component waveform spectra and propagation velocity.*

can be accomplished by adding a second component to the program line which computes *x* as, for example:

$$x = 2(\text{rnd} - 0.5) + \left(\sin\left(2\pi \frac{i}{4}\right)\right) + \left(\sin\left(2\pi \frac{i}{4.1}\right)\right)$$

A rule of thumb is to keep the AR model order smaller that one-third of the number of available data samples and to allow for at least twice the number of expected spectral components. The code in Listing 1 announces an error whenever mathematical ill-conditioning is encountered due to too-high a model order. However, an unreasonably "peaky" spectrum is often obtained before ill conditioning can be detected.

AN APPLICATION: ARRAY SIGNAL PROCESSING

The greatest interest in high-resolution spectral estimators has been generated in the field known as *array signal processing*. Here, a number of sensors are placed at various locations in space to detect traveling waves.

For example, in seismology, a number of sensors capable of detecting the shock waves of a tremor or earthquake are spread over a certain area. As the shock waves travel under the sensor array, signals from each sensor can be sampled in real time, producing a data record which also contains information regarding the spatial characteristics of the waves (because the sensor locations are known). The processing of resulting spatiotemporal data is meant to estimate the number, vector velocity (speed and direction), and wave shape of the overlapping traveling waves in the presence of interference and noise.

Array signal processing has been successfully applied to many fields besides seismology [4]. For example, in exploration seismogeology, it has been used to image limited regions of the Earth's interior. In passive sonar, it can determine the location and signature of submerged sound sources; in radar, it acquires targets with very high spatial resolution; and in biomedical diagnosis, it tracks weak electrical potentials from the brain, nerves, and

muscles. Other applications include image reconstruction from projections such as radio astronomy and medical tomography.

The most common form of traveling wave is the planewave. In its simplest form, a planewave is a sinusoidal wave that not only propagates through time *t*, but also through space. In the direction of propagation *r*, this wave can be represented by:

$$g(t, r) = A \sin\left(2\pi\left(ft - \frac{r}{v}\right)\right) \qquad (6)$$

where ***A*** is the amplitude of the wave, *f* is its temporal frequency (Hz = 1/s), and v is the velocity (in m/s or other suitable velocity units) at which the wave propagates through space.

If one such simple planewave is sampled discretely along time and space, we would obtain a record similar to that presented in the left side of Figure 4a. As you see, at any given time the spatial sampling of the wave also forms a sinusoid with frequency k,. The spatial frequency (in 1/m) of such a simple planewave is called the ***wave number,*** and is given by:

$$k = \frac{f}{v} \qquad (7)$$

Its physical meaning indicates that at a distance *r* from the origin, the phase of the wave accumulates by $2\pi kr$ radians. The two-dimensional spectrum of the planewave in our example would be an impulse δ (the spectrum of a sinusoid) located in the frequency-wave number $(f - k)$ plane at f_1, ***k,.*** Through this kind of spectral analysis, we infer the components of the waveform and their velocity because the slope at which the components are found is equal to their propagation velocity or, in this case,

$$v_1\left[\frac{m}{s}\right] = \frac{f_1\left[\frac{1}{s}\right]}{k_1\left[\frac{1}{m}\right]}.$$

Adding a second component (Figure 4b) with a different frequency and propagation velocity to the original component, we obtain a planewave (Figure 4c) that, regardless of its simplicity, can hardly be recognized in the space-time domain.

Listing 2—*This* program *generates* a *file containing* ***data*** **synthesized** *from a single* **sinusoidal** *signal* **contaminated** *by* **random** *noise.*

```
pi = 3.14159262
OPEN "noise.dat" FOR OUTPUT AS #1
FOR i = 1 TO 256
  x = 2 * (RND .5) + (SIN(2 * pi * i / 4))
  PRINT #1, x
NEXT i
CLOSE #1
```

However, the two-dimensional frequency-wave number spectrum of the signal clearly resolves the components and their propagation velocities.

The two-dimensional spectrum can be computed with ease knowing that the two-dimensional DFT is computed as a sequence of one-dimensional DFTs of the columns of the data array, followed by a sequence of one-dimensional DFTs of the rows of this new array, or vice versa. As such, the most simple two-dimensional PSD estimator is implemented through the FFT. In practice, however, due to the limited number of spatial samples [because only a few sensors are normally used), the use of high-resolution estimators is essential.

Considering that enough samples $x_0, x_1, \ldots, x_{N-1}$ can usually be obtained from each of the ***R*** sensors through time, a hybrid two-dimensional spectral estimator can be implemented by combining the classical and the high-resolution spectral estimation approaches. As shown in Figure 5, using spatiotemporal data $g(t, r)$,

$$g(t, r) = \begin{vmatrix} x_{0,0} & x_{1,0} & \cdots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & \cdots & x_{N-1,1} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ x_{0,R-1} & x_{1,R-1} & \cdots & x_{N-1,R-1} \end{vmatrix} \qquad (8)$$

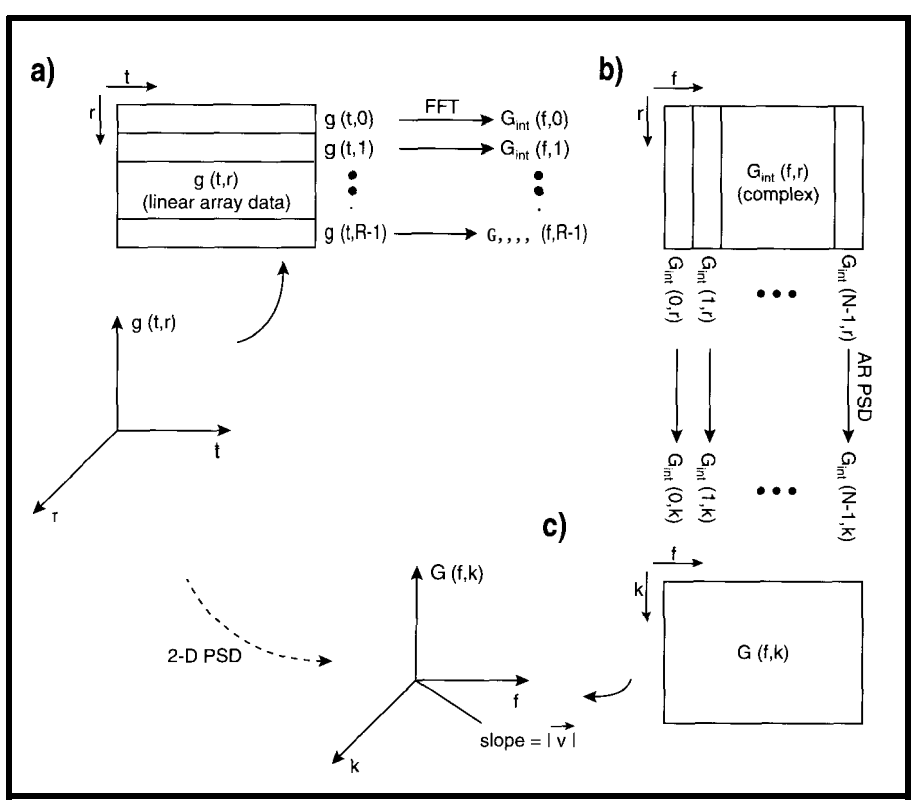


Figure 5-These *images illustrate a hybrid two-dimensional spectra/ estimator.* ***Spatiotemporal data*** *(a) is transformed along time-domain into an intermediate array(b) through the application of a windowed* ***FFT*** *to each and every row of the original data. Applying an* ***AR-PSD*** *estimator to every column of the intermediate array completes the* fwo-dimensional ***PSD estimation process.***

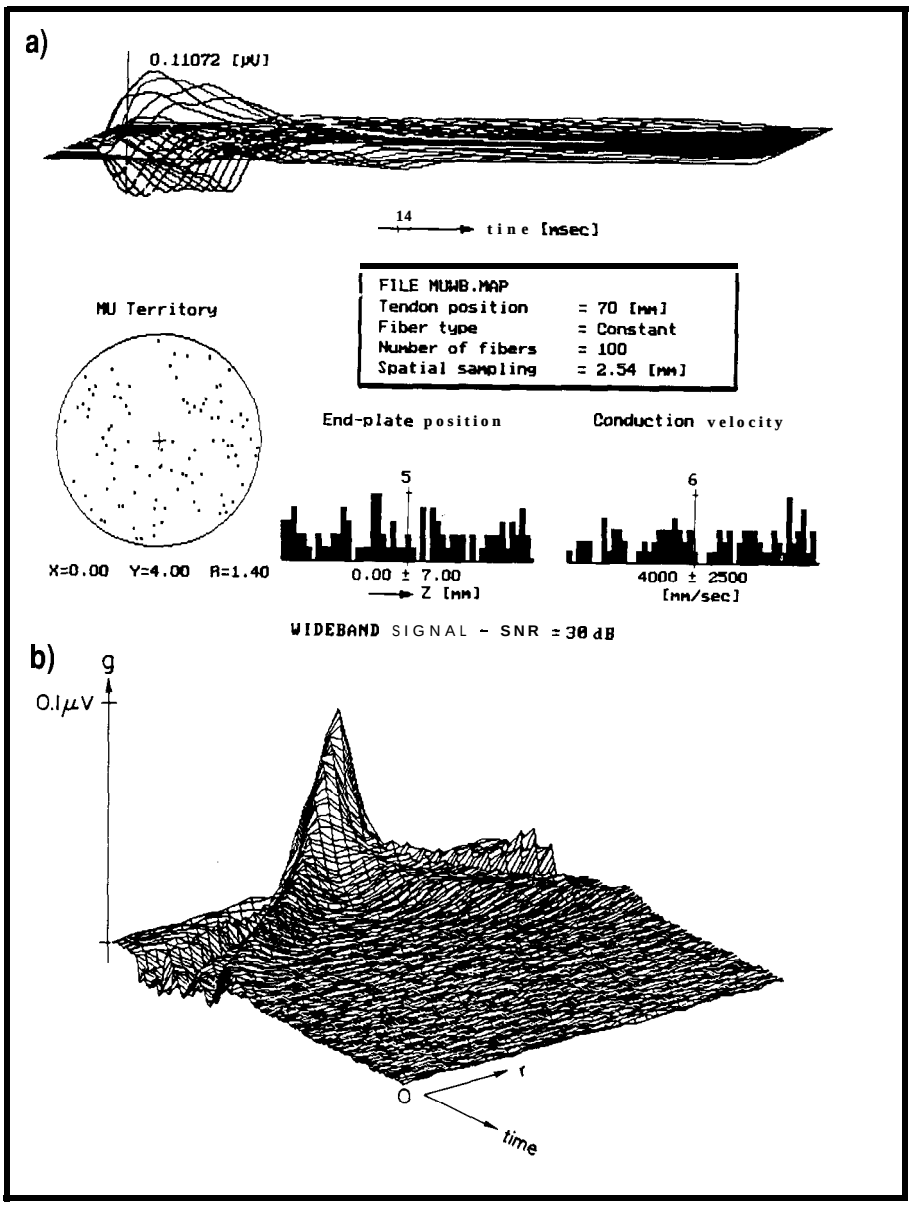


Figure 6—*Frequency-wavenumber spectral estimation has been applied to the analysis of the potentials recorded from* a muscle twitch. In (a), *the complex* spatiotemporal *waveform has been analyzed to show information about the conduction velocity, origin, and location of the component potentials; (b) shows a magnified view of the spatiotempo-*ral data.

an intermediate transform $G_{int}(f, r)$ is computed by applying the FFT along each row (time domain) of appropriately weighted data. The two-dimensional spectral estimate $g(f, k)$ is then completed by obtaining the AR-PSD of each column of complex numbers in the intermediate transform.

In the more general case, using an array of sensors spread out over an area with a planewave traveling in any direction under the array, a three-dimensional hybrid spectral estimator determines not only the wave's components and its velocities, but also each component's bearing.

For example, tiny electrical potentials can be picked up from muscle fibers using electrodes attached to the skin. These potentials are caused by pulses (action potentials) that travel down every muscle fiber causing the contraction of muscles. The conduction velocity as well as the origin of these potentials enclose a wealth of information which can be used as an aid in the early diagnosis of nerve and muscle diseases. The large number of convoluted signals and the very small differences between their waveforms makes it impossible to determine this information from spatiotemporal data (Figure 6b). However, a complete analysis [Figure 6a) is possible through the use of multidimensional spectral estimates.

IN CONCLUSION

Of course, the BASIC program listed here may be too slow to cope with most real-time applications, but implementing both classical and high-resolution methods on DSP is a relatively easy task. First of all, modern DSP chips are specifically designed to perform the convolution, vector arithmetic, and FFT operations in a minimal number of clock cycles. In addition, optimized subroutines implementing the most popular high-resolution algorithms are available often in the public domain.

Multidimensional PSD estimation has a very high intrinsic parallelism because spectral estimates are taken independently for every dimension and, as such, can be solved efficiently in parallel. In other words, since tasks in array-signal processing require specific operations to be performed on innumerable data blocks, a parallel system exploits the full power of a number of processors working concomitantly on different portions of the data to solve the larger problem.

High-power computational engines (e.g., Intel's i860) and floating-point DSPs (e.g., Texas Instruments' TMS320C30 and the AT&T DSP32) possess the raw floating-point performance necessary to efficiently implement the relevant algorithms. Unfortunately, however, these chips do not present the flexibility required to implement multiprocessor architectures which can optimally exploit intrinsic parallelism. Moreover, parallel DSP systems using these chips would most likely encounter serious communication bottlenecks imposed by their classical bus-based architectures. In these cases, RISC chips, such as the Transputer family, or DSP chips, such as the TM5320C40, which are designed for parallelism, display the full power of a scalable and very flexible architecture.

I have tried to show you that spectral analysis is a very convenient tool that serves a number of engineer-

ing applications. Moreover, with today's PCs, you have the power to implement modern PSD estimation algorithms with sufficient efficiency for experimenting and even for some real applications. With the enhanced capabilities of DSP chips, PCs with DSP coprocessors and laboratory spectrum analyzers with embedded DSPs become truly powerful and useful instruments.

However, as you understand by now, obtaining good spectral estimates is not only a matter of blindly applying the algorithm and watching the screen. Rather, knowledge about the spectral estimation methods and empirical experience of their use are of foremost importance in obtaining consistent results. ❑

David Prutchi has a Ph.D. in Biomedical Engineering from Tel-Aviv University. He is an engineering specialist at Intermedics, and his main R&D interest is biomedical signal processing in implantable devices. He may be reached at *davidp@mails.imed.com.*

REFERENCES

1. Welch, P.D., "The Use of a Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short Modified Periodograms," ***IEEE Trans. Audio Electroacoust.,*** 15(6), 1967, pp. 70–73.

2. Jangi, S. and Y. Jain, "Embedding Spectral Analysis in Equipment," ***IEEE Spectrum,*** Feb. 1991, pp. 40–43.

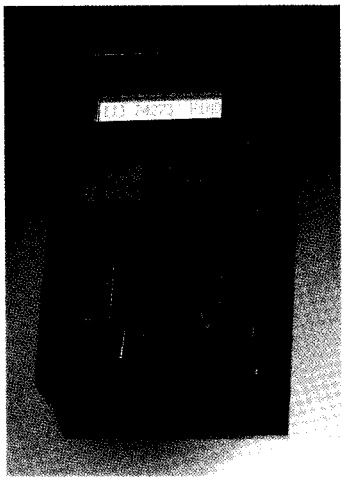
3. Marple, S.L. Jr., ***Digital Spectral Analysis with Applications,*** Englewood Cliffs, NJ: Prentice-Hall, 1987.

4. Haykin, S. ed., ***Array Signal Processing,*** Englewood Cliffs, NJ: Prentice-Hall, 1985.

I R S

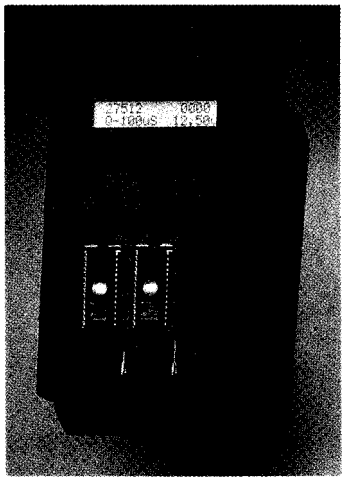
404 Very Useful
405 Moderately Useful
406 Not Useful

Data Genie offers a full line of test & measurement equipment that's innovative, reliable and very affordable. The ***'Express Series"*** of standalone, non-PC based testers are the ultimate in portability when running from either battery or AC power. Data Genie products will be setting thestandards for quality on the bench or in the field for years to come.



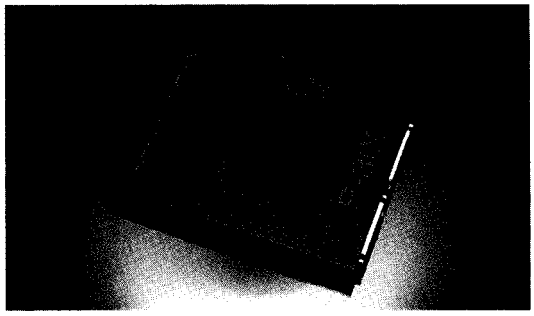
HT-28 *Express*

The HT-28 is a very convenient way of testing Logic IC's and DRAM's. Tests most TTL 74, CMOS 40/45 and DRAM's 4164-414000, 44164-441000. It can also identify unknown IC numbers on TTL 74 and CMOS 40/45 series with the 'Auto-Search' feature.
$189.95



HT- 14 *Express*

The HT-14 is one-to-one EPROM writer with a super fast programming speed that supports devices from 27328 to 27080. with eight selectable programming algorithms and six programming power (VPP) selections.
$289.95



P-300

The Data Genie P-300 is a useful device that allows you to quickly install add-on cards or to test prototype circuits for your PC externally. Without having to turn off your computer to install an add-on cards, the P-300 maintains complete protection for your motherboard via the built-in current limit fuses.
$349.95

MING
Microsystems
Division of MING E&P. INC™
17921 Rowland Street
City of Industry. CA 91748
TEL :(818) 912-7756
FAX : (818) 912-9598

Call for a dealer near you.
1-800-473-6606

Data Genie products are backed by a full 1year limited factory warranty.

©1994 MING Microsystems. All rights reserved. Data Genie logo is a registered trademark of MING Microsystems. CCDG01

#122

FEATURE ARTICLE

Alan Land

Introduction to Doremi-DSP

It may have taken Julie Andrews to make the song famous, but she likely never would have guessed that "DO-RE-MI-fa-sol-la-ti-do" would inspire a name for a new DSP technique. It goes one step beyond the old FFT.

A new standard is being sought for audio and video compression. The communication channels are crowded beyond capacity. Interactive multimedia, HDTV, image recognition, and artificial reality are as yet unfulfilled. The future of DSP, it seems, depends on finding a better way.

These statements, put in bold headlines by the media, are but symptoms of the real problem, what I call the ***DSP barrier.***

THE DSP BARRIER

The constant radix-2 record size and the constant sampling frequency of the Fast Fourier Transform (FFT) combine to create the DSP barrier. There is only one "harmonic structure" in the FFT spectrum that can easily and accurately be represented or generated. All other sine wave frequencies, except the octave harmonics of the imposed periodicity, are difficult to produce and inherently inaccurate.

The DSP barrier results from breaking a time-domain sample stream into finite-length records. As Chamberlin puts it, "If FFT synthesis is to be useful, a way must be found to produce such intermediate frequencies accurately" (Chamberlin, 1980). (The frequencies Chamberlin refers to as intermediate include all frequencies other than the apparent fundamental frequency and its exact harmonics.)

In examining the DSP barrier, it is necessary to take a general look at DSP and a much closer look at the FFT. The two outside parameters of the FFT are f, the system sampling frequency, and F, the system fundamental. F's relationship to f is measured in octaves. The number of octaves between f and F determines the system's bandwidth. However, the number of samples in the FFT record is always radix-2 and also determines-or is determined by-the number of octaves in the bandwidth.

The DSP barrier is caused by using a single sine table. Octaves of ***F*** can easily be derived from such a table by "skipping" through the sine table using power-of-two "skips." The harmonics of ***F*** (other than the octave harmonics) pose trouble. Nonoctave harmonics and non-power-of-two skips do not fit the FFT record size. The result is distorted signal and computational difficulties.

To combat this problem, a new digital signal compression technique called ***multirate sampling*** has been introduced by Aware Inc. Multirate sampling maintains a constant record size, but has variable sampling frequencies. In multirate sampling, every sine wave has the same number of samples regardless of the bandwidth. Each sample's duration is "scaled" according to the periodicity of its bandwidth.

Multirate sampling more closely defines the effect of fixed, radix-2

Note	Equation	Frea. (Hz)	Koday Equiv.
A	$440(r^0)$	440	do
A#	$440(r^1)$	466.16	
B	$440(r^2)$	493.88	re
C	$440(r^3)$	523.25	
C#	$440(r^4)$	554.37	mi
D	$440(r^5)$	587.33	fa
D#	$440(r^6)$	622.25	
E	$440(r^7)$	659.26	sol
F	$440(r^8)$	698.46	
F#	$440(r^9)$	739.99	la
G	$440(r^{10})$	783.99	
G#	$440(r^{11})$	830.61	ti
A2	$440(r^{12})$	880	do

Table 1-A musical *octave* is broken into *12 equally* spaced notes (eight of those make ***up*** a normal scale).

record sizes. However, multirate sampling does not solve the DSP barrier. All sampling frequencies-and thus bandwidths-must be subharmonics of the highest sampling frequency.

We are searching for a way to overcome the DSP barrier.

OVERCOMING THE DSP BARRIER

Having defined the DSP barrier, we must now focus on what we need from the solution. It should

- generate closely spaced, variable, and arbitrary sine wave frequencies
- compress digital signals without loss in real time
- create more accurate filters, capable of discerning very narrow bandwidths
- generate and control chromatic spectrums without the need for previously sampled signals.
- reduce power consumption

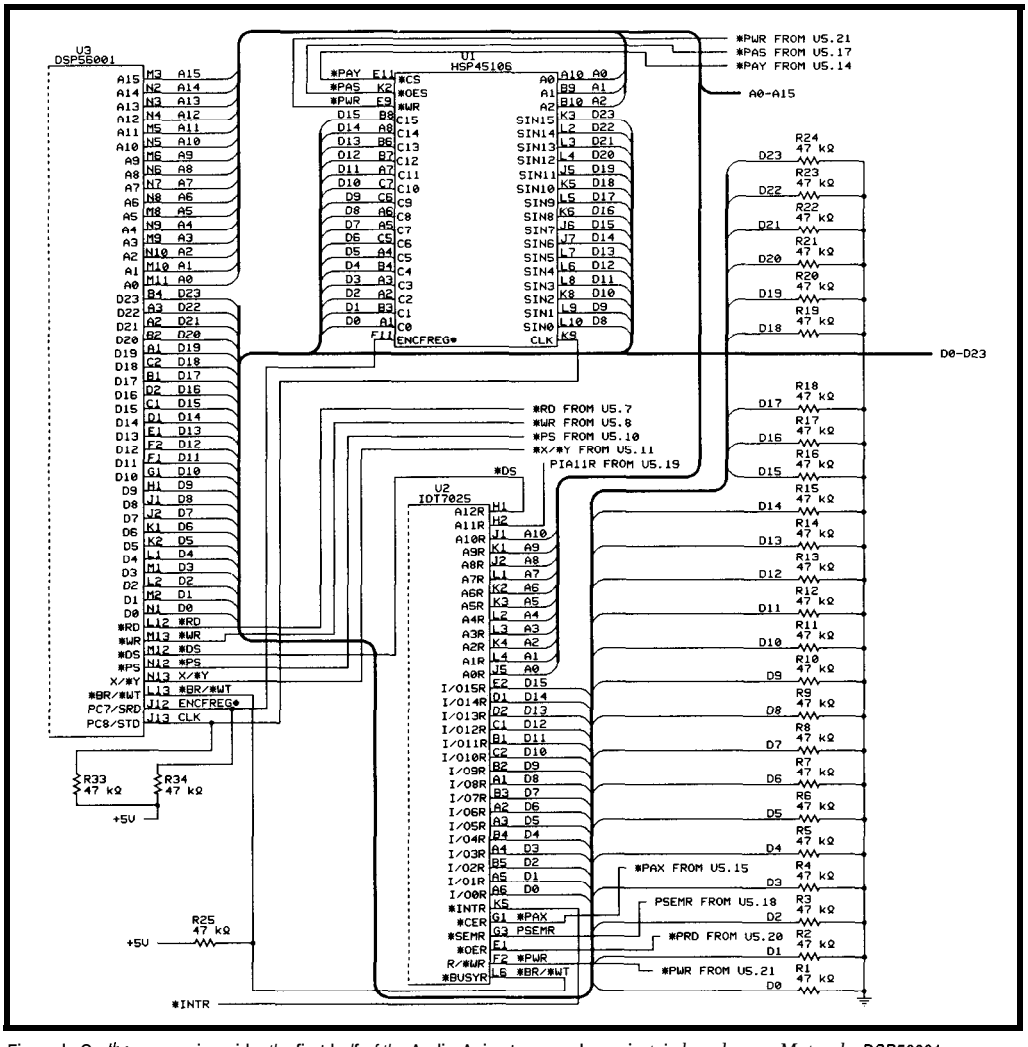


Figure la-On *the* processing side, *the* first ha/f *of the* Audio Animator sample *project* is *based on a Motorola DSP56001 processor, HSP45106* digital *sine-wave* oscillator, *and one* side *of an IDT7025* dual-ported RAM.

Ideally, this could be accomplished by increasing the usefulness of existing media and would not require "retooling the industry." In interactive multimedia, this requires a unifying theory for audio and video signals.

DOREMI-DSP

Doremi-DSP, another new DSP technique, synthesizes chromatic spectrums and compresses the signal in the process. And, it does provides a solution to the DSP barrier.

Unlike the multirate sampling technique, Doremi-DSP uses a constant sampling frequency and different record sizes. In this respect, it is the opposite of multirate sampling. The smallest record size is 2 samples while the largest record size is unlimited, (In general, the maximum record size is the same as that used for the *F* sine table.)

Doremi-DSP makes use of all the periodicities that are possible between *f* and *F,* in 1/f increments. Frequencies are computed by the equation:

$$f_{n,m} = \frac{1}{N \times \frac{1}{f}} \qquad (1)$$

where *n,m* represents an array and N is the number of samples. Like N, the array *n,m* always contains whole numbers. The following examples will show you how to use these values.

To find N, we use the equation:

$$N = INT\left(\frac{f}{\text{desiredfrequency}}\right) \qquad (2)$$

in which ***INT*** refers to the integer part of a number (the fractional part is discarded].

For example, if *f* equals 44.1 kHz and the desired frequency is 523.25 Hz, then N is equal to the integer portion of $^{44100}/_{523.25}$ or 84. If the system fundamental, F, is 441 Hz, then for *f* = 44.1 kHz, we'd have 100 equally spaced frequencies in the first octave. If we were to make a table of the equally spaced frequencies in the bandwidth, we would notice that only the frequencies in the first octave are unique and that 100 is half the total number of frequencies. The other 100 are octaves of those first-octave frequencies, spread out over the $INT\{^{logN}/_{log2}\} - 1$ octaves. We can also see that the system Nyquist equals:

$$\frac{1}{2\left(\frac{1}{f}\right)}.$$

Music theory also makes use of an equally divided octave. Originally called the equally *tempered scale,* the musical octave is divided into 12 equally spaced frequencies which use the twelfth root of two. To determine

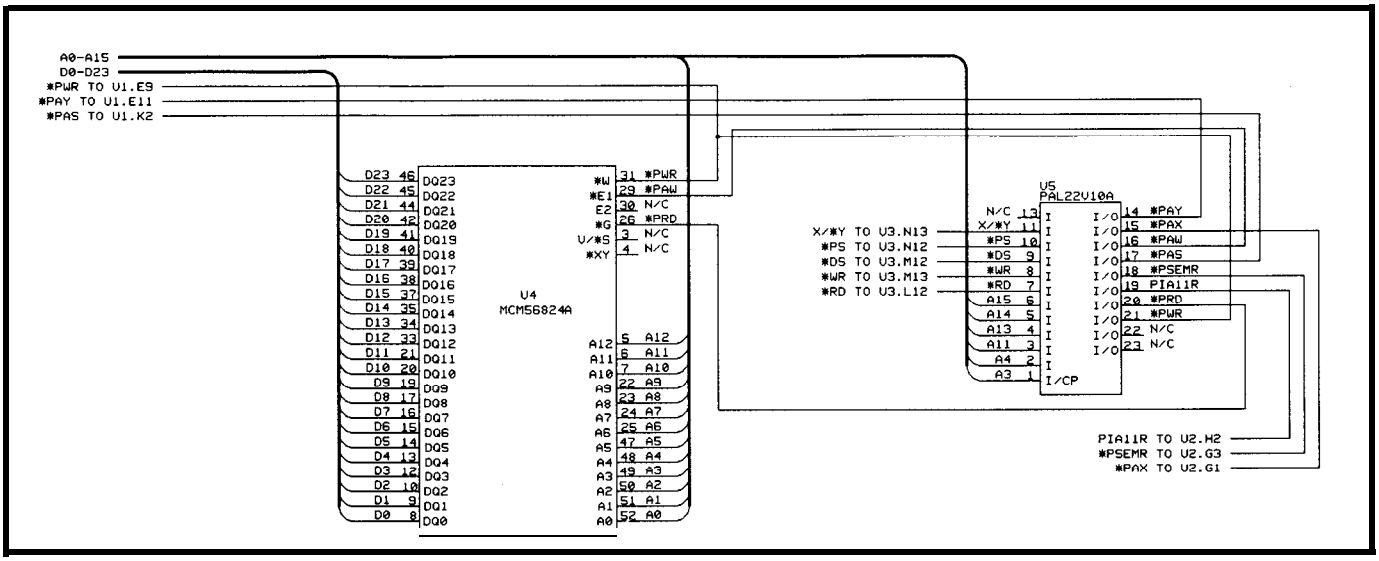


Figure 1 b—*The other ha/f of fhe Audio Animator's processing side consists of an MCM56824A DSP RAM chip and a custom PAL.*

the frequencies of specific notes of the scale, you must first determine the exact value of a twelfth root:

$$r = 2^{\frac{1}{n}}$$

where r is the value of the root and n is the equal number of divisions of the octave. Since we are dealing with the 12-tone chromatic scale, n equals **12.** If you implement the value of r (r = 1.0595) into the equations of Table 1, you get the frequencies of the "A440" musical scale.

With the equally divided octave, the frequency is arbitrarily chosen. To see the real mathematical structure, we could have used 1 Hz as the frequency. However, Doremi-DSP cannot exactly imitate the algorithm of the equally tempered scale, so we emulate it using the samples as the most obvious divisor.

Previously, we mentioned that N has a minimum of two and no maximum value. We saw in the example Doremi-DSP digital spectrum that for N = 100, we have 100 equally spaced frequencies and 100 samples for the F sine table. Each one of the 100 frequencies can be made into a sine table.

According to this method, each sine table must contain a perfect sine wave which is continuous from end to beginning within the N samples. The only way to fill in the sine table is to use a highly oversampled sine table from which we derive the other sine tables. For high fidelity, the amount of oversampling should be at least the square of the maximum record size—its effect is similar to extrapolation.

Since we can derive all the other spectrum frequencies from the sine tables of the first octave, we can cut in half the number of sine tables needed for the entire bandwidth. We can derive the other frequencies of the spectrum by skipping through the oversampled sine table using harmonic numbers.

The "how-tos" of skipping will be covered as we proceed. I'll show that, for the large number of sine tables that can be generated, very little storage space is needed. We could compute the sine samples instead of looking them up in a table. However, for our purposes, we cannot use that method.

The significance of Doremi-DSP is that we have compressed the entire spectrum into the first octave of equally divided frequencies, a location where an FFT has no frequencies at all.

DOREMI'S STANDARD SPECTRUM

Doremi-DSP is a simplified view of both analog and digital signals. Each sine wave consists of four parameters: frequency, phase, amplitude, and time envelope. Every frequency is mathematically related to f, and each parameter is measured in intervals of fixed amounts (1/f, for instance).

The standard spectrum of Doremi-DSP eliminates the need to store, transmit, or compute with digitized analog signals. As long as the transmitter or recorder and receiver or playback share the same standard spectrum, we can reconstruct the signal to the scalable resolution of the standard spectrum. We only need to store, transmit, or compute with the dynamic parameters.

It is important to realize that each frequency can be considered to be a fundamental and therefore has its own associated harmonics. However, the fundamental and its harmonics are still derived from the first-octave sine table-only they are named differently. After all, we cannot expect every harmonic structure to have a fundamental in the first octave.

As an exercise, construct a Doremi-DSP spectrum using Equation 1. With the array $f_{n,m}$, m represents each frequency of the equally divided octave and its octaves in the spectrum above the first octave and n represents the harmonics of the fundamental.

So, if we use N = 256 and f = 48 kHz, how many frequencies do we derive? Use Equation 2 to compute the number of samples needed for each harmonic. Note that n can range from

Physical Address	Logical Address
$1 FFF	P:$2FFF (top)
$OFFF	X:$27FF (top)
$07FF	Y:$27FF (top)
$0000	P:, x: & Y:$2000

Table **2**—*The configuration of the IDT7025 dual-ported RAM in the circuit overlaps some logical addresses to allow for token passing between sides.*

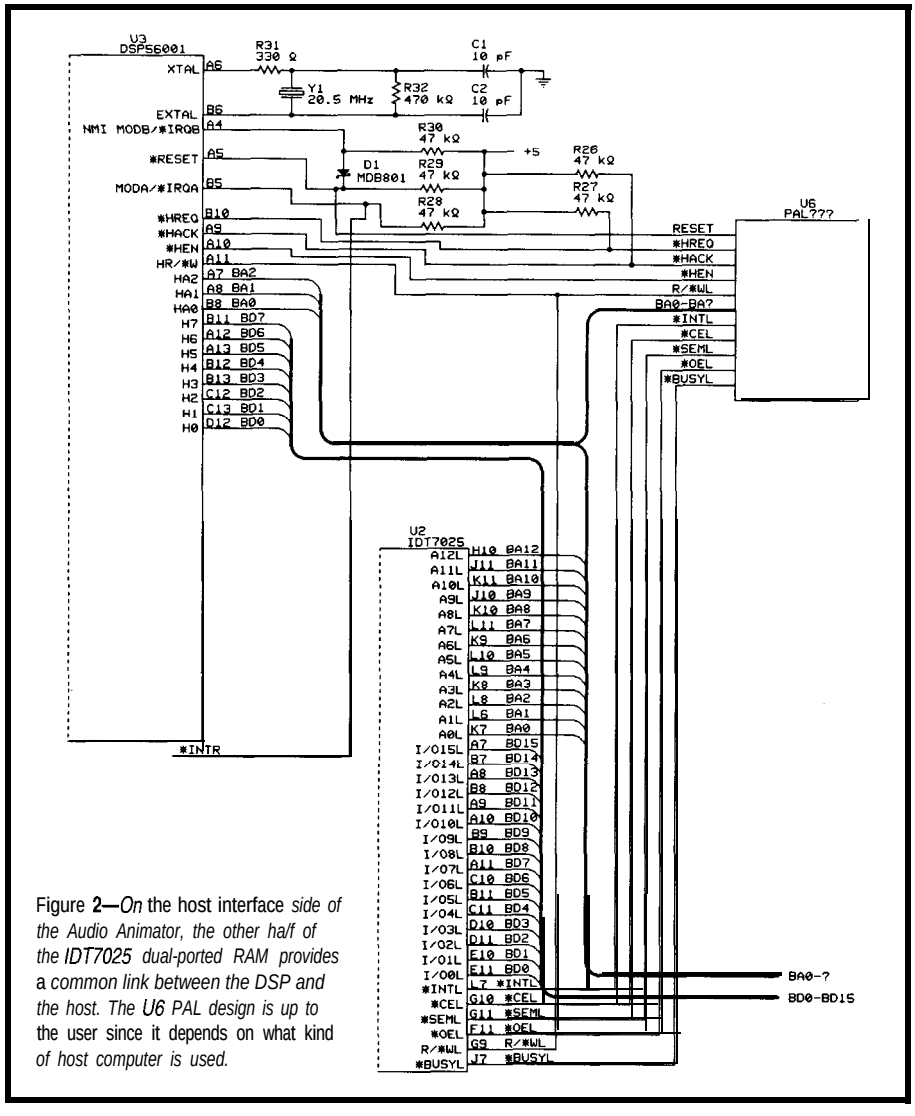


Figure 2—*On the host interface side of the Audio Animator, the other half of the IDT7025 dual-ported RAM provides a common link between the DSP and the host. The U6 PAL design is up to the user since it depends on what kind of host computer is used.*

1 to some number less than ½ (i.e., N cannot be less than 2).

The standard spectrum is the truest example of vaporware we need to encounter-it requires no storage at all. The only time sine samples are needed is during conversions between the time dom ain and the dynamic parameters representation used in Doremi-DSP.

Hence, we have accomplished many of our goals. We found a way to generate closely spaced, arbitrary-frequency, precision sine waves in the digital realm. We developed rules to ease the use of these newfound frequencies. We compressed the entire bandwidth into the first octave of the spectrum and then compressed the first octave into a single, but highly oversampled, sine table. Although the sine tables have different numbers of samples, we found a way to use them under a constant sampling frequency. Most importantly, we found many practical applications for Doremi-DSP-you can use it for compression, analysis, modification, and synthesis of digital signals, thereby improving your throughput.

Industries that may be interested in the Doremi-DSP theory include communication, entertainment, medical, scientific, education, defense, engineering, and art. Specific products targeted include telephones, television, radios, VCRs, music and voice synthesizers, spectrum analyzers, imaging equipment, and so on.

THE AUDIO ANIMATOR

Obviously, I could choose many products to illustrate Doremi-DSP. However, it was originally designed for a music synthesizer, so I will use the Audio Animator to run Doremi-DSP through its paces. The Audio Animator circuit is far from ideal. But, it does let us explore some of the more important algorithms of Doremi-DSP without resorting to custom VLSI.

Figures 1 and 2 contain only the important chip connections for the Audio Animator due to space constraints. I wanted to give you the flavor of what is necessary in implementing Doremi-DSP, so have left out application-specific information. A complete list of pins and chip interconnections is available on the Circuit Cellar BBS for anyone who really wants to duplicate my circuit. Listing 1 offers the equations for the U5 PAL chip. The choice of the U6 PAL is left to the user. See Motorola's literature for suggested host computer interfaces and PAL equations.

Doremi-DSP is in a constant evaluation process that uses more than one sine table, additive synthesis, and multiple modulo table pointers. Doremi-DSP depends on having a highly oversampled sine table and an unusual address generator. The addressing needs are emulated using a Motorola DSP56001. Although the spectrum is limited to audio frequencies in the Audio Animator, the principles apply to any spectrum.

FUN WITH VLSI DSP

The Audio Animator circuit design is as important to understand as the software used to drive it. The output of the Audio Animator is digital audio, suitable for a 16-bit D/A converter. The input to the Audio Animator represents a time-domain digital signal that has been converted to dynamic parameter representation.

The Audio Animator is built around the Harris Semiconductor HSP45 106, a numerically controlled digital sine-wave oscillator. This implementation is nonstandard, however, since we use it more like ROM than an oscillator. The host computer is interfaced though the DSP56001 and an Integrated Device Technology IDT7025 dual-ported RAM. A Motorola MCM56824A DSP RAM is used to store the temporary sine tables that the DSP56001 derives from the HSP45 106.

The Motorola DSP56001 is chosen for its unique host interface and Address Generator Unit (AGU). The IDT7025 provides a nearly ideal interface between the host compute and the Audio Animator. It serves as interface, I/O, and program and data memory for the Audio Animator as well as interface, I/O, and program and array memory for the host computer.

The IDT7025 right port is segmented so that the addresses seem to overlap (see Table 2). The user may define $2FFF and $2FFE as a mailbox. The left port interrupt flag /INT1 is set when the right port writes to memory location $2FFE and the left port clears the interrupt flag by reading address location $2FFE. The same pattern works conversely when the left port does the writing to $2FFF.

The ideal Doremi-DSP AGU would have one triplet for each sine wave as well as a fourth register for the amplitude coefficient. The AGU fetches one sine sample and its amplitude for each component sine wave of the spectrum. Next, it multiplies each sine sample by its amplitude, and finally adds the products together in the DSP56001 processor's MAC accumulator. Through this process, a convolution is performed on each time sample.

If there are five component sine waves describing the signal to be synthesized, five triplets and five amplitude coefficients are needed. Each time sample of the sequence requires five lookups (including both sine and coefficients), five multiplies, and five adds. The accumulator then outputs the product and clears for the next pass.

For the Audio Animator, we have to emulate the ideal using the DSP56001 processor's AGU. We replicate the triplets in the IDT7025 array so that the host can change them and the resulting signal in real time. The imitation triplets have the form Y:Rn,m,Y:Nn,m,andY:Mn,m.The coefficients have the form X: An , m.

We use the DSP56001 processor's AGUO (RO , N 0 , MO) triplet and move the imitation triplet into and out of IDT7025. The imitation triplet is fetched, used to fetch a sine sample, and then put back, having been automatically updated by the AGU in the process.

Remembering that *n* marks the harmonics and *m* the sine table fundamentals (or frequencies), it is easier to trace register activity. With Y: Rn , m, we have the start address and phase offset of the sine. Y : N n , m holds the harmonic number minus 1, Y: Mn , m gives the number of samples minus 1, and X: An , m gives the amplitude coefficient.

Four registers per component sine wave are written and updated by the host computer. The registers are continuously updated (automatically) for the life of the time sequence-or wave form-by the Audio Animator. We use the modified modulo addressing mode of the AGU for the triplet and simple modulo addressing for the amplitude coefficient.

The IDT7025 is an 8K x 16 dual-ported RAM which the DSP56001 sees as:

Listing 1—*The PAL on the processing side of the Audio Animator eliminates the need for a handful of discrete chips.*

```
PAL = 22V10

Notes: / = invert, * = AND, + = OR. # = Active low signal. Pin
numbers are not "fixed" in this example. You can let your circuit
board determine the best choice. "XY" is a DSP56001 signal, not a
PAL equation. The VPAL decodes the DSP56001 addresses for the
Audio Animator. Note that PRD# and PWR# are merely delayed.

Inputs: (pin = signal name)
1 = A3, 2 = A4, 3 = A11, 4 = A13, 5 = A14, 6 = A15, 7 = RD#, 8 =
WR#, 9 = DS#, 10 = PS#, 11 = XY

Outputs: 14 = PAY#15 = PAX#, 16 = PAW#, 17 = PAS#, 18 = PSEMR#,
19 = PIA11R, 20 = PRD#, 21 = PWR#

PAY# = (A3*/A4)+(DS#*XY*A15*A14*A13)
PAX# = /(/(PS#*DS#)*/A15*/A14*A13)
PAW# = /(DS#*XY*/A15*A14*A13)
PAS# = (DS#*XY*A15*A14*A13)+(/A4*/A3*/RD#)
PSMEMR# = (DS#*XY*A15*A14*A13)+(A4*/A3)
PIA11R = (/DS#*XY)+(/PS#*A11)
PRD# = /(/RD#*/(PS#*DS#))
PWR# =/(/WR#*/(PS#*(DS#))
```

Listing 2—*After putting a sine table into the DSP RAM, imitation triplets can be used in a simplified synthesis loop.*

```
RUN       DO Y1, GOROUND                    Y1 = 0
ALGO      DO X1, TIME                       X1 = no. of sines
          MOVE Y:(R1)+,R0   X:(R2)+,X0
          MOVE Y:(R1)+,N0
          MOVE Y:(R1)+,M0                   X0 = coeff.
          NOP                               pipe1 ine
          MOVE Y:(R0)+,Y0                   Y0 = sample
          MACR X0,Y0,A      R0,Y:(R3)+      ;sum of products
          MOVE N0,Y:(R3)+                   ;put back
          MOVE M0,Y:(R3)+                   ;imitation AGU
TIME      MOVE A,X:(R4)-                    ;R4 = 400FF
          CLR A                             ;output
          CMP #$0080,R4                     ;then clear
          JEQ BUFERROR                      ;full?
GOROUND   CMP #$0000,Y1                     ;done?
          JEQ IDLE
          JMP RUN
IDLE      END0
          STOP
BUFERROR                                    ;error handler
```

4K x 16 P: memory; $2000 to $2FFF
2K x 16 X: memory; $2000 to $27FF
2K x 16 Y: memory; $2000 to $27FF

Listing 2 gives an example of how the imitation triplets are used in a simplified synthesis loop. Before we can use this program, however, there must be a sine table in the MCM-56824A RAM. Putting a sine table into the RAM involves setting the Center Frequency (CF) register of the HSP-45106. The CF register is 32 bits long, but is written as two 16-bit words at Y:$FFC8 (CF LSW) and Y:$FFC9 (CF MSW). The value loaded by the DSP56001 into the CF register of the HSP45 106 is computed by the equation:

$$CF = 2^{\frac{32}{N}}$$

where N is the desired sine table size.

The sequence to load the CF register-setting it up to make a sine table-involves DSP56001's ports C and A. First of all, port C must be programmed to have two output bits, HSP CLK and HSP ENCFREG#, both normally high. The DSP56001 reads the data to place into the CF register from the IDT7025 where the host has placed it. After the DSP56001 has written to the CF register via port A, the ENCFREG# line must be held low while the HSP CLK line is toggled (see Listing 3). The ENCFREG# line is then returned high. Although this may seem unnecessary, it can't be avoided since the HSP45 106 registers are double buffered. We have to first load the CF register of the HSP45 106 then clock the internal CF register into the active Phase Frequency Control Section (PCFS).

Before creating the sine table, we have to load the DSP56001 processor's AGU with the base address of where we want the table to go in the MCM56824A. (Note: there are specific rules for modulo address space discussed in the DSP56001 user's manual.) Register Y1 needs to be loaded with N. The pseudocode for making the table is:

```
SINE    DO Y1, UNSINE
        BCLR #8,$FFE5       ;CLK
        BSET #8, $FFE5
        MOVE Y:$FFC0, Y:(R0)+
UNSINE ...
```

Listing 3—*Loading the CF (Center Frequency) register to sef if up to make a sine fable involves two bits on porf C plus all of port A.*

```
CFREG   MOVEP  Y:$2000, Y:$FFC8    ;CF lsw
        MOVEP  Y:$2001, Y:$FFC9    ;CF msw
        BCLR #7,X:$FFE5            ;clr ENCF#
        BCLR #8,X:$FFE5            ;clr CLK
        BSET #8,X:$FFE5            ;set CLK
        BSET #7,X:$FFE5            ;set ENCF#
```

BOOTTRIAL

Listing 4 is what we upload from the host computer through the HI port of the DSP56001. System vectors can be loaded at the same time as the **Boottria** 1 program into the internal P: memory of the DSP56001. Because we are using the bootstrap mode of the DSP56001, we do not need a reset vector. Instead, we load the instruction **JMP $0040** into P:$OOOO to point to the start of the program

The bootstrapping mode of the DSP56001 fetches bytes from the HI port and reconstructs them into 24-bit words, which are placed sequentially starting at P:$OOOO. Three or four bytes per P: address can be sent, but only the bytes that go to H:$5, H:$6, and H:$7 get used. H:$4 is included in case the host computer cannot break its bus into bytes. [See sections 10.2.6.2.3 of the DSP Digital Signal Processing *Databook* for more instructions.)

Boottria 1 loads important registers and tests memory. If all goes well, **Boottri** al tests the HSP45106 by creating a sine table and moving it around. After the program runs, the user should examine bits 3 and 4 of H:$2. If either bit 3, which registers a RAM error, or bit 4, which signals a

Listing 4—*The* **Boottrial** **program is uploaded from the host** computer **through** *the* HI *port of the DSP56001.*

```
P:$0000 0C0040            JMP $0040        ;skip over vector area
;
P:$0040 4FF413  CLRA      MOVE #$000000,Y1
        000000
P:$0042 08CE18  CLRB      MOVEP A,X:<<$FFE1
P:$0043 08F4A3            MOVEP #$000180,X:<<$FFE3
        000180
P:$0045 08F4A3            MOVEP #$000180,X:<<$FFE5
        000180
P:$0047 08F4BF            MOVEP #$001111,X:<<$FFFE
        001111
P:$0049 07F08C            MOVE(M) P:$2FFE,A1
        002FFE
P:$004B 60F413  CLRA      MOVE #$004000,R0
        004000
P:$004D 56F000            MOVE X:$2000,A
        002000
P:$004F 57F400            MOVE #$00AAAA,B
        OOAAAA
P:$0051 0521EE  CMP B,A
P:$0052 0E207B            JNE #007B              ;ram  error
P:$0053 44F400            MOVE #$001FFF,Y0
        001FFF
P:$0055 06C600            DO Y0,$0058            ;tests mem
        000058
P:$0057 5C5800            MOVE A1,Y:(R0)+
P:$0058 06C600            DO Y0,$005D            ;pass
        00005D
P:$005A 5DD800            MOVE Y:(R0)+,B1
P:$005B 200005  CMP B,A
P:$005C 0E205E            JNE $005E              ;bad
P:$005D 0C0060      pass  JMP $0060              ;hsp
P:$005E 00008C       bad  END0
```

(continued)

Listing 4—*continued*

```
P:$005F 0C007B            JMP $007B              ;ram  error
P:$0060  096708      hsp  MOVE Y1,Y:<<$FFC8
P:$0061  09F4C9           MOVEP #$000180,Y:<<$FFC9
P:$0062  0AA508           BCLR#7,X:<<$FFE5
P:$0063  0AA500           BCLR#8,X:<<$FFE5
P:$0064  000000           NOP
P:$0065  0AA528           BSET#8,X:<<$FFE5
P:$0066  0AA527           BSET#7,X:<<$FFE5
P:$0067  06FF80           Do #$FF, $006~         ;sine
         000060
P:$0069  0AA508           BCLR#8,X:<<$FFE5
P:$006A  000000           NOP
P:$006B  0AA528           BSET#8,X:<<$FFE5
P:$006C  0958C0           MOVEP Y:<<$FFC0,Y:(R0)+
P:$006D  61F400      sine MOVE #$002000,R1
         002000
P:$006F  06FF80           Do #$FF,$0073          ;idt
         000073
P:$0070  5CD800           MOVE Y:(R0)+,A1
P:$0071  000000           NOP
P:$0072  565900           MOVE A1,Y:(R0)+
P:$0073  0EE076      idt  JLS $0076              ;limit error
P:$0074  0c0077           JMP $0077              ;exit
P:$0075  0AA924     ram er BSET#3,X:<<$FFE9
P:$0076  0AA923     lim er BSET#4,X:<<$FFE9
P:$0077  200013      exit CLRA
P:$0078  218618           CLRB A1,Y0
P:$0079  219000           MOVE A1,R0
P:$007A  219100           MOVE A1,R1
P:$007B  000000           NOP
P:$007C  0080FA           OR1 #80,OMR
P:$007D  0AF080           JMP P:$2000
         002000
```

limit error, is high, there was trouble. The IDT7025 should contain a sine table from IDT:$0000 to IDT:$OOFF.

Priortousing Boottrial, we need to put the value $AAAA into IDT:$0800. **Boot t. r i a 1 uses** that location to find the word used to perform the memory test. A small program must also be loaded into IDT:$1000, which is P:$2000 to the DSP56001 and where Boottrial jumps when it is finished. The program can be anything, but first try something simple, such as STOP or **WAIT.**

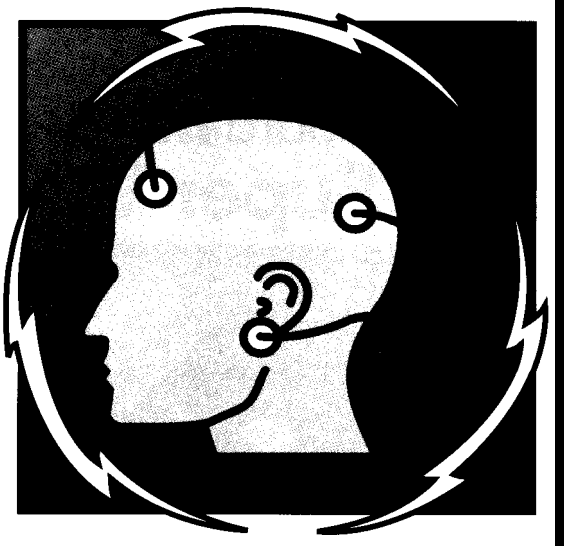
Boo t t. r i a l is simply a diagnostic trial program, not an operating system. It is meant to show some of the very first routines needed for running the Audio Animator. Since Boot t r i a 1 does not load any vectors, do not try forcing an I NT **R** until a handler is installed.

CONCLUSION

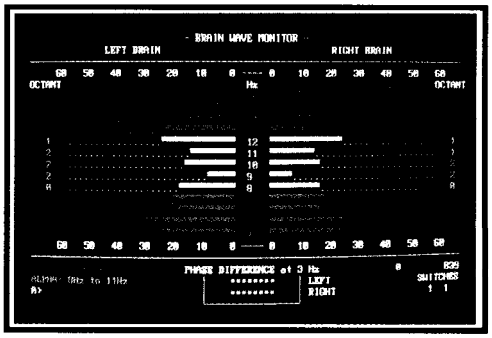
The Audio Animator circuit cannot be directly interfaced to a PC bus due to the large amount of con-

HAL-4

EEG Biofeedback Brainwave Analyzer



The HAL-4 kit is a complete battery-operated 4-channel electroencephalograph (EEG) which measures a mere 6" x 7". HAL is sensitive enough to even distinguish different conscious states-between concentrated mental activity and pleasant daydreaming. HAL gathers all relevent alpha, beta, and theta brainwave signals within the range of 4-20 Hz and presents it in a serial digitized format that can be easily recorded or analyzed. HAL's operation is straightforward. It samples four channels of analog brainwave data 64 times per second and transmits this digitized data serially to a PC at 4800 bps. There, using a Fast Fourier Transform to determine frequncy, amplitude, and phase components, the results are graphically displayed in real time for each side of the brain.



HAL-4 KIT.....NEW PACKAGE PRICE- $279 +SHIPPING

Contains HAL-4 PCB and all circuit components, source code on PC diskette, serial connection cable, and four extra sets of disposable electrodes.

to order the HAL-4 Kit or to receive a catalog,

CALL: (203) 875-2751 OR FAX: (203) 875-2204

CIRCUIT CELLAR KITS • 4 PARK STREET

SUITE 12 • VERNON • CT 06066

• The Circuit Cellar Hemispheric Activation Level detector is presented as an engineering example of the design techniques used in acquiring brainwave signals. This Hemispheric Activation Level detector is not a medically approved device, no medical claims are made for this device, and it should not be used for medical diagnostic purposes. Furthermore, safe use requires HAL be battery operated only!

tiguous memory space used in the interface. The circuit is meant to be used as a coprocessor or part of a multiprocessor environment. In a circuit such as this, we need the host tightly coupled to a dual-ported RAM for real-time, automatic, dynamic parameter changes. The Audio Animator is built for speed, not for comfort.

The Doremi-DSP project is future oriented. Much more needs to be said regarding the crucial time-domain digital-signal-to-dynamic parameters representation mentioned previously. The Audio Animator offers an experimental platform capable of letting you derive your own conclusions. We only need to agree on a standard spectrum for storage, transceiving, and synthesis, and there is no longer the need for the pulse-coded, digitized analog signal. Please try the exercise previously mentioned; a picture is worth a thousand words. The core logic depends on the ideas contained in that exercise.

At worst, you'll end up with one hell of an audio synthesizer.

Alan Land is an independent contractor to the communications industry and does custom computer designs.

REFERENCES

Chamberlin, Hal. *Musical Applications of Microprocessors.* Rochelle Park, NJ: Hayden Book Company, Inc.,1980, ISBN O-8104-5753-9.

Harris Semiconductor. DSP *Digital Signal Processing Databook,* DB302A, 1993.

Integrated Device Technology, "Integrated Device Technology Specialty Memories," 1990/9 1.

Motorola. *DSP56000/DSP56001 Digital Signal Processor User's Manual* {DSP56000UM/AD Rev 2}, 1991.

Stautner, John P. "High-Quality Audio Compression for Broadcast and Computer Applications," 26th Annual SMPTE Advanced Television and Electronic Imaging Conference.

CONTACTS

Aware, Inc.
One Memorial Dr.
Cambridge, MA 02142
(617) 577-1700
Fax: (617) 577-1710

Harris Semiconductor
1301 Woody Burke Rd.
Melborne, FL 32902
(407) 724-3000

Integrated Device Technology
3236 Scott Blvd.
Santa Clara, CA 95054
(408) 727-6116
Fax: (408) 988-3029

Motorola
P.O. Box 20912
Phoenix, AZ 85036
(602) 952-4103
Fax: (602) 952-4067

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

Embedded PC
with on-board Ethernet
and Super VGA



•25 MHz 386SL™ CPU; including up to 10 MByte DRAM
•On-board Super VGA LCD/Video controller
•On-board Ethernet, Featuring AUI and 10 BASE•T interfaces
•On-board SCSI, Floppy controllers and 2 MByte Flash Eprom Solid State Disk
•3 Serial Ports, Parallel/Printer port

PC/II+
4"x 4" Small Rugged Format
For more information call:
Megatel Computer Corporation
125 Wendell Ave., West&, Ont.
M9N 3K9 Fax: (416) 245-6505
(416) 245-2953
Megatel is a registered trademark of the Megatel Computer (1986) Corporation 386sL is a trademark of Intel Corporation

megatel

#124

C COMPILER FOR PIC CONTROLLERS

- Integrated software development environment including an editor with interactive error detection/correction.
- Access to all hardware features from C.
- Includes libraries for RS232 serial I/O and precision delays.
- Efficient function invocation mechanism allowing call trees deeper than the hardware stack.
- Special built-in features such as bit variables optimized to take advantage of unique hardware capabilities.
- Interrupt and A/D built-in functions for the C71.
- Easy to use high level constructs:

```
#include <PIC16C56.h>
#use  Delay(Clock=20000000)
#use  RS232(Baud=9600,Xmit=pin_1,RCV=pin_2)

main  0  {
   printf("Press any key to begin\n");
   getc();
   printf("1 khz signal activated\n");
   while (TRUE) {
      output_high(pin_8);
      delay_us(500);
      output_low(pin_8);
      delay_us(500);
    }
}
```

PCB compiler $99 (all 5x chips)
PCM compiler $99 ('64, '71, '84 chips)

Pre-paid shipping $5
COD shipping $10

CCS, PO Box 11191, Milwaukee WI53211
414-781-2794 x30

#125

FEATURE ARTICLE

Michael Smith and Chris Lau

Fast-scaling Routine for Floating-point RISC and DSP Processors

While the bulk of many applications can be written in a high-level language, some situations require the speed of hand-tuned, special-purpose assembly code. Check out one example.

Many algorithms require that a data array be scaled by a power of two. For example, the inverse fast Fourier transform algorithm (FFT) requires that all data values be scaled by a factor of $1/M$, where M ($=2^P$) is the number of points used [i.e., M = 64, 128, 256,...].

If the algorithm is being performed in integer arithmetic, it is simple to implement a fast-scaling operation in a single cycle using an arithmetic-shift instruction of the form:

```
; result = N / M = N / 2^p or
; result = N >> p
    SRA result, N, p
```

when p is known. This instruction operates far faster than true division.

However, a problem with integer arithmetic occurs when numbers get too large or too small to be properly stored internally. During each pass in the FFT algorithm, the numbers grow until eventually the largest numbers are too big, resulting in overflow. This problem is overcome by scaling the data after every pass. However, as the small numbers (the fine detail) get continually scaled down, accuracy is lost through truncation errors since you can't have half an integer.

To avoid these problems, it is more convenient to design algorithms using floating-point numbers since there are fewer problems with overflow, truncation, and the design of the algorithm. Many new RISC and DSP chips are capable of handling floating-point operations as quickly as integer. These processors are designed with high-speed floating-point units capable of putting out a floating-point **FADD** or **FSUB** result every cycle.

However, as with integer processors, division is performed less efficiently. The scaling of a floating-point number performing division **(FDIV)** takes 11 clock cycles on the Advanced Micro Devices Am29050 scalar RISC. Other chips perform slower as many don't have a specific floating-point hardware-divide instruction and must perform the calculation in a software routine.

The Intel i860 superscalar RISC and the Motorola MC88 100 scalar RISC take 22 and 30 cycles, respectively. Specialized DSP chips such as the Motorola DSP96002 and Texas Instrument TMS320C30 take 8 and 35 instructions, respectively, which translates into 16 and 70 clock cycles because of the longer instruction cycle. So scaling a floating-point array by the power of two takes 11-70 times longer than scaling an integer array. Even with a 40-MHz clock, that is slow.

These timings are worst-case estimates. Many of the processors are capable of performing other operations in parallel with the division instruction or procedure. If suitable instructions can be found, the effective number of cycles for the division may be somewhat smaller.

This article explains the typical floating-point-number storage format and uses this information to provide a faster scaling of a floating-point number by a power of two.

Number	Internal Representation
1.0	0x3F 80 00 00
32.0	0x42 00 00 00
31.98125	0x41 FF D9 9A
1023.4	0x44 7F D9 9A

Table 1—*IEEE defines a standard infernal representation for floating-point numbers. Here, the two pairs of numbers differ by a scale factor of32.*

FLOATING-POINT NUMBER REPRESENTATIONS

The Am29050, i860, MC88100 RISC, and DSP96002 DSP microprocessors support single- and double-precision floating-point formats that comply with the IEEE Standard for binary floating-point numbers (ANSI/ IEEE Std. 754, 1985). The TMS320C30 DSP has a similar floating-point-number representation. Table 1 illustrates the internal representations of a number of floating-point numbers using the IEEE format. The number 3 1.98 125 (see Table 1) was chosen because it represents the result of the scaling operation

$$31.98125 = \frac{1023.4}{32} \quad \frac{1023.4}{2^n}$$

Just by looking at the numbers, it is evident that the internal representations of 1 .O and 32.0 have a lot in common as do the representations of 31.98125 and 1023.4. To understand this relationship, we must go into the representations more deeply.

Figure 1 shows that, in the IEEE standard, the floating-point number is broken up into three parts in which s represents the sign bit, *bexp,* the biased exponent, and *fract,* the fractional part. The standard states that a floating-point number N will be stored internally as:

$$(-1)^n \times 1.frac \times 2^{(bexp\,127)}$$

To see how this magic incantation works, we should reconsider the numbers split into these three fields. Table 2 offers some sample numbers.

We can see where these values come from by noting that 1 .O can be written as a power of 2 using 0x1 .O x 2^0. Through this, we have:

$$-1^0 \times 1.0000 \times 2^0 = -1^n \times 1.0000 \times 2^{127-127}$$

Thus, *bexp* for the number 1 .O is equal to 127 or 0x7F, the s bit is 0, and the *frac* is 0000. The breakup of the other numbers follows a similar rule. For example, the number 10.0 is % 1010.00 in binary or $\%1.01000 \times 2^3$, which is $0x1.4 \times 2^3$.

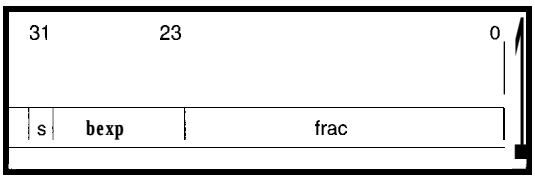


Figure 1—*The IEEE standard format for the internal representation of a sing/e-precision floating-point numbers includes the sign bif at the top of the number.*

The similarities we noticed before now can be explained by the fact that the numbers have the same *fract* parts. (The "." in 0x1.0 or 0x1.4 is not a decimal point marking the place between 1 and 1/10 in our normal numbers. Instead, it is a hexadecimal point which marks the place between 1 and 1/16 in hexadecimal numbers.)

FAST FLOATING-POINT SCALING BY A POWER OF TWO

In addition to the pattern in the *fract* part of the numbers, we can also now see a pattern in the *bexp* parts. The *bexp* from 1 .O and 32.0 differ by 5 {0x84–0x7F} as it does for 31.98125 and 1023.4 {0x88–0x83}.

This pattern occurs because both these sets of numbers differ by a scaling of 32 or 2^5. This suggests that if we can simply decrease the *bexp* of a number by 5, then we can get a quick scaling by 32. All we have to do is put the 5 in the right location as is shown using the Am29050 RISC processor syntax:

```
; set up the power
    CONST BEXPchange, 5
; shift power into "bexp" field
    SLL BEXPchange, BEXPchange, 23
; result = N / 32
    SUB result, N, BEXPchange
```

This routine takes three instructions. If we are doing many divisions by 32, the first scaling takes three instructions and the rest are done in one instruction as we can reuse the value BEXPchange.

Suppose we want to scale by a general floating-point number $M = 2^p$. To scale by *M, we* need to change *bexp* by *p.* If we know *p* beforehand, we can simply set the first instruction to:

```
; set up the power
    CONST BEXPchange, power
```

If we know *M* as a floating-point number, we can make use of the fact that the representation of *M* as a floating-point number differs from the representation of 1 .O by exactly the right factor to cause a scaling. We can modify the code to be:

```
; 1.0 into a register
    CONST ONE, $float(1.0)
    CONSTH ONE, $float(1.0)
; bexp has a value of p
    SUB BEXPchange, M, ONE
```

This revised code takes three cycles as the floating-point representation of 1 .0 is a 32-bit number that must be loaded into the register of a 32-bit RISC processor 16 bits at a time.

If we know *M* as an integer number, we could use shift operations to determine the power *p,* but this would take 4 x *p* instructions. It is far simpler to use a C 0 NV E RT instruction or subroutine to change *M* into a float. It would appear that this would add between 4 and 7 extra cycles to the fast-scaling routine for the Am29050 and i860, respectively.

However, the RISC chips are highly pipelined and the C 0 N V E RT instruction operates in parallel with other instructions such as the C 0 N ST. If you can fill the transparent processor stalls with useful instructions and use the register-forwarding capability of RISC, CONVERT takes only 1 or 4 extra cycles for the Am29050 microprocessor or i860 microprocessor, respectively. You can achieve this by:

```
; M = (float) M
    CONVERT M, M, float, integer
; 1.0 into a register
    CONST ONE, $float(1.0)
    CONSTH ONE, $float(1.0)
; transparent stall, bexp = p
    SUB BEXPchange, M, ONE
```

Number	s	bexp	frac
1.0	0	0x7F	0x00 00 00
32.0	0	0x84	0x00 00 00
31.98125	0	0x83	0x7F D9 9A
1023.4	0	0x88	0x7F D9 9A

Table *2-Breaking the numbers shown in Tab/e 1 info three separate fields gives a better idea of how they are made up.*

With any of these approaches, once the initialization has been done, further fast scaling only takes a single cycle.

Since we are changing the bit patterns, we are using integer instructions to do floating-point operations. We now have a fast floating-point scaling which takes only 1 cycle on average compared to the 1 l-70 cycles for the true FD I V instruction or subroutine. Since even the complete scaling routine operating on a single value is faster that the FD I V, this approach will work on the Am29050, i860, and MC88100 microprocessors, which have a similar number representation. The TMS320C30 routine will need some minor modifications because of its different format.

You will notice that we did not mention the DSP96002. This "oversight" is intentional; it already has an F SC A L E operation which takes a single instruction cycle (2 clocks). That is slower than the RISC performance because of the longer DSP instruction cycle, but avoids the problems discussed in the next section.

THERE ARE PROBLEMS?

This new procedure looks good and provides a very fast special floating-point scaling by numbers that are a power of two.

But, does it always work?

The answer is a definite *most of the time.*

To see a possible problem, let us suppose that N is 0.0. When we perform $^{0.0}/_{32}$, with fast scaling we should get 0.0. Instead, line two of Table 3 shows what we actually get. This response corresponds to a very large negative number (-2.126 x 10^{37}).

A similar sort of problem occurs when scaling any number whose size is smaller in size than $2^{(p\,127)}$. With fast scaling, we get a strange result: a floating-point underflow which is not detected until we output the number.

If you can guarantee that the numbers you use will never be small (or 0), then the single-instruction, fast-scaling method will work. Otherwise, we must use a more complicated routine that checks and corrects the underflow. Listing 1 gives the Am29050 RISC code for scaling an array with checks.

Number	s	bexp	frac
0.0	0	0x00	0x000000
?	1	0xFB	0x00 00 00

Table 3-Using *the technique outlined in the article,* scaling 0.0 by a *factor* of 32 leads to an incorrect *value, so additional checks are necessary.*

As the code demonstrates, after setup, the fail-safe fast scaling on the chips takes 5 or 6 instructions depending on whether or not the underflow occurs. Although this is not equivalent to the single instruction of the integer scaling, it is considerably faster than the 1 l-70 cycles of the FD I V instruction. (Note: You will have to make a few minor changes if you need to scale by a negative number $M = -2^{P}$.)

SAFELY GOING FASTER STILL

Since we could not rely on the numbers staying large enough during our algorithm, we used a routine which is 5 or 6 times slower than the single-cycle performance we wanted. For a scaling by 32.0, the number has to be smaller than 2^{5127} or 2^{-122} before problems occur. Since 2^{-10} is roughly 10^{-3}, the small number below which there is a problem corresponds to 10^{-36}. In real applications, the chances of such a small number occurring are very small. However, just once is enough to wreck your algorithm.

With the Am29050 RISC processor, there is a way of speeding the scaling and avoiding the problems by using an ASSERT instruction. The ASSERT effectively works as a software interrupt. Using this instruction, we get a fast-scaling program (see Listing 2a). In a single cycle, the instruction ASGE asserts that temp, the absolute value of the floating-point number *N,* is greater than or equal to BEXPchange. If this is true, the program can continue without jumps or delay slots to be filled. This achieves a fast scaling in only three cycles.

However, for a value that is a really small number, the program traps to a location determined by TRAP - N UM B E R. There we have the program section included in Listing 2b. This segment sets N to a number that will not cause problems when we change *bexp.* Scaling the small number takes 5 cycles plus the trap overhead of about 4 cycles for a total of 9 cycles.

Although this is slower than the 5 cycles we had before, it is faster than the 11 cycles of the FD I V instruction. However, since small numbers do not appear frequently, overall Am29050 processor performance using the fast-scaling program on an array of floating-point numbers is close to 3 cycles.

Similar code can be added to any processors that have the ability to do a "test greater than and branch" capability in a single cycle. However, the MC88100, i860, and TMS320C30 processors did not have such an instruction. Instead, they are limited

Listing *I--The first attempt at a scaling program on the Am29050 RISC processor results in somewhat slow code.*

```
        CONST   NOSIGNBITMASK, 0x7FFFFFFF; set up a sign bit mask
        CONSTH  NOSIGNBITMASK, 0x7FFFFFFF

LOOP:   LOAD    0, 0, N, arraypt          ; get value from memory
        AND     temp, N, NOSIGNBITMASK    ; get absolute value of N
        CPLT    boolean, temp, BEXPchange ; will it underflow?
        JMPT    boolean, UNDER            ; if so, clear it
        NOP                               ; unfilled delayed branch

        JMP     OKAY
        SUB     N, N, BEXPchange          ; filled delay slot

UNDER:  AND N, N, 0                       ; underflowed-set to 0.0

OKAY:   STORE   0, 0, N, arraypt          ; store the scaled value
        JMPFDEC arraysize, LOOP           ; check counter and jump
        ADD     arraypt, arraypt, 4       ; adjust array address
```

Listing 2—*The scaling routine can be sped up by adding an ASSERT trap in the main loop (a) and an ASSERT service routine (b) on the Am29050 or by using a look-up table (c) with other processors.*

```
a)          AND temp, N, NOSIGNBITMASK          ; as before
            ASGE TRAPNUMBER, temp, BEXPchange ; ASSERT software trap
            SUB N, N, BEXPchange                ; as before

b)                                  ; Jump to location "TOOSMALL" set up
                                    ; in "VECTOR TABLE" initialization
TOOSMALL: ADD N, BEXPchange, 0 ; value = BEXPchange
            RTI                     ; return from trap

c)
LOOP:       LOAD HIGHHALF, temp, arraypt  ; load the high half word
            SLL temp, temp, 4             ; turn into a word offset.
            ADD address, temp, tablestart. ; get into the table
            LOAD LOHALF, temp, address    ; get the changed bexp
            STORE HIGHHALF, temp, arraypt ; store the scaled bexp
            JMPFDEC arraysize, LOOP       ; adjust loop counter
            ADD arraypt, arraypt, 4       ; next float
```

to fast scaling in approximately 6 cycles.

With the superscalar i860's dual-instruction capability, it may be possible to initiate other floating-point operations in parallel with the integer operations of the fast scaling, so that the effective time for scaling is reduced. However, the time savings this achieves would be algorithm dependent.

Another approach is possible if you have a processor capable of half-word memory access with no penalty. All possible *bexp* and *s* values can be set up in a precalculated table. These values can then be fetched and stored over the top half word of the floating-point number. See Listing 2c.

This code only requires an additional 3 cycles to that of the loop overhead. However, it presupposes no-penalty, single-cycle access of half-word addressable memory. The setup time of the 64K-word-long table must also be taken into account. In a dedicated system in which the same calculation is repeated often, it might be worthwhile. The approach is more feasible for a processor with a floating-point representation similar to that of the TMS320C30, which has the *bexp* field entirely in the high byte. In this case, the table would only need to be 256 words long, although now fast-byte-access memory is required.

Since the FSCALE instruction on the DSP96002 conforms to the IEEE standard, the result is set to zero automatically if underflow occurs. You only add instructions for checking if you actually need to determine that fact and correct it. Normally, underflow checking is not as critical as checking overflow. Thus, the DSP-96002 performs the scaling in 1 instruction or 2 clock cycles.

AND AFTER ALL THAT?

The FSCALE instruction on the DSP96002 takes 2 clock cycles and the Am29050 RISC processor is fractionally slower (at 3 cycles) than the specialized 96002 DSP chip for this instruction. If the 3 cycles of the fast-scaling approach is not fast enough for your application, then the only thing you have left is sending nasty letters to chip designers encouraging them to add this instruction to the next chip revision. After all, it must be their fault since the Am29050 processor already performs a pipelined CONVERT operation which outputs a result every clock cycle. That instruction requires essentially all the same hardware and steps that would be needed for a true FSCALE instruction.

If you have other applications on DSP or RISC chips that ought to go fast but don't because your favorite processor lacks a particular instruction, please send details to the authors. Your problem or solution may make interesting reading for others in a future article. Or, we may wake the chip designers up to the customer's needs. ❑

Michael Smith is a professor of Electrical and Computer Engineering at the University of Calgary. He teaches courses in computer graphics, comparativeprocessorarchitecture, and systematic programming techniques. He may be reached at smith@enel.ucalgary.ca.

Chris Lau is a recent M.Sc. graduate who currently works as a cellular radio designer at Bell-Northern Research in Ottowa. His research interests include signal processing and performance analysis for indoor cellular communications systems.

REFERENCES

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

Burrus, C. S., and T. W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation,* Toronto: John Wiley and Sons, 1985.

Margulis, N. *i860 Microprocessor Architecture,* Berkely, CA: Osborne McGraw-Hill, 1990.

Motorola, *DSP96002 IEEE Floating-Point Dual-Port Processor User's Manual,* Motorola, 1989.

Motorola, MC88100 *RISC Microprocessor User's Manual,* Motorola, 1990.

Texas Instruments, *TMS320C3x User's Guide,* Texas Instruments, 1991.

Smith, M. R., "To DSP or Not to DSP?", *Computer Applications Journal 28* (August/September), 1992.

Smith, M. R., "How RISCy Are DSP Applications?", *IEEE Micro Magazine* (December] 1992.

Smith, M. R., "FFT: fRISCy Fourier Transforms," *Microprocessors and Microsystems* 17 (9), 1993.

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

DEPARTMENTS

54 Firmware Furnace

62 From the Bench

70 Silicon Update

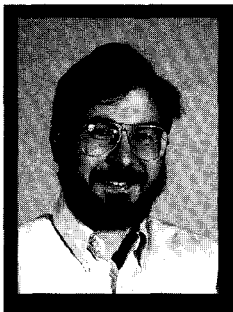
76 Embedded Techniques

83 ConnecTime

FIRMWARE FURNACE

Ed Nisley

Journey to the Protected Land: Base Camp at 1 Megabyte



While there are protected-mode operating systems already available, rolling your own is the best way to learn. Ed skips the specialized tools to show that real-mode tools can be used just as effectively to generate PM code.

Before Hillary and Norgay stood atop Mount Everest in 1953, there had been three survey missions and seven unsuccessful expeditions. None of the previous attempts made the history books, nor did any of the following climbs rate more than a passing note. There is only one First Climb and one team with name recognition.

Firmware development follows a different model. A good team can create a bauxite mine, smelt aluminum, machine ingots into carabiners, and assemble a mountain range from mine tailings before starting the climb. The race begins when they spot other explorers climbing their own self-imposed slopes in splendid isolation.

In return for this, of course, no firmware team ever gets name recognition. Ya gotta love it.. .

Several folks on the BBS suggested that, as long as I was doing protected-mode programming, I should use <name of UNIX-oid 3%bit PM operating system> because it has a small, easily understood kernel only <small integer x 100> kilobytes long. After all, <***X> supports <large integer> of <peripheral device list> and comes with <extensive tool list>. Best of all, <.* ● X> is available <on CD-ROM | by Internet ftp | from a BBS | as freeware>.

Certainly, if you have a project requiring extensive PM programming, don't start by writing the operating system! But if you'd like to know how that operating system connects to the silicon underneath, then our tiny Firmware Furnace Task Switcher should be an interesting effort because it's hard to get lost in a forest where there are so few trees. Besides, you don't have to figure out how to install and run a completely alien OS just to venture into the Protected Land.

This month the FFTS project returns to protected mode, having used a real-mode loader to read the binary file from diskette. As before, we start from scratch with the first instruction, build the new Global and Interrupt Descriptor Tables, fill in the interrupt handlers, and set the shape of the code to follow. What's new and different this time is that we're running with no support code: no protected-mode DOS extenders, no PM operating system, no device drivers, no nothing.

We're all alone with the silicon up here above 1 MB...

SMALL FOR ITS SIZE

Although FFTS runs in pure 32-bit protected mode, my choice of standard real-mode development tools imposes some unnatural restrictions. If you have a protected-mode programming environment and tools to match, be it ***X, OS/2, 32-bit Windows, or whatever, then these restrictions simply Go Away after you figure out how to load a file without an operating system. It turns out, though, that we can make considerable headway using the familiar, paid-for, DOS programs already on your system.

In fact, we need some fairly detailed knowledge of how real-mode segments work in order to prepare a protected-mode program. Even if you don't plan to write PM code, you'll probably learn something new here. I certainly did!

TASM and TLINK can produce programs that use 32-bit instructions and operands. Because the programs are intended for real mode, the tools cannot handle segments larger than 64 KB or FAR addresses using protected-mode segment descriptors without using DOS extenders. Oddly enough, though, it's not all that difficult to write pure protected-mode firmware with real-mode tools.

Listing 1 shows the standard setup lines appearing in each FFTS assembly language file. The P386 directive enables all the real- and protected-mode instructions available in 80386 CPUs. The MODEL directive selects SMALL memory model with USE32 specifying 32-bit operands and addresses. The FARSTACK option removes the stack from its normal home in the data segment and places it in a separate stack segment.

SMALL memory model also tells the assembler that all the program's code resides in the default CODESEG, which cannot exceed 64 KB. While we can (and will) define other code segments, SMALL model allows us to use NEAR CALLs and sidestep segment register reloads until we're ready.

The default SMALL model data segment contains a group of three related segments: initialized data, uninitialized data, and constants. This collection, called DGROUP, must fit in a single 64-KB segment, but we are free to define other segments to hold other data items.

Listing 2a shows the definition of one such data segment. The "constant" data segment in DGROUP can't be protected by a protected-mode,

Listing 1-Although Borland's TASM normally produces 16-bit real-mode programs, it includes features that support 32-bit protected-mode code. These lines appear in each file of the FFTS project to set the default conditions for our programs. The P386 directive enables all the instructions unique to the 386 CPU in both real and protected mode. The MODEL directive enables 32-bit code and operands, places the 32-bit stack in its own segment, and creates standard SMALL model segmentation. The INCLUDE directives pull in a variety of constants, structures, and suchlike; I put them in a common directory for all these projects.

```
IDEAL
P386
LOCALS

INCLUDE     "..\386base.inc"
INCLUDE     "..\ffts.inc"
INCLUDE     "..\firmdev.inc"
INCLUDE     "..\pmsegs.inc"

MODEL       USE32 FARSTACK SMALL,C
```

Listing 2-a) The _protconst segment defined here provides an iron-clad defense for its data. Any attempt to change a constant triggers a protection violation. b) It is no more difficult to use protected-mode segments than it is in real mode. DATASEG variables are initialized by the startup code, while UDATASEG contains uninitialized data. The constant segment is an ideal spot for values that will never change, such as messages and configuration constants.

a)

```
SEGMENT     _protconst PARA PUBLIC USE32 'PROTCONST'
ENDS        _protconst
```

b)

```
            DATASEG
StatusCtr   DD          55h
ShiftLeft   DD          00000001h
ShiftRight  DD          00000080h

            UDATASEG
PortCtr     DD          ?

            SEGMENT     _protconst
SignonMsg   DB          NL,'Firmware Furnace Task Switcher '
            DB          '-- Ed Nisley   (c) 1994 CAJ',NL
            DB          NL,'Hello from 32-bit protected mode!',NL,EOS
            ENDS        _protconst
```

read-only, data-segment descriptor because the same area is occupied by read-write data. I elected to put all the genuine constants in a separate segment called _protconst. The CPU traps any attempt to change them and pinpoints the errant instruction. This response is much better than trying to figure out where the bizarre trash came from, which is what happens when you hose the constant segment in real mode.

Segments are just as easy (or just as difficult) to use in protected mode as in real mode. Any initialized or uninitialized variables are in the default DATASEG or UDATASEG segments, respectively. The constant segment can't take advantage of TASM's simplified segment directives, which means you must remember the ENDS statement to close the segment. Listing 2b is an example of how to put data into specific segments.

As before, the GDT segment descriptors must match up with what we tell the assembler. That requires the combined efforts of the linker, LOCATE, and the FFTS startup code, so we had best begin at the beginning. I'll cover the details of real-mode segment linking because we need to understand how it works to write the startup code that loads the tables.

BACK TO BINARY

When you compile a normal DOS program, the linker produces an EXE file. Because the load address varies every time you run the program, the linker can't put the actual segment values in the file. Instead, it identifies each spot where a segment is used by making entries in the EXE file header. The DOS loader reads the file from disk, uses the EXE header entries to set the segment values to match the load address, and transfers control to the first instruction.

The LOCATE program we've been using performs the same segment fixups as the loader. The key difference is that we have precise control of the program's segment addresses. Instead of executing the tweaked program, however, LOCATE writes it back to disk as a binary file with all the segment fixups intact. You can burn the file into EPROM or, as we do, load it into RAM at the right address and execute it with no further changes.

Listing 3-Although the *Paradigm LOCATE utility works with real-mode programs, we can produce protected-mode code as long as we observe some restrictions. This CFG file tells LOCATE to put the code, data, and stack info three separate 64-KB segments, then produce a binary output file containing the code, constants, and initialized variables. The startup code relocates the segments above the 1-MB line by loading the appropriate protected mode descriptors.*

```
hexfile binary offset=00000h size=8 // binary file for boot loader

listfile segments

map 0x00000 to 0x1ffff as rdonly    // code segment
map 0x20000 to 0x2ffff as rdwr      // data segment
map 0x30000 to 0x3ffff as rdwr      // dummy stack segment
map 0x40000 to 0xfffff as reserved // the rest is unused

dup     DATA ROMDATA                // copy initialized vars to image

class   CODE = 0x0000               // Code
class   DATA = 0x2000               // Data
class   STACK = 0x3000              // dummy stack

order   DATA            \           // data organization
        BSS

order   CODE            \           // ROM organization
        PROTCONST       \
        ROMDATA

output  CODE            \           // Output file classes
        PROTCONST       \
        ROMDATA
```

Any instruction referring to code or data with a full segment:offset address requires a segment fixup. For example, a FAR CALL must include both the segment and offset of the target instruction, and an LES instruction requires a fixup for the segment address loaded into ES. In each case, the assembler reserves a word for the segment address in the instruction, and the linker puts an entry in the EXE file header indicating that a segment fixup is needed at that spot.

Each segment in the source has both a name and a class, which leads to considerable confusion. The name identifies a segment associated with a single segment-register value throughout the program: SMALL programs have a single code segment regardless of the number of source files. The linker combines all like-named segments into a single block (can't exceed 64KB), then assigns the same segment fixup value to each reference in the program.

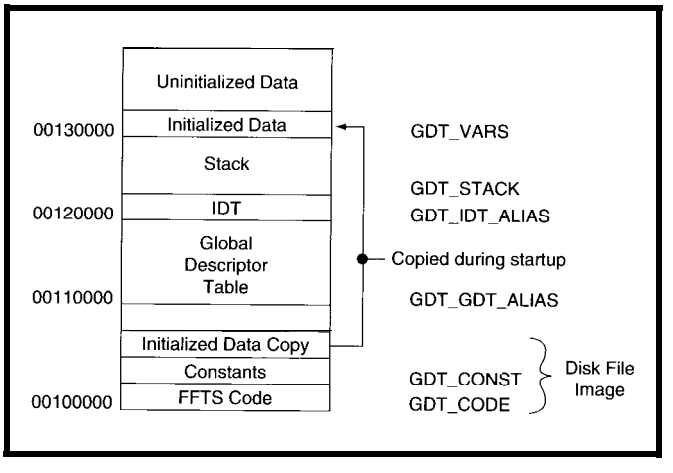


Figure 1--The *SMALL memory model has three essential segments: code, data, and stack. The startup code copies initialized data from the disk file to the start of the data segment. FFTS adds a constant data segment for unchanging values, and the GDT and IDT must be covered by data alias descriptors to change their entries. This figure shows how the various segments are laid out starting at the 1-MB line.*

Listing *4-The FFTS startup code begins by clearing storage starting at what will become the new GDT. The OLD_DS32 selector is the data segment defined by PML oade r covering RAM beyond the1-MB line. The L S L instruction fetches the segment limit in bytes, which we convert to a doubleword count. Because this code runs in 32-bit profecfed mode, a single STOS instruction can clear up to 15 MB in one shot!*

```
MOV       EAX,OLD_DS32            target in ES:EDI
MOV       ES,AX
LSL       ECX,EAX                 get limit in 32-bit register
INC       ECX                     convert from limit to bytes
MOV       EDI,BASE_GDT            starting offset for fill
SUB       ECX,EDI                 knock off the offset
SHR       ECX,2                   … convert to DWORDs
XOR       EAX,EAX                 get zeros for fill
REP  STOS [DWORD  PTR  ES:EDI]; zap!
```

A key point is that the linker processes segments and parts of segments in the order they occur in the source files. Values in the first file have offsets starting at zero, and values in the last file are assigned at the end of their accumulated segment. Controlling this order is easy in assembler but can be quite difficult for high-level language programs.

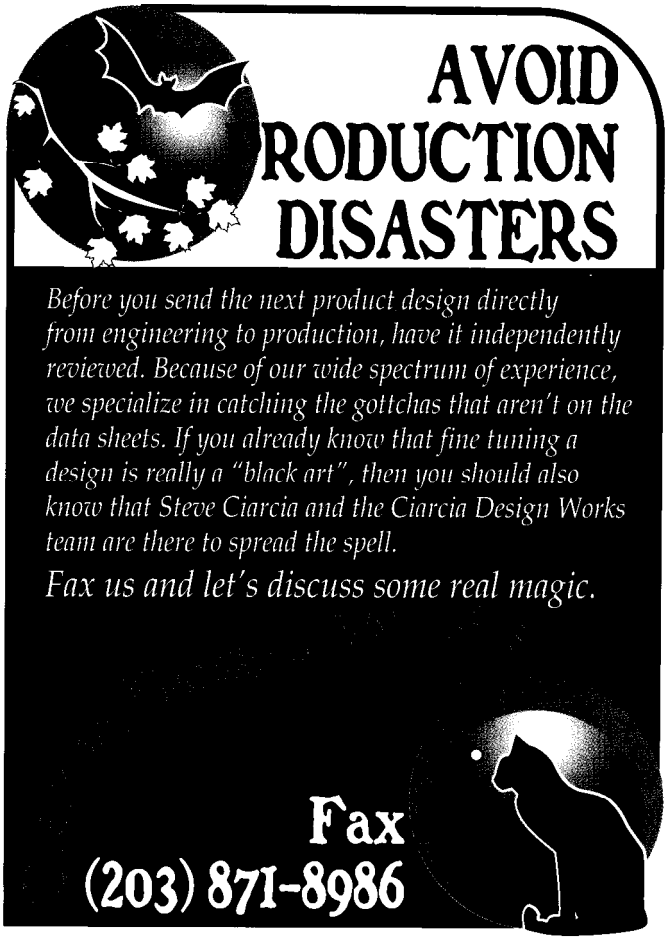
The segment's class identifies a collection of related segments that the linker handles as a unit. Each segment within a class retains its unique name and segment register value, thus the complete class may exceed 64 KB. LARGE model programs produce a separate code segment in the C 0 DE class for each source module. Earlier columns in this series used a similar technique to combine 16-bit real code with 32-bit protected code.

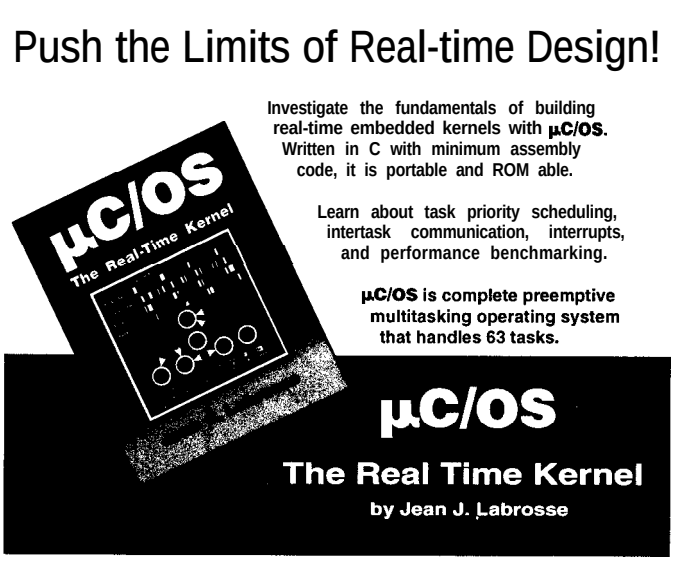
But, we're not done yet. There is a third way to combine segments! The G ROU P directive tells the assembler and linker to combine several segments into a single lump that can be accessed by a single segment-register value. The standard memory models put the initialized data, uninitialized data, and stack segments into a group called DGROUP.

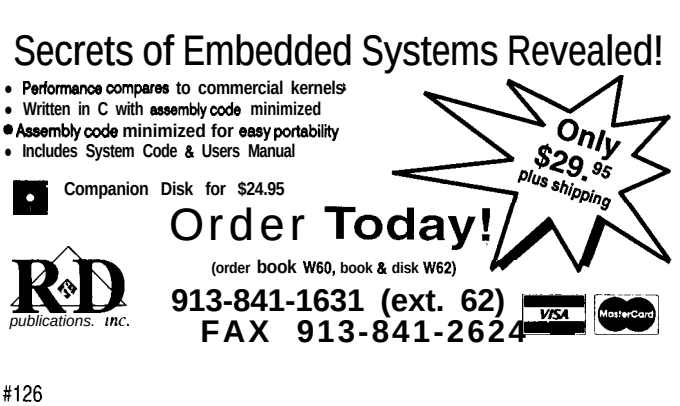
The key difference between a class and a group is that the assembler adjusts the offset of each variable in a G ROU Ped segment so it is relative to the start of the group, not the individual segment. DGROU P is often referred to as the "near data" segment. DS is loaded once at the start of the program to give access to all the segments and thus all the data in that group.

Listing 3 shows the F FTS . C FG file that tells LOCATE how to produce the binary file for our 32-bit protected-mode program. Now that you know about segments, classes, and groups, this should be easier to understand.

The C LASS directives put the CODE, DATA, and STACK classesat specific memory addresses which







#126

normally correspond to the target system's EPROM and RAM. In our application, the class addresses are essentially arbitrary because we will relocate them using PM descriptors and take care not to refer to them by their real-mode values.

The ORDER directive specifies which classes should be concatenated into a single sequence. Because ORDER uses class names instead of segment names, you can put all the CODE segments in one place with a single statement.

It turns out that the EXE file header does not include any information about the contents of GROUPs. Because LOCATE cannot discover them on its own, you must manually put the same segments and classes in the same sequence in both the GROUP and ORDER directives. The assembler and linker have already adjusted the group's segment and offset values. Dire bugs await mismatched programs.

ORDER can collect unrelated segments into a single block. The second ORDER directive in Listing 3 defines the layout of the binary file that will eventually be written to disk. As you'll see later, the FFTS startup code depends on this sequence to sort out the segments.

The COPY directive performs a vital service; it duplicates a class in a different location. Your program expects its initialized data to reside in the data segment at addresses assigned by the assembler and linker. Those initial values, however, must also be in the disk file or EPROM at an address that's not in the data segment (because you can't write to EPROM). The startup code copies the values from the file into the data segment before starting the program.

In this case, COPY duplicates the DATA class, containing the initialized data,into ROMDATA. The ORDER directive tucks ROMDATA just after the PROTCONST segment, which holds all the read-only constants.

The OUTPUT directive defines the sequence of classes in the disk file. The FFTS startup code assumes the OUTPUT and ORDER directives put the same classes in the same sequence. They're under your control for your

Listing 5—*This code copies !he GDT prepared by PM L odder to the RAM at 00110000, then loads fhe CPU's GDT register. The LGDT instruction requires a six-byte storage operand ho/ding the GDT's size and starting address, which we create in what will become the initialized data segment.*

```
MOV     EAX,OLD_GDTALIAS      ; set up source in FS:ESI
MOV     FS,AX
LSL     ECX,EAX               ; get unscrambled limit
INC     ECX                   ; convert to size in bytes
MOV     ESI,0                 ; source offset is always zero

MOV     EAX,OLD_DS32          ; set up target in ES:EDI
MOV     ES,AX
MOV     EDI,BASE_GDT

REP MOVS [BYTE PTR ES:EDI],[BYTE PTR FS:ESI]

MOV     ESI,BASE_DATA         ; aim ES:ESI at init data area

MOV     [WORD PTR ES:ESI],(8192 * SIZE DESC_NORM)-1
MOV     [DWORD PTR ES:ESI+2],BASE_LOAD+BASE_GDT
LGDT    [FWORD PTR ES:ESI]
```

Listing 6-a) *STARTUP.ASM is linked first to ensure that fhe segments are defined in the correct order. This code puts a label at the start of the _protconst segment and defines a few bytes to flag if in the binary fife. b) FINAL.ASM consists entirely of labels marking the end of segments. Each label's offset is equal to the number of bytes in ifs segment and thus the segment length. However, FINAL ASM must be linked last to ensure that the linker puts these parts of the segments after all the others. c) Code in STARTUP.ASM fills in the GDT descriptors with the starting address and limit (length-I) of each segment. The constant segment begins at the next 16-byte address boundary after the end of the code segment because _protconst was defined with PARA alignment. Rounding PMCodeLength to the next multiple of 16 gives the correct value. The ACC_DATA32 constant allows read/write access to the segment, so this code clears the ReadWrite bit to ensure that the constants cannot be changed.*

a)

```
SEGMENT  _protconst
LABEL    PMConstBase
DB       'constant'
ENDS     _protconst
```

b)

```
CODESEG
PMCodeLength:
PUBLIC   PMCodeLength

SEGMENT  _protconst
LABEL    PMConstLength BYTE
PUBLIC   PMConstLength
ENDS     _protconst
```

c)

```
MOV     EDI,BASE_GDT+GDT_CONST
MOV     EAX,(OFFSET PMConstLength)-1
MOV     [GDT_PTR.SegLimit],AX
MOV     EAX,BASE_LOAD + OFFSET PMCodeLength + 0Fh
AND     AL,0F0h
MOV     [GDT_PTR.SegBaseLow],AX
SHR     EAX,16
MOV     [GDT_PTR.SegBaseMid],AL
MOV     [GDT_PTR.SegBaseHigh],AH
MOV     [GDT_PTR.Access],ACC_DATA32 AND NOT MASK ReadWrite
MOV     [GDT_PTR.Attributes],ATTR_32BIT
```

Listing 7—*LOCATE puts a copy of the initialized variables after the code and constant values in the disk file. As before, the segments start at even paragraph boundaries, so this code rounds the actual lengths up before adding them together. DS contains the original data descriptor for the segment starting at 00100000, making offsets in the file numerically equal to offsets in the segment.*

```
MOV       EAX,GDT_VARS  ; set up target
MOV       ES,EAX
XOR       EDI,EDI       ; offset is always zero

MOV       EAX,(OFFSET PMCodeLength) + 0Fh
AND       AL,0F0h
MOV       EBX,(OFFSET PMConstLength) + 0Fh
AND       BL,0F0h
ADD       EAX,EBX
MOV       ESI,EAX

MOV       ECX,OFFSET PMDataLength
REP MOVS  [BYTE PTR ES:EDI],[BYTE PTR DS:ESI]
```

projects. The only vital requirement is the CODE class come first, so the first instruction is at offset 0 in the file.

The HEXFILE directive, modified by the BINARY option, produces a binary output file starting at address 00000. The file includes the classes in the order defined by the OUTPUT directive. As with assembler segment classes, the file may exceed 64 KB even in SMALL model. You can produce output files in a variety of formats for special purposes; if you're actually burning EPROMs, LOCATE's various hex options will come in handy.

The end result of all this machination is the FFTS.PMO disk file that PMLoader reads and copies to address 00100000. As you saw last month, the current version of PMLoader can handle a file up to 64 KB. The code this month fits neatly in an 8-KB file, giving us plenty of room for growth.

FILLING THE TABLES

Figure 1 shows the storage layout used by FFTS starting at the 1-MB line. The disk file image occupies the first 64-KB block, with the remaining storage defined by the GDT descriptors we are about to create.

The first step clears storage from 00110000 to the end of RAM. Recall that PMLoader set GDT[9] to a data segment covering all of RAM above 1 MB. The code in Listing 4 converts that descriptor's segment-limit field into a count that writes up to 15 MB of zeros with a single REP STOS instruction. No 64-KB limits here!

Next, we copy PMLoader's GDT to address 001100000 which sets the CPU's GDT register to the new GDT. This is safer than trying to create a whole new GDT from scratch; if you get something wrong, the old GDT

C COMPILERS
CROSS ASSEMBLERS
DEBUGGERS

68HC08 8051/52
6809 8080/8085
68HCII 8086/186
68HC16 8096/196

Low Cost!! PC based cross development packages which include EVERYTHING you need to develop C and assembly language software for your choice of CPU.

- MICRO-C compiler, optimizer, and related utilities.
- Cross Assembler and related utilities.
- Hand coded (efficient ASM) standard library (source mcluded).
- Resident monitor/debugger< source included)*
- Includes text editor, telecomm software and many other utilities.

*68HC08 and 68HC16 kits do not include monitor/debugger.

Each Kit: $99.95 + s&h (please specify CPU)

SPECIAL!! Super *Developer's Kit*
Includes all 8 kits above, plus additional assemblers for 6800, 6801/6803, and 6502. *Reg. $400.00 NOW $300.00*
(Ask for SDK Special)

DDS
Dunfield Development Systems
P.O. Box 31044 Nepean, Ont. K2B8S8
CANADA
Tel/BBS: 613-256-5820 Fax: 613-256-5821

Protected Mode Stress?



Ease into Protected Mode with pmEasy™

pmEasy is a complete protected mode environment for embedded systems. It initiates protected mode and provides an application loader, trap handler, error handler, memory manager, debugger support, screen writes and more. *pmEasy* is integrated with low-cost 16- and 32-bit development tools from Microsoft, Borland, Periscope, and others.

Why **struggle developing your own protected mode environment?** *pmEasy* lets you focus on your application.

pmEasy16 or 32 **$495** including source code

BUY AND TRY 30 day money-back guarantee

MICRO DIGITAL INC **1-800-366-2491**
Developer of SMX
Cypress, CA, USA FAX 714-891-2363 VISA, MC, AMEX

#127

188SBC

Use Turbo or MS 'C'
Intel 80C188XL
Two 1 meg Flash/ ROM sockets
Four battery backed, 1 meg RAM
16 channel, 12 or 16 bit A/D
8 channel, 12 bit D/A
2 RS-232/485 serial, 1 parallel
24 bits of opto rack compatible I/O
20 hits of digital I/O
Real-time clock
Interrupt and DMA controller
8 bit, PC/104 expansion ISA bus
Power on the quiet, 4 layer board is provided by a switcher with watchdog and power fall Interrupt circuitry.
The 188SBC is available with Extended Interface Emulation of I/O - a Xilinx Field Programmable Gate Array and a breadboard area. Define and design nearly any extra Interface you need - we'll help! 188SBC prices start at $299.
Call right now for a brochure!

552SBC

The 80C552 is an 8051 with:
8 ch. 10 bit A/D **2 PWM outputs**
Cap/cmp registers **16 I/O lines**
RS-232 port **Watchdog**
We've made the 552SBC by adding:
2-RS-232/485 multi-drop ports
24 more I/O **Real-time Clock**
EEPROM **3-RAM & 1-ROM**
Battery Backup **Power Regulation**
Power Fail Int. **Expansion Bus**
Start with the Development Board - all the peripherals, power supply, manual and a debug monitor for only $349. Download your code and debug it right on this SBC. Or ken use OEM boards from **$149.**

True Low-cost In-circuit Emulation

The DrylCE Plus is a low-cost alternative to conventional ICE products. Load, single step, Interrogate, disasm, execute to breakpoint. Only **$448** with a pod. For the 805 1 family, including Philips and Siemens derivatives. Call for brochure!

8031SBC as **low** as $49

Call for your custom product needs. Quick Response.

VISA MasterCard

HITECH Equipment Corp.
9400 Activity Road
San Diego, CA 92126
[Fax: (619) 530-1458]
Since 1983
(619) 566-1892
70662.1241 @ compuserve.com

#130

values are still in place and should catch the problem. Listing 5 shows how to use the old GDT's data-segment alias to derive the byte count. The remaining entries in the new GDT are all zero and will trigger a protection fault should a program load them into a segment register.

The old GDT was just large enough to hold the few descriptors we needed, while the new GDT has 8 192 descriptors (mostly null] occupying 64 KB. FFTS uses several blocks of descriptors for system calls and other special functions, so we may as well allocate the storage now and be done with it. Of course, you need not be so profligate in your application because the CPU will trap any access beyond the end of the GDT.

The code then aims the stack descriptor at the new stack area, updates the code descriptor with the actual size of the code segment, creates an IDT aimed at the new unexpected-interrupt handler, and fills in the few remaining GDT entries we need to get started. All of the GDT and IDT entries are accessed using the data-segment descriptor set up by PMLoader.

The only tricky part of this process is calculating the starting address and length of the segments. Listing 6 shows one technique applied to the _prot. const segment, which is located just after the code segment in the disk image. Unlike the initialized-data segment, these values need not be copied elsewhere because they can't be changed!

Recall that the linker uses the segment name to combine parts of a segment that appear in separate source files. The STARTUP .ASM file is linked first, putting its code, variables, and constants appearing in it at the lowest offsets in their respective segments. Listing 6a shows the beginning of the _protconst segment, marked by a simple ASCII string to make it stand out in a storage dump.

FINAL.ASM, as the name suggests, is linked last to place its values at the end of each segment. Listing 6b shows the tail of the code and_protconst segments. Because these sections don't define any storage, the linker doesn't extend the previous segment, and label offsets are the actual segment lengths.

The chunk of code in Listing 6c loads the _protconst descriptor into the new GDT. There are three key fields: limit, base address, and access bits. I discussed the descriptor structure in *CAJ* 49; refer to that column for more details.

The segment limit is the last valid offset in the segment (when the G bit is zero anyway), which is just PM - ConstLength – 1. Only thelow-order 16 bits are useful for segments shorter than 64 KB, placing the offset well within the Seg Limit field's 20 bits.

The segment starts at the next paragraph boundary after the end of the code segment. The PMCodeLength label provides the exact code segment length, which is rounded up to the next multiple of 16, added to the load address, and then sliced up to fit the three sections of the base address field.

The Access field determines the type of segment and whether write accesses are allowed for data segments. The _prot. const segment must be read-only, implying that its ReadWrite bit must be zero. Note that this isn't absolute protection as you can access those same bytes using an overlapping segment with its ReadWrite bit turned on. At least you can't inadvertently clobber them through the _protconst descriptor.

Setting up the FFTS data segment is similar, with the added step of copying the initial values from the file image to the new segment. Listing 7 has the few lines of code needed for this. Note that the starting address includes two rounded-up segment lengths. The destination offset is zero, of course, because the first byte of data was defined at the beginning of the segment.

After loading the GDT and IDT, copying the initial data values, and aiming the segment registers at the new segments, the startup code branches to what will become the FFTS kernel. As the kernel becomes more complex, we'll need a few more startup functions and suchlike. In any event, you've got enough now to support truly nontrivial programs!

You can also calculate the seg-

ment locations using the real-mode segment values, but I'll save that for a later column when this stuff isn't quite so new. Hint: as you saw earlier, converting a real-mode segment:offset address into a PM 32-bit linear address is quite easy. If you put a label at the very start of the segment, the offset might be zero.. .. Or, it might not, which is why I'm punting it for now.

The kernel code initializes the serial port in polled mode and spits out a welcoming message before entering the spin loop. You should see a blizzard of activity on the parallel port LEDs tracking the code's progress through the PMLoader and 32-bit PM startup code, then a message on the serial port from the kernel, and finally a conspicuous blinking pattern on the parallel port along with an ascending count on the FDB's LEDs.

The serial port message comes from the _protconst segment and the count should begin at 55 hex because I used an initialized variable. If the text is garbled or the count starts at zero, the GDT segment values are probably incorrect, although "that can't possibly happen" here, right?

The serial ports will run in polled mode for the next few columns as we accrete more kernel functions and hardware support, then switch to interrupts when we need them for multitasking. Debugging is a lot easier with readable messages instead of blinking LEDs. Nonetheless, those blinking dots carried us quite a distance on this expedition.

You can see the protected land from here!

RELEASE NOTES

The code this month reflects the increasing complexity of the FFTS kernel. There are several ASM files with corresponding INCs defining their EXTRN procedures and variables as well as overall INC files holding global definitions. The MAKEFILE ties this all together, so you should be able to rebuild FFTS.PMO in a single step.

The Circuit Cellar BBS has a LOCATE.EXE file that originally accompanied an article in ***Dr. Dobb's Journal*** written by Rick Naro. He now runs Paradigm Systems, which produces the LOCATE utility I've been using. Although I haven't run this code through the BBS version of LOCATE, the family resemblance is clear.. . *You* get the complete source code, so you can tinker whatever improvements you think are warranted.

Next month, we'll add character output to the Firmware Development Board's Graphic LCD Interface and a standard VGA display. I'll also tell the chilling mystery story "The Case Of The Capital T." ❑

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.

413 Very Useful
414 Moderately Useful
415 Not Useful

▶Good Stuff◀

Bar Code Sensor
Battery Controllers
Clock/Calendars
Digital Power Drivers
DTMF & Phone Interfaces
Firmware Furnace Widgets
HCS-II Hard-to-find Parts
I²C Bus ICs
IR LEDs & Photodiodes
IR Data Link Parts
IR Remote Control
Laser Diode Controllers
Linear Hall Effect Sensor
Multiplexers & Crosspoints
Power Op Amp
Remote Temperature Sensor
Stepper Motor Drivers
Watchdogs & Power Monitors
8051 Information
and more!

Use a soldering iron? Get the parts!

UPS: Ground/2nd day $6.50/9.00 to 48 US states, COD add $4.50. PO Boxes and Canadian addresses: $6 for USPS mail. Check, MO, or COD only; no credit cards, no open POs. NC residents add 6% sales tax. Quantity discounts start at five parts. Data sheets included with all parts.

Call/write/FAX for *seriously* tempting catalog...

Pure Unobtainium

▶ Your unusual parts source ◀

13109 Old Creedmoor Road Raleigh NC 27613-7421
FAX/voice (919) 676-4525

The *only* 8051/52 BASIC compiler that is 100 % BASIC 52 Compatible *and* has full floating point, integer, byte & bit variables.

- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 805 1/8031 or 8052/8032 CPUs*
- *Compatible with any RAM or ROM memory mapping*
- *Runs up to 50 times faster than the MCS BASIC-52 interpreter.*
- *Includes Binary Technology's SXA51 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51 $ 295.***

508-369-9556
FAX 508-369-9549

❑

Binary Technology, Inc.
P.O. Box 541 • Carlisle, MA 0 1741

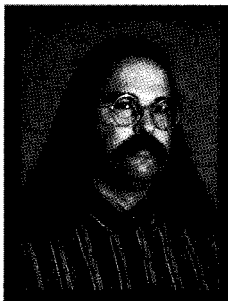
#131

FROM THE BENCH

Jeff Bachiochi

Does Anyone Have the Time?

A Comparison of Real-time Clocks



Every chip manufacturer seems to feel they can do the other guy one better when it comes to clock-calendar chips. Jeff takes a quick survey of the market and compares what's available today.

Does anybody know what time it is!
Does anybody really care!
(Chicago Transit Authority)

"So, what's with all these marks on the wall?"

"Those? That's the number of nights since the moon was last full-28 nights or what I like to call a moonth"

"And those?"

"These marks number the nights since last harvest. Here it is four seasons later, and there are over 300 nights."

"That's odd. If we spend 1 moonth in each of the twelve constellations, that's 336 days."

"336? Yes, that seems about right. Our year must have 336 days."

It didn't take the ancient magician priests of Babylon and Chaldea long to figure out that this perfection was flawed. Even increasing the month to 30 days had its inaccuracies. Only then at 360 days (30 x 12), the year needed an extra month once every six years.

This level of accuracy was pretty good considering the tools available then. Today we think of a year as 365 days or, to be more precise, 365¼ days. But, even with that extra day we add every four years, the calendar doesn't come out even because a year is more accurately defined as 365 days, 5 hours, 48 minutes, and 46 seconds. And, what about that fraction of a second.. .? Who's counting?

The day was divided into two parts (dark and light), each half having 12 hours (12 pops up again from the 12 seasonal constellations) or one rotation of the hour hand on the clock's face. For years, the hour hand was all that was used (or needed), but as technology progressed, time was broken down into smaller fragments: 60, being a powerful number with an ability to be divided by 2, 3, 4, 5, 6, 10, 12 (ah, there's that 12 again), 15, 20, and 30, was used as a divisor.

So, why do we now divide seconds by tenths, hundredths, and thousandths? I'm surprised that anything so unmetric could have its origin in Eurasia. I'm for the metric system. I like shifting the decimal point around rather than having to divide (or multiply) by some constant just to move between units of measure. But, with the year consisting of 365¼ days (minus 11 minutes and 14 seconds), it's clear the metric system wasn't invented by God.

All the same, why couldn't we have 1000 hours to the day! Then each milliday (about 3 minutes) could have 1000 microdays (about ⅕ second). What we could accomplish with 1000-hour days would be staggering. Ah, wait, that puts the work week at about 1500 hours. On second thought, never mind.

REAL-TIME CLOCKS, CALENDARS, AND THE CPU

While a CPU has the ability to count known quantities of time (i.e., oscillator periods) and calculate the passing of seconds, minutes, hours, days, months, and years, there are usually far more important issues at hand. Removing the burden of counting "tics" isn't free, but today, what is? Fortunately, neither the financial nor real-estate costs are extreme.

Although interfacing techniques widely differ, dedicated clock and calendar circuitry is basically the same. The heart of the RTC (real-time clock) pumps at 32.768 kHz, a nice round 15-bit number. This timebase is divided into small increments (i.e., 1 s or smaller). The one-second tics are

accumulated until they roll over into the next digit's place, at which time they return to zero and continue accumulating. When the tens-of-seconds register rolls over to 6 (actually, back to 0), the minutes register is incremented. And on, and on until the last register, usually the tens-of-years register, is updated.

The RTCs often have additional functions associated with them. Periodic or alarm interrupts can signal a CPU of an elapsed time condition. Hours may be held in the standard 12-hour A.M./P.M. format or in military 24-hour format. The year register increases the length of February to 29 days for automatic leap-year recognition. Sophistication levels reach a pinnacle in providing an automatic adjustment to daylight saving time.

Not all these functions are included in every model, so you must decide what functions you require. Table 1 lists a number of RTC manufacturers along with their functions, size, and interface type.

Let's explore each by interface type.

PC/AT STYLE RTC

One of the first expansion boards I added to my original PC was a clock-calendar card. No longer would I have to answer the time and date prompts which popped up with each DOS cold boot. They were such a pain that most of the files created back then had a 1/1/80 default creation date. Today's machines come with the time and date set in a battery-backed RTC as well as DOS, Windows, and who knows what else preinstalled.

The PC/AT standard clock-calendar chip was the Motorola MC146818. Dallas Semiconductor and Benchmarq both make drop-in replacements for that old workhorse. One of the most unique features of the Motorola device was the pin-configurable interface. Pin 1 defined the interface type as Motorola, which uses a RD/* WR pin with a data strobe (DS), or as Intel, which uses separate *RD and * WR strobes. Hard to imagine a manufacturer designing with that kind of common sense, isn't it!

Part #	Manuf.	Features	Size	Interface	Comp.
BQ3287	Benchmarq	Time Date Alarm 12/24 format Daylight saving 1141242 bytes NVRAM Internal crystal & lithium battery	24-pin DIP	Motorola or Intel bus	MC146818
804287	Benchmarq	Time Date Alarm 12/24 format Daylight saving 1141242 bytes NVRAM Internal crystal & lithium battery	24-pin DIP	Intel bus	DS1287
BQ3285	Benchmarq	Time Date Alarm 12/24 format Daylight saving 1141242 bytes NVRAM	24-pin DIP	Motorola or Intel bus	DS1285
BQ4285	Benchmarq	Time Date Alarm 12/24 format Daylight saving 1141242 bytes NVRAM	24-pin DIP	Intel bus	DS1285
DS1202	Dallas	Time Date 12/24 format	8-pin DIP	3-bit clocked serial	
DS1215	Dallas	Time Date 12/24 format	16-pin DIP	Phantom clock	
DS1243 DS1244	Dallas	Time Date 12/24 format	28-pin DIP	JEDEC footprint with phantom clock	
DS1248	Dallas	Time Date 12/24 format	32-pin DIP	JEDEC footprint with phantom clock	
DS1283 DS1284 DS1286	Dallas	Time Date Alarm 12/24 format	28-pin DIP	JEDEC footprint	
DS1285 DS12885	Dallas	Time Date Alarm 12/24 format Daylight saving 50/114 bytes RAM	28-pin DIP	Motorola or Intel bus	
DS1287 DS12887	Dallas	Time Date Alarm 12/24 format Daylight saving 50/114 bytes RAM Internal crystal & lithium battery	24-pin DIP	Motorola or Intel bus	MC146818
DS1642	Dallas	Time (24 format) Date 2K × RAM Internal crystal & lithium battery	24-pin DIP	JEDEC footprint	MK48T02
DS1643	Dallas	Time (24 format) Date 8K x RAM Internal crystal & lithium battery	28-pin DIP	JEDEC footprint	MK48T08

(continued)

Table I--Numerous *manufacturers make whole lines of clock-calendar chips with various shapes, sizes, and feature lists. In some cases, they are a/so plug compatible with other popular chips.*

Part #	Manuf.	Features	Size	Interface	Comp.
NJV6355	JRC	Time Date Low-voltage alarm	8-pin DIP	4-bit (clocked serial)	
MC146818	Motorola	Time Date Alarm 12/24 format Daylight saving 50 bytes RAM	24-pin DIP	Motorola or Intel bus	
MM581 67	National	Time Date (DD-WW-MM) Alarm	24-pin DIP	4-bit address and data	
MM58174	National	Time Date (DD-WW-MM) Periodic Timer	16-pin DIP	4-bit address and data	
MSM5832RS	Oki	Time Date 12/24 format	18-pin	4-bit address and data	
MSM58321 RS	Oki	Time Date 12/24 format	16-pin	4-bit multiplexed address and data	
MSM624BRS	Oki	Time Date 12/24 format	18-pin	4-bit address and data	
PCF8583	Philips	Time Date (month-day-dow) Alarm Event counter 12/24 format	8-pin DIP	I²C	
RP5C01	Ricoh	Time Date 12/24 format Alarm	18-pin DIP	4-bit address and data	
RP5C62	Ricoh	Time Date 12/24 format Alarm Periodic timer	18-pin DIP	4-bit address and data	
MK48T02	Thomson	Time (24 format) Date 2K × 8 RAM Internal crystal & lithium battery	24-pin DIP	JEDEC footprint	
MK48T08	Thomson	Time (24 format) Date 2K × 8 RAM Internal crystal & lithium battery	28-pin DIP	JEDEC footprint	
TC8250	Toshiba	Time (24 format) Date	16-pin DIP	4-bit multiplexed address and data	

Table I-continued

Ten addressable registers hold time and date information in binary or BCD format. These registers include seconds, minutes, hours, day of week (dow), day, month, and year along with seconds, minutes, and hours alarm registers. The alarm registers compare to the active registers and can institute an interrupt on a proper match.

The active registers are updated from the divided timebase. One of three crystals can be used as a timebase: 4.194 MHz, 1.049 MHz, or 32.768 kHz. Four more registers, A, B, C, and D, are used to indicate functions like crystal selection, periodic interrupt rates (none-30.5 µs), binary or BCD data format, 12/24-hour mode, and daylight-saving-time enable.

Interrupt source enables and polling flags are also included. I don't know of any PC that actually uses the daylight-saving-time function. However, when enabled an hour is added at 1:59:59 A.M. on the last Sunday in April and an hour is subtracted at 1:59:59 A.M. on the last Sunday in October.

In addition to the clock-calendar function, 50 bytes of NVRAM is available to the system (or user). On the PC, this RAM holds system configuration information (this is the "CMOS" configuration RAM). While keeping pin compatibility with the Motorola device, Dallas and Benchmarq have included extended versions in their product lineup. Up to a couple of hundred extra RAM locations as well as an internal quartz crystal and ten-year lithium power source are available in various combinations.

At this point in our RTC overview, Dallas and Benchmarq have taken slightly differing tacks. Benchmarq chooses to remain pin-count compatible and allow the clock's battery backup circuitry to also control and protect an external SRAM. This retains the Motorola clock-calendar access, yet greatly extends NVRAM size. Dallas, on the other hand, chooses to increase the pin count, adding the extra NVRAM within the clock-calendar IC. What you end up with, although functionally compatible with the Motorola MC146818, is no longer a drop-in replacement.

TIME TO LEAVE PC LAND

You say you don't care about Motorola compatibility or even PCs for that matter? Don't leave now—there is plenty more to tell.

Let's stick with the NVRAM idea a minute, however, since everyone uses or is familiar with it. Being a very practical company, SGS-Thomson understood that many systems use SRAM, and that adding a clock calendar to a JEDEC-standard SRAM would be a hot item. They accomplished this by setting aside the top eight RAM locations for the seconds, minutes, hours, dow, day, month, year, and control register.

The control register performs three functions: write-enable, read-enable, and count-trim adjustment. The typical clock calendar can be off ±1–2 minutes per month. Adjustments are made in the clock circuitry to

counteract this. The SGS-Thomson part can be calibrated by adjusting the counts (either adding or subtracting "tics") over a 64-minute period. If your time error falls within typical parameters, this adjustment refines resolution to ±5 seconds per month.

Dallas, recognizing a good thing, second sources the SGS-Thomson clock calendar, but also tosses the phantom timekeeper-their own variation-into the ring. This device, the DS **1215,** contains NVRAM control and clock-calendar registers which remain transparent until a particular 64-bit serial access sequence has been recognized. The sequence is composed of writes to any NVRAM location protected by the part.

Once the pattern has been identified, the DS1215's ● CE is rerouted from the SRAM to the timekeeper for the next 64 accesses. After 64 writes (updating the clock-calendar registers) or 64 reads (reading the new time and date), accesses revert back to the SRAM for continued normal operation.

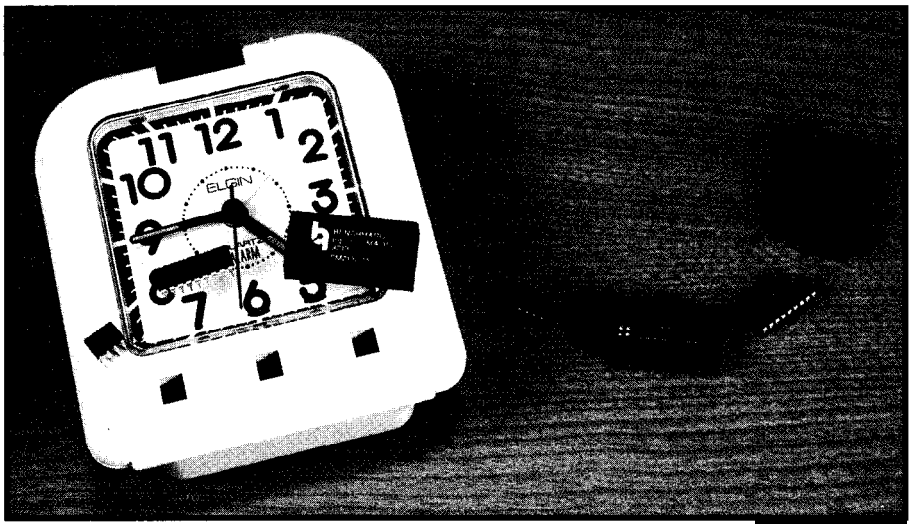


Photo I--Time *marches on with a* parade *of clock-calendar* chips *of all sizes high-stepping across the o/d mechanical workhorse.*

Like Benchmarq's drop-in JEDEC-footprint chip with NVRAM, clock, and calendar, Dallas provides JEDEC footprint NVRAM with the DS1215 phantom clock-calendar built in. It's a bit more difficult to access, but the DS1215 has the advantage of being able to be used with either RAM or ROM. This is a neat trick for writing to a read-only memory device.

NO FRILLS, NO OPTIONS, JUST REAL TIMEKEEPING

From here, we move out of the land of confusion and into the no-nonsense world of cut-throat time-

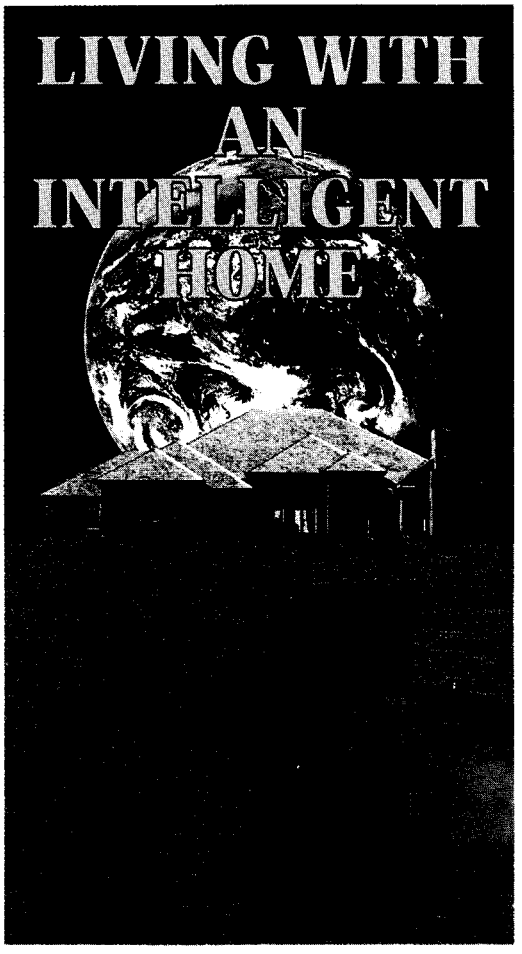
Find out *how* you can add intelligence to any home, at a cost that's within your budget.

LIVING WITH AN INTELLIGENT HOME
will change the way you live.

Written by David *Gaddis*, author of *Understanding & Installing Home Systems*

This VHS video cassette *retails for $24.95 plus $5* s&h. It is being o&red to Computer Applications Journal *subscribers for* only *$17.95* plus $4 *s&h (U.S.)*

ORDER TODAY! Don't *let* this exiting technology *opportunity* pass you by!



Take a tour through modern technology in today's home...

Circuit *Cellar*, Inc.
4 *Park St.* • Vernon, *CT* 06066
***Tel:* (203) 8752751**
***Fax* (203) 872-2204**

keeping. Here every manufacturer has their own ideas on just how to package the time and date. Device sizes range from 24- to 8-pin DIPs.

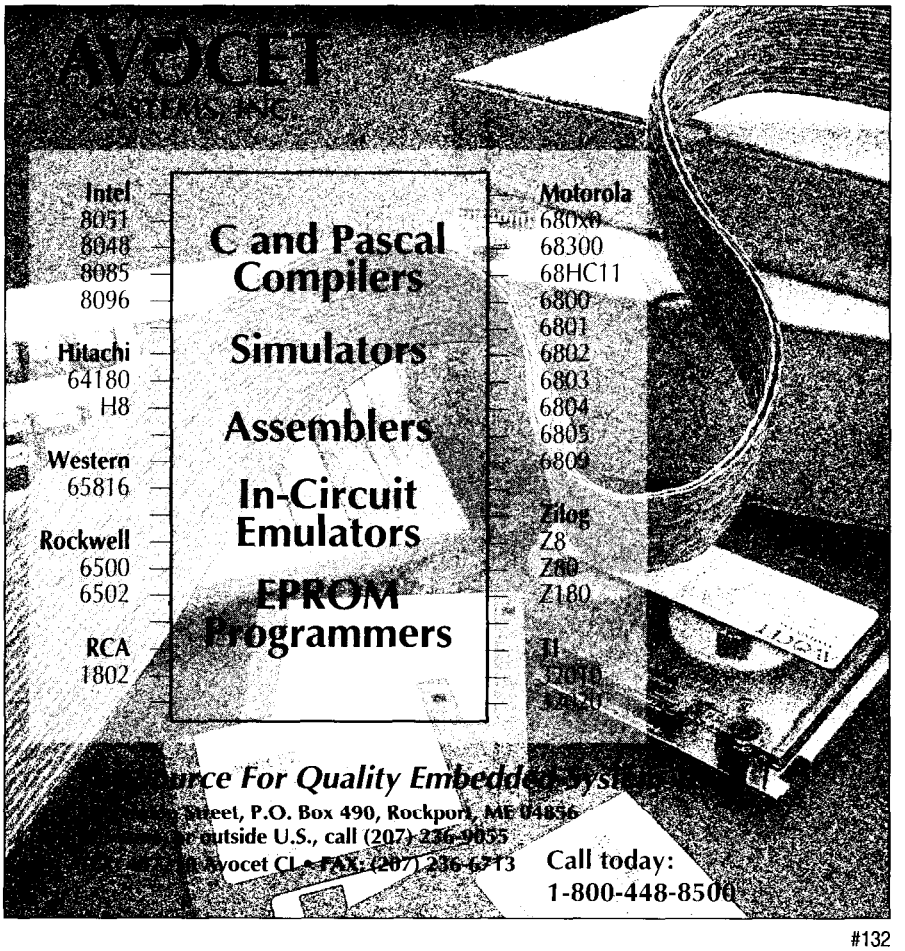
Starting with the largest, National Semiconductor's MM58167 is a bit unique. It keeps track of thousandths of a second up to 12 months and has a full duplicate set of registers for alarm comparisons. This is the last device to use an 8-bit data path. And, with the introduction of the nybble transfer comes a smaller outline package.

National's 16-pin MM581 74 did away with the alarm compare registers, but added a one-time output or periodic output which is programmable to 0.5, 5, or 60 seconds.

Ricoh's 18-pin RP5C01 includes seconds through years registers and can alarm on comparisons of the minutes through days registers. This device uses a bank-switching technique to add a bank of comparison registers and two additional banks of 13 x 4 NVRAM for the user. A unique ±30-s adjustment pin is available which will zero to the nearest minute whenever the input is raised to a logic high.

The RP5C62 is also an 18-pin clock-calendar device. This device limits the banks to two, so it has no user NVRAM. The alarm comparisons are also limited to minutes and hours, and the ±30-s adjustment is controlled by an internal register instead of an external pin. An added periodic timer has its own output pin (separate from the alarm output) which can be used as a watchdog timer. Periods can be set from 4 to 562 ms. In watchdog mode, if you don't write a 1 to the TMR bit before the timeout of each period, a logic low will be output at *TMOUT. This low level can be used as a reset to the system's CPU.

Toshiba's RTC, the 16-pin TC8250P, uses a 4-bit multiplexed bus which requires an ALE signal to internally latch the 4 bits of address. This gives the TC8250 a few extra I/O pins. Toshiba makes use of these by providing a means of trickle-charging a rechargeable back-up battery while the device is fully powered by the system.



#132

A 4-kHz, 50% duty cycle, TTL output is provided separately from the periodic programmable TOUT (l-2048 Hz). Register protection is furnished by a KEY register which must have a 5 written to it before any other register can be modified. Furthermore, if brownout occurs, the KEY register is cleared to 0.

Oki's18-pin introduction, the MSM5832RS, is very straightforward. Thirteen registers hold all the time and date information. A hold input prevents the registers from updating while they are being accessed, but can cause loss of time if the hold input is held too long. The MSM58321RS reduced pin count to 16 (by requiring an ALE) and adds a BUSY output pin which indicates when the register updates are happening (once a second).

The final offering from Oki reverts back to an 18-pin device with the MSM6242BRS. Time and date are held within the first thirteen registers, and an additional three registers contain control bits and flags which previously were I/O pins. These registers also select a ±30-second error correction and periodic outputs of 1/64 second, 1 second, 1 minute, and 1 hour.

Please note that Oki is not the only manufacturer to experience potential operational violations while attempting to update time and date registers. Read data sheets carefully to determine if any accesses to the device may violate internal operations.

8-PIN DIP RTCS

Yup, we're down to the little guys now. Little in size, but not reduced in functions.

The first is from JRC (New Japan Radio). The NJU6355 requires three output bits (CE, I/*O, and CLK) and one I/O bit (DATA) in a clocked-serial format. Fifty-two bits of data are transferred into or out of the '6355 depending on the logic level of the I/*O line. BCD nybble format is sent in an LSB-first sequence which starts with the year and then the month, day, dow (only 1 BCD digit), hour, minute, and second.

The second entry is from Dallas. The DS1202 is a three-wire device requiring two output bits (• RST and

CLK) **and one** I/O bit (DATA). Clocked serial commands consist of a command byte and one or more data bytes. The command byte selects access from either the timekeeper or from the 24 bytes of NVRAM available to the user. It also selects which register will be read from or written to. You can access a single register or use a burst mode to access all time or RAM registers in one continuous stream. The time and date registers are similar in structure to Dallas's phantom clock-calendar chip, which includes provisions for either 12- or 24-hour formats. An additional register provides write protection for all clock and RAM registers.

The final offering is from Philips (Signetics). The PCF8583 is an I^2C bus component including a clock-calendar and 256 bytes of SRAM. The device requires one output bit (CLK) and one I/O bit (DATA] for communication. The PCF8583 responds to a fixed 7-bit address of 101000x (where the x is replaced with the logic level applied to the A0 address input). Time and date registers consist of hundredths of a second, seconds, minutes, hours, days (including 2 bits for the leap-year cycle), and months (including dow). Additional registers hold the same information for alarm comparisons. And, here's a strange application for you: if no crystal is used, the '8583 will count input pulses on OSC1 up to 999,999.

ACCURACY

We are used to perfect clocks. Plug-in clocks are based on the 60-Hz line frequency. As long as someone pays attention to the power grid's frequency over the period of a day (and they do), we have perfect time. RTCs are not based on the always-accurate 60-Hz source. Instead, most are based on a 32.768-kHz crystal. Some chips incorporate the crystal internally while most require an external crystal and sometimes one or two caps.

The actual operating frequency (@25°C) can be shifted slightly by changing the value of the external capacitor(s) by a few picofarads. Crystal tolerances are in the ±20 PPM. Since there are 2.592 million (60 x 60 x 24 x 30) seconds in a month, unadjusted extremes may be ±51.84 (20 x 2.592) seconds per month for just the crystal tolerance.

Let's say you tweaked that error out to zero seconds per month. Other factors still have an effect on accuracy. When the RTC operating at 5 V goes into battery backup mode, the frequency typically shifts ±4 PPM, but can shift as much as ±15 PPM in severe temperature extremes. This large shift must be blamed to some extent on the capacitor design, but it almost entirely depends on the crystal's characteristics (even crystal aging can have a 2-PPM effect).

The weak link in overall accuracy lies with the crystal's frequency stability over temperature extremes. Although considered zero at 25°C, at the extremes of the commercial temperature range (-30 to +70°C), the deviation can reach -100 PPM. Luckily, for most of us, our computers are housed in a comfort zone which limits the deviation to about -10 PPM. The deviations do add up and can make us (or our automated procedures) off by a couple of minutes each month.

When using an RTC with external components, pay special attention to PCB layout. Keep the crystal and any associated capacitors as close as possible to the input pins. Don't run any traces carrying fast signals underneath any RTC components. Always use a ground plane under the crystal to isolate the capacitance coupling of any nearby high-speed signals. And, remember to bypass the RTC's power and ground with a ceramic 0.1 -µF capacitor.

One last point I wish to cover is power consumption. Since most RTCs are of CMOS construction, they require little operating current. Standby operating currents of 0.5-15 µA are typical at back-up voltages of 2 V. Modules with internal batteries to keep clock-calendars and NVRAM alive during standby are designed for a worst case of 10 years without power-thanks to today's lithium cells.

You can't create the perfect timepiece, but you can design in an RTC which will bring you acceptable results. Just remember to read and compare the specs carefully and ask for application notes. ❑

***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

Benchmarq Microelectronics, Inc.
2611 West Grove Dr., Ste. 109
Carrollton, TX 75006
(214) 407-0011

Dallas Semiconductor
4401 South Beltwood Pkwy.
Dallas, TX 75244-3292
(214) 450 0470

JRC Corp.
340-B East Middlefield Rd.
Mountain View, CA 94043
(415) 961-3901

National Semiconductor Corp.
2900 Semiconductor Dr.
Santa Clara, CA 95052-8090
(800) 272-9959

Oki Semiconductor
785 North Mary Ave.
Sunnyvale, CA 94086
(408) 720-8940

Signetics Corp.
811 East Arques Ave.
Sunnyvale, CA 94088-3409
(408) 991-3737

Ricoh
2071 Concourse Dr.
San Jose, CA 95131
(408) 432-8800

SGS-Thomson
1000 East Bell Rd.
Phoenix, AZ 85022
(602) 867-6100

Toshiba
15621 Redhill Ave., Ste. 205
Tustin, CA 92680
(714) 259-0368

IRS

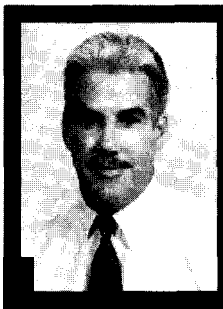
416 Very Useful
417 Moderately Useful
418 Not Useful

SILICON UPDATE

Tom Cantrell

Hot Chips VI

Image Compression, 3D, and RISC



Tom turns up the heat with a visit to the latest Hot Chips show. Video and graphics are the hot topics these days, as shown by the latest round of silicon.

Getting up at 6:30 A.M. on a Sunday morning to attend a technical seminar may sound crazy. Nevertheless, duty calls and that's why your humble reporter was on the road early to Hot Chips VI, held August 14-16 at Stanford University. The good news is there wasn't much traffic-likely due to the fact that people with any sense were still at home in bed.

Fortunately, the early bird tutorial "Algorithms and Hardware for Video Compression," presented by Stanford Professor Teresa H. Meng, was truly interesting (admittedly, fueled with copious shots of high-octane coffee). Her presentation was notably not an arcane math exposition, but instead focused on the bottom-line impact of various schemes in terms of price and performance while recognizing the marketing realities that ultimately prevail over bits-and-bytes religious wars.

Those of you who've been on another planet for the last few years may need to be reminded that video compression is, and will remain, a very hot topic. Notably, it is a critical enabling technology behind all sorts of next generation gadgets-everything from the "information superhighway" to HDTV, video games, multimedia PCs, videoconferencing and on and on.

The problem is simple-so many bits, so little bandwidth. Consider that a 512 x 512 x 24–bit image is 768 KB. Worse-motion video requires delivery at thirty frames per second, calling for a whopping 20+ MBps. Of course, to meet diverse consumer demand for everything from the "Laverne and Shirley" channel to TV gambling calls for 500 channels, or a ludicrous 10 GBps.

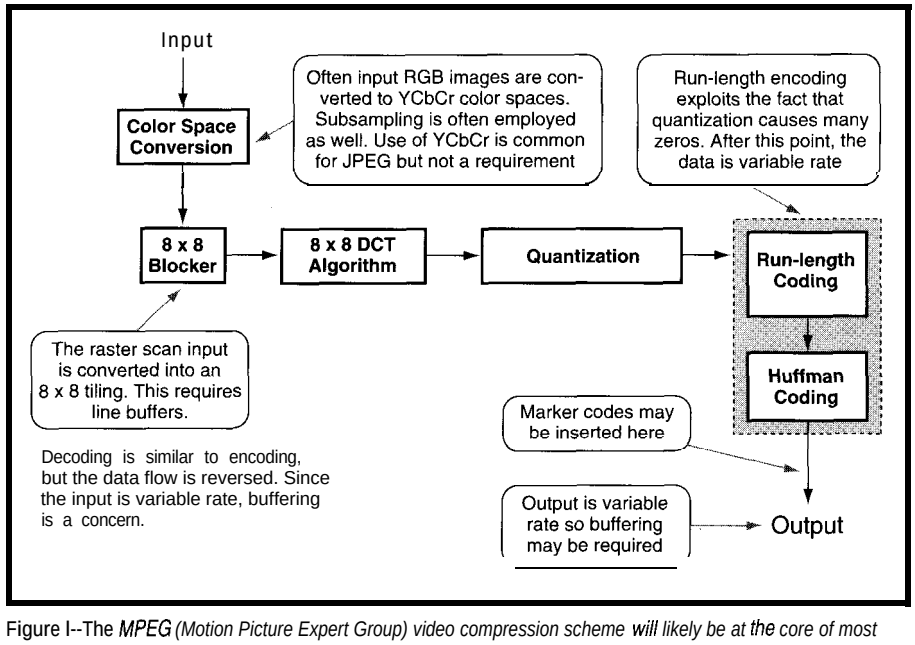


Figure I--The *MPEG (Motion Picture Expert Group) video compression scheme will likely be at the core of most upcoming PC-based multimedia and game applications.*

Yeah, try shoveling that in or out of your PC, much less over a phone wire.

AND THE WINNER IS...

There are a variety of compression algorithms, each with strengths and weaknesses that better suit it for certain tasks. Besides the obvious feature of compression ratio, the alternatives are differentiated by characteristics such as compression versus decompression symmetry, memory requirements, error tolerance, and the size, speed, and power of the requisite LSI.

While various applications (especially closed ones in which the compression is internal to the box or system) may exploit a particularly optimal algorithm, a marketing reality is that MPEG (Motion Picture Expert Group) is going to dominate thanks to blessing by a variety of big guns.

MPEG I serves as the basis for video CD and will thus be at the core of most PC-based multimedia and game applications. Meanwhile MPEG II has been chosen by the "Grand Alliance," a consortium of broadcasters and equipment providers, as the basis for the forthcoming digital HDTV.

Figure 1 shows the core sequence of MPEG processing. Note that the foil refers to JPEG (Joint Photographic Expert Group), which is a still (not motion) image-compression scheme. The point is, ignoring the issue of motion, JPEG and MPEG standards are based on the same coding scheme (i.e., a single frame of motion video can be coded as a still image).

Often overlooked is the first step-colorspace conversion-in which RGB data is converted to the so-called *YUV* format in which Y refers to luminance (brightness) and *U* and *V* to chrominance or color. However, color conversion shouldn't be ignored because it presents an opportunity for belt tightening and also-as the first step-can have a big impact on the subsequent outcome.

The compression opportunity of color conversion arises from the simple fact that the human eye is much more sensitive to luminance than chroma. Thus, some chroma information can be tossed without affecting our perception. This is accomplished by downsampling the *U* and *V* components so that a typical scheme has one chroma sample per four luminance samples (referred to as 4: 1: 1 sampling). Compared to a 4:4:4 sampling—voilà, instant 2: 1 compression right off the bat.

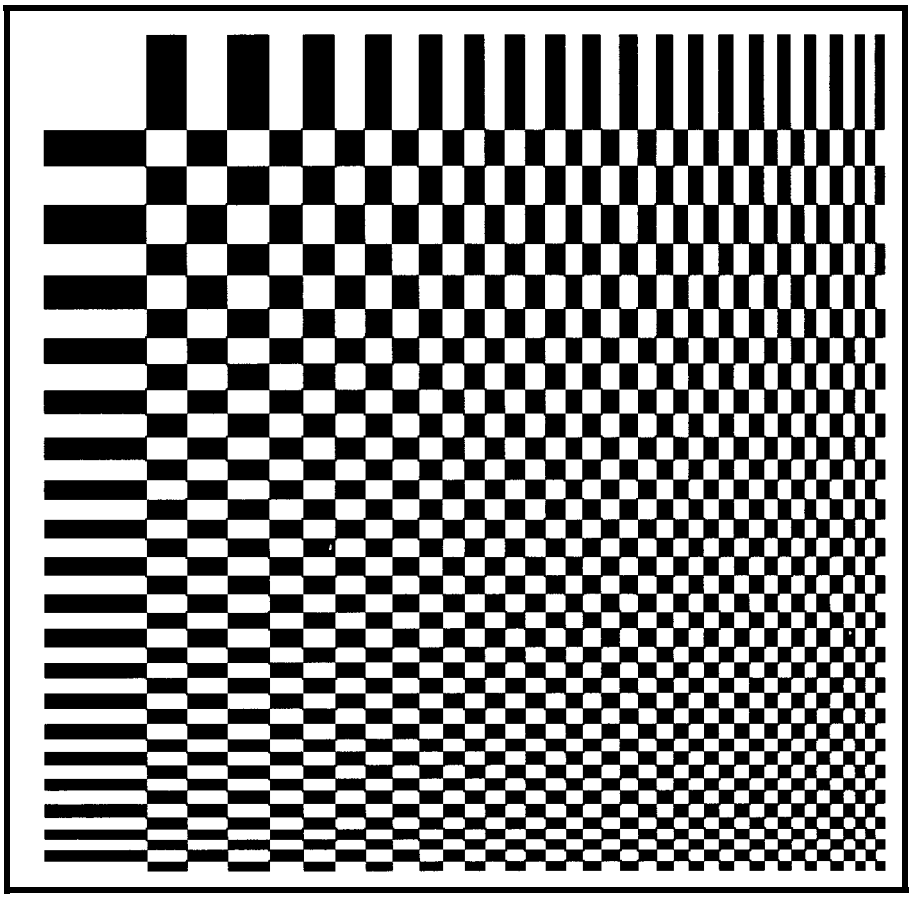


Figure 2-One *video compression method involves the DCT (discrete consine transform) to transform 8 x 8 blocks of pixels. The goal is to organize visual data by frequency with low frequency in the upper-left corner and high in the bottom-right.*

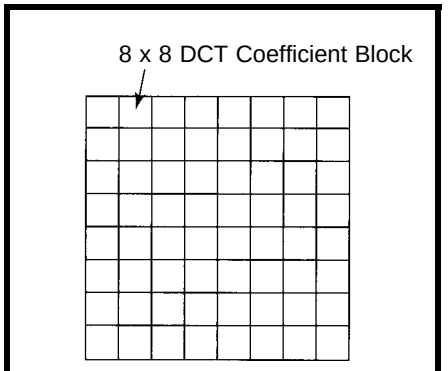
But beware.

You may remember from your DSP-101 class that the down- (and subsequent up-) sampling presents an opportunity for aliasing that may cause problematic artifacts depending on your source material. Thus, filtering plays a key role and is somewhat of a "black art" since it involves assumptions about the source-the "right" sampling and filter for a natural image may fall apart when faced with a cartoon.

Next comes the famous DCT (Discrete Cosine Transform) that transforms 8 x 8 (or sometimes 16 x **16)** blocks of pixels. Usually presented as complicated math gobbledegook, you can consider it the same as an FFT, except that the fundamental function is based on cosines instead of e. Even better, check out Figure 2, which makes it clear that the goal is to organize the visual data by frequency (horizontal and vertical) with low frequency in the upper-left corner and high in the bottom-right. Note that the DCT is performed separately on the Y, *U,* and *V* components.

Once again, exploiting psycho-visual phenomena (the eye sees the sharp edges, not the noise), a compression opportunity emerges.

The DCT itself doesn't shrink the data, but does put it in a form receptive to further crunching in the quantization step. As shown in Figure 3, the components of the matrices (Y and *UV*—note the difference in reflectivity vs. color perceptibility) are quantization step sizes (i.e., the number by which corresponding elements of the DCT transform matrix are divided).



Y Component Matrix

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	58	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Cb Cr Component Matrix

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Figure 3—*The DCT itself doesn't shrink the* **video data, but** *instead puts the quantization step sizes into matrices which are* **more** *conducive to compression.*

You should also know there is nothing sacred about these tables. They're basically derived from the "experts" sitting around the tube and saying, "That looks pretty good to me." Most notably, the tables may be scaled linearly to increase or decrease the compression ratio and/or nonlinearly to better suit a particular image.

Dividing by the big numbers in the lower right will lead to a lot of zeros. Better yet, scanning the matrix in a zig-zag fashion from top left to bottom right puts the zeros together, where they are handily dispatched to the big bit-bucket in the sky by simple RLL (Run-Length-Limited) coding.

The final step, Huffman coding, exploits the fact that many scenes contain large areas of repetitive data (i.e., single-colored objects of interest). Originally developed to compress text, the Huffman scheme creates a variable-length alphabet with shorter codes used for more probable symbols. As for the quantization tables, default Huffman tables are defined, but the standard allows for "application specific" tables to be used as well.

I can see that even this simple overview is consuming column space at a prodigious rate, so I'm going to have to cut corners on the "motion" issue.

It would seem simple enough to code each frame in the previous manner and be done with it (a technique known as *motion-/PEG).* But, noooo..

The greedy designers, ever in search of freebie bandwidth, recognized that often there is little variability from frame to frame (one of the best examples being the "talking heads" that litter the airwaves). They ended up defining three types of frames-intra (I), predicted (P), and interpolated (B)—to exploit the temporal correlation.

An intraframe is a fresh coding. Predicted frames rely on motion estimation by the source such that only a motion vector and difference block need be sent to construct a predicted frame from a previous intra (or even a prepredicted!) frame.

Predicted frames don't necessarily immediately follow an intra (or predicted) frame, so the gap is filled with interpolated frames. Not only can interpolation take place between intra and predicted frames, but the interpolation may take place in a forward or backward direction (Figure 4). Consider an opening door sequence in which the stuff behind the door can only be derived from the later (door open) and not the earlier (door closed) frame.

If you've got the idea that all this motion stuff is a horribly complicated computational nightmare, proceed to the head of the class. I suppose it must work-after all, the "experts" certainly must know what they're doing, right?

AND THE LOSER IS...

Anyway, now knowing enough about MPEG to be dangerous, we're fully qualified to move on to the "gripes and gotchas" section. Remember, complaining is somewhat futile (due to inevitable standardization), but it's still fun.

First of all, note that the lossy stages up through quantization are of fixed bandwidth (i.e., the amount and speed with which data is crunched is fixed). However, the subsequent lossless steps (RLL and Huffman coding) introduce variability into the data rate. Though well understood, it's still a pain to have to haul out the FIFOs, interrupts, and statistical guesswork a variable data rate implies.

Eliminating redundancy sounds good until you realize that TV would not have been possible without it. The fact is, the airwaves (and even analog cable) are hard pressed to deliver a perfect signal. Fortunately, all the redundancy in an uncompressed signal means you don't miss the big touchdown just because your neighbor's air conditioner kicks on.

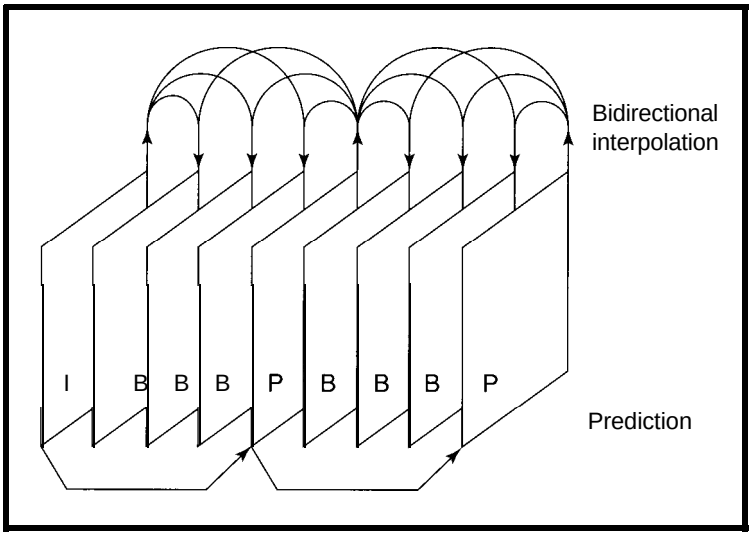


Figure 4—*Predicted frames rely on motion estimation by the source such fhaf on/y a motion vector and difference block need be sent to construct a predicted frame from a previous frame. Interpolation may fake p/ace in eifher a forward or a backward direction.*

Your eye-brain combo happily overlooks a transient tear, glitch, or snow.

However, losing even a single bit of MPEG-coded data (due to the elimination of redundancy, every bit counts) can cause quite a disaster (i.e., a frame of garbage). Better yet, contemplate frames predicted from and interpolated between garbage-smelly stuff, indeed.

Ironically, the solution called for is to reintroduce some redundancy in the form of error-correction code. Don't bother questioning the spending of transistors and cycles to take out redundancy and more transistors and cycles to put it back-only "experts" can understand these things.

Decoding is well defined, so one decoder should work pretty much as well as another. However, the encoding process is much more loosey-goosey, especially since so many facets of the algorithm-notably the quantization and Huffman tables as well as the decision whether to send I, P, or B frames-are affected by the type of source. There will be a big difference between good and bad encodings, with the former requiring much more compute power or even manual intervention (e.g., to explicitly intra-frame-code key frames such as scene changes). It makes you wonder, "Will the zillions of hours of old movies be carefully encoded, or just spit through a dumb encoder in real time?"

Finally, compression seems to encourage an annoying tendency to be too stingy when doling out bandwidth. I myself have suffered through more than my fair share of bad MPEG demos. Watch out when the snake oil salesman says he can deliver 100: 1 compression-he can deliver it, but you won't be able to watch it.

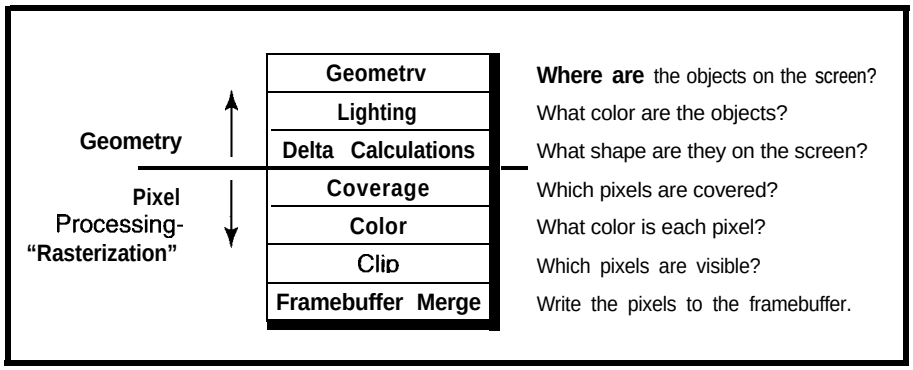
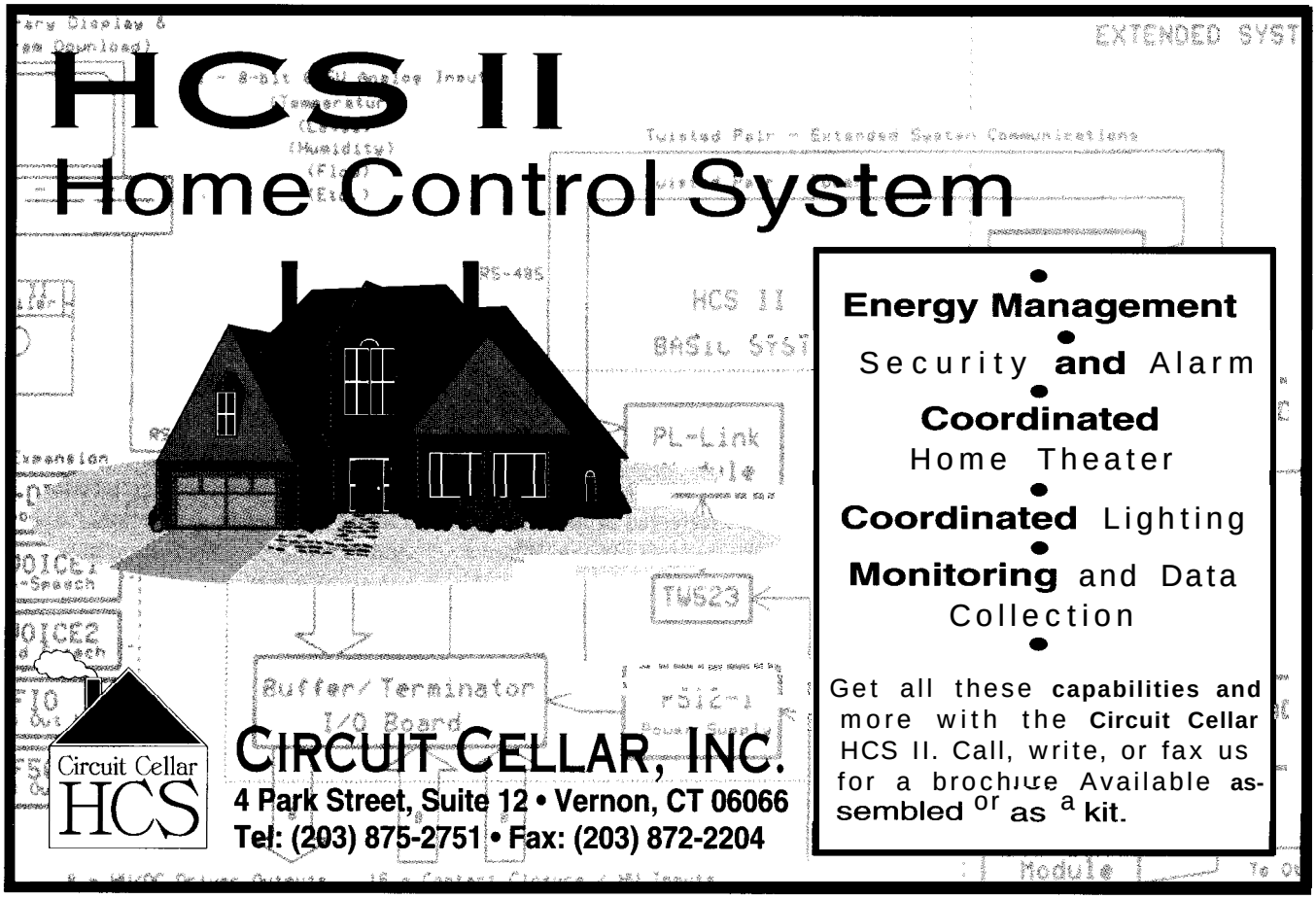


Figure 5—*The* field of three-dimensional graphics encompasses a series of *operations, roughly* categorized as geometry and *rasterization.*

3D OR BUST

Beyond video, 3D graphics are getting great attention. I must admit, it's a lot more fun watching a 3D graphics demo than reviewing the latest superduper CPU block diagram.



#133

Typically, 3D encompasses a series of operations, roughly categorized as geometry and rasterization as shown in Figure 5. The former, involving the projection of a 3D object onto a 2D screen, is largely a computational (trig) exercise while the latter is mainly pixel crunching including hidden line or surface removal (Z-buffer), clipping, antialiasing (smoothing the "jaggies"), color mixing, dithering, and so on.

Fast 3D isn't easy and thus has remained largely the province of the high-end workstation suppliers. The clear leader in the field is Silicon Graphics, whose MIPS-based workstations dominate the Hollywood special effects industry (e.g., Terminator, Jurassic Park, not to mention ever more synthetic commercials).

The topical question is whether 3D hardware will migrate from workstations onto PCs? One company that answers "yes" is the aptly named 3Dlabs. They, along with suppliers like S3, Cirrus, and S-MOS, are preparing to offer chipsets that can bring workstation-like imaging onto your desktop.

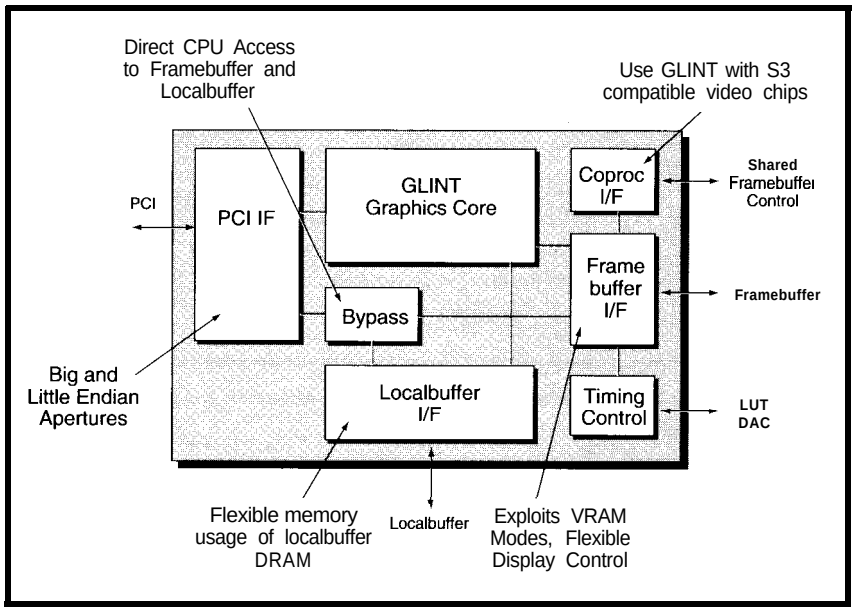


Figure 6—*One company frying to migrate 3D applications from workstations to PCs, 3Dlabs, promotes the philosophy that 3D is best partitioned between the host CPU and special hardware, namely their GLINT chip.*

3Dlabs promotes the philosophy that 3D is best partitioned between the ever more powerful host CPU and special hardware, namely their GLINT chip (Figure 6), with the former handling geometry and the latter, rasterization.

As an aside, 3Dlabs actually doesn't sell a chip. Instead, they sell a VHDL model which is easily modified and then synthesized for manufacturing in a particular fab. In their words, while "fabless" chip companies have been the trend, the next step may indeed be "chipless" chip companies.

The bad news is a 304-pin, 50-MHz, **1.1** -million-transistor chip won't be cheap. The good news is it will be cheaper tomorrow than today and presumably, at some point, cheap enough to be compelling.

A key issue is if and when software developers will drive 3D-enabled programs into the market. A critical factor is the adoption by Microsoft of the OpenGL standard (originally defined by Silicon Graphics) as the standard 3D API for Windows.

[Editor's note: It will also be interesting to see how Microsoft's purchase of SoftImage will affect things. SoftImage is responsible for much of the software behind Jurassic Park, The Mask, and many of the new glitzy advertisements.]

But, do people want 3D?

One obvious idea is to simply attack the existing 3D market by offering workstation functionality on PCs. However, this strategy may be flawed given the relatively small volume and the fact that customers are much more interested in maximum performance and full service. When a typical Hollywood epic costs tens of millions of dollars, is anyone really interested in going out on a limb with a no-name 3D clone just to save a few thousand bucks?

Instead, the driving force for PC 3D will be those truly "mission critical" applications-games! Already incorporating ad hoc 3D, the question is whether game designers will start to write to the OpenGL API, rather than rolling their own 3D routines. If so, the stage will definitely be set—assuming the price is right-for the emergence of 3D accelerators.

Beyond games, a very interesting question is whether 3D can migrate into the user interface itself. If a text directory listing is 1D, and a folder 2D, imagine a "beaker" 3D directory filled with liquid files. Copying would then be accomplished by "pouring" the contents of one beaker into another. "Disk Full" would be signaled by a spill, transfer errors by the appearance of bubbles, erasing files by flushing them down the toilet icon (with appropriate sound effects, of course).

It may sound dumb, but remember how most people thought the Mac was dumb too. Now, many of them are using a Mac—and the rest are waiting for a version of Windows that makes their PC look like one.

RISC IS DEAD—LONG LIVE RISC

Sitting around with other old hands, basking in the glow of the California sun (OK, and a little wine), the rhetorical question was posed, "Is RISC dead?"

Fulfilling my self-proclaimed role as Silicon Valley Guru, while at the same time obeying the pontificators' prime principle ("A meaningless prediction is never wrong"), I can clearly say the answer is "yes," "no," and "maybe." It depends on how the question is framed and who the questioner is.

Despite nearly a dozen RISC presentations, it is clear that as an architectural concept RISC is, if not dead, pretty senile. Looking beneath the surface, most of the new RISCs break no new ground. Instead, they focus on architecture-invariant implementation issues such as more

cache (112 KB on the DEC Alpha 21164), more pins (512 on the IBM Power2+), more MHz (500 for the NEC Gallop) and so on.

This is bad news for professors, researchers, Ph.D. students, and various other academic types who are faced with the choice of getting a real job or figuring out something new to investigate. The savior is the previously considered fringe VLIW (Very Long Instruction Word) concept which now, juxtaposed against unemployment, is starting to look pretty good.

The concept of VLIW may best be summarized by the title of a seminal paper, "Parallel Processing: A Smart Compiler and A Dumb Machine," by J. Fisher, et al in *Proceedings of the SIGPLAN '84 Symposium on Compiler Construction. Yes, you* could argue that this is a RISC concept too. So, substitute "smarter" and "dumber" for a more apt description of VLIW.

Superscalar RISC attempts to take a sequential program and, relying on a hardware dispatcher, tries to parallelize it at runtime. The VLIW, on the other hand, dispatches with the dispatcher in favor of parallelizing the code at compile time.

The superscalar RISC approach works OK for a few execution units (e.g., 3 in the case of Pentium), but tends to fall apart beyond that. First, the dispatch circuitry "explodes" in a nonlinear manner as the number of execution units and instructions examined increases. Perhaps more importantly, short of very messy "speculative execution" techniques, instruction reordering and scheduling is limited to basic blocks-instruction sequences with only one entry/exit (i.e., delimited by the dreaded conditional branch) which tend to be short (less than a dozen instructions].

It's been said that RISC stands for Relegate the Impossible Stuff to the Compiler, a concept which VLIW adopts with a passion. First, this shifts the cost from the silicon to compile time-generally a good thing since a program is usually compiled fewer times than it is run. Also, without cumbersome dispatch logic in the

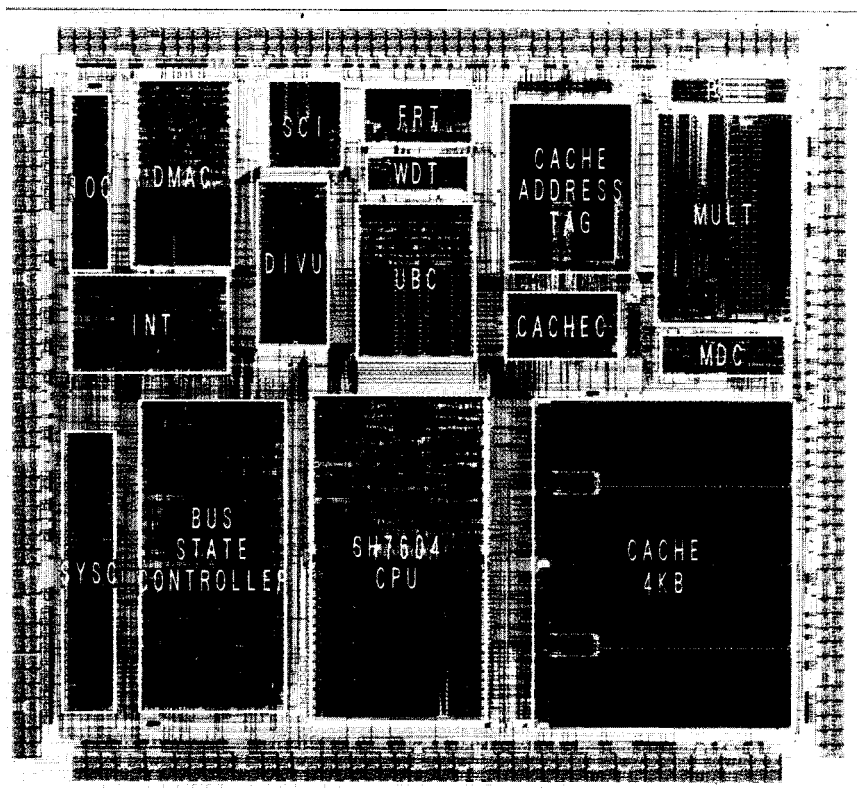
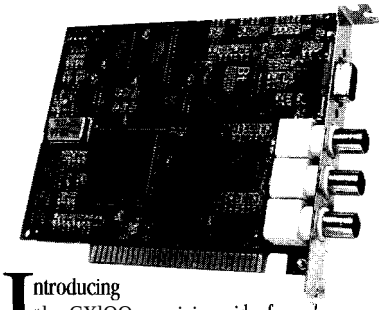


Photo 1--Besides the *usual* **peripherals** *such as DMAC, UART, and Timer, the SH2 contains a direct synchronous DRAM (SDRAM) interface, software-controlled PLL clock generator, and special-purpose multiplier and dividers.*

PRECISION FRAME GRABBER FOR ONLY $495*



Introducing the CX100 precision video frame grabber for OEM, industrial and scientific applications. With sampling jitter of only ±3 nS and video noise less than one LSB, ImageNation breaks new ground in imaging price/performance. The CX100 is a rugged, low power, ISA board featuring rock solid, crystal controlled timing and all digital video synchronization. A Software developers will appreciate the simple software interface, extensive C library and clear documentation. The CX100 is a software compatible, drop-in replacement for our very popular Cortex I frame grabber. A Call today for complete specifications and volume pricing.

— CX100 FEATURES —

- Crystal Controlled Image Accuracy
- Memory Mapped, Dual-Ported Video RAM
- Programmable Offset and Gain
- Input, Output and Overlay LUTs
- Resolution of 512x486 or Four Images of 256x243 (CCIR 512x512 & 256x256)
- Monochrome, 8 Bit, Real Time Frame Grabs
- Graphics Overlay on Live or Still Images**
- External Trigger Input
- RGB or B&W, 30 Hz Interlaced Display
- NTSC/PAL Auto Detect, Auto Switch
- VCR and Resettable Camera Compatible
- Power Down Capability
- BNC or RCA Connectors
- Built-In Software Protection**
- 63 Function C Library wlth Source Code
- Text & Graphic Library with Source Code
- Windows DLL, Examples and Utilities
- Software also available free on our BBS
- Image File Formats: GIF, TIFF, BMP, PIC, PCX, TGA and WPG

** THESE OPTIONS AVAILABLE AT EXTRA COST.

* $495 IS DOMESTIC, OEM SINGLE UNIT PRICE.

ImageNation Corporation
Vision Requires Imagination
800-366-9131

P.O. BOX 276 BEAVERTON, OR 97075 USA PHONE (503) 641-7408 FAX (503) 643.2458 BBS (503) 626-7763

#134

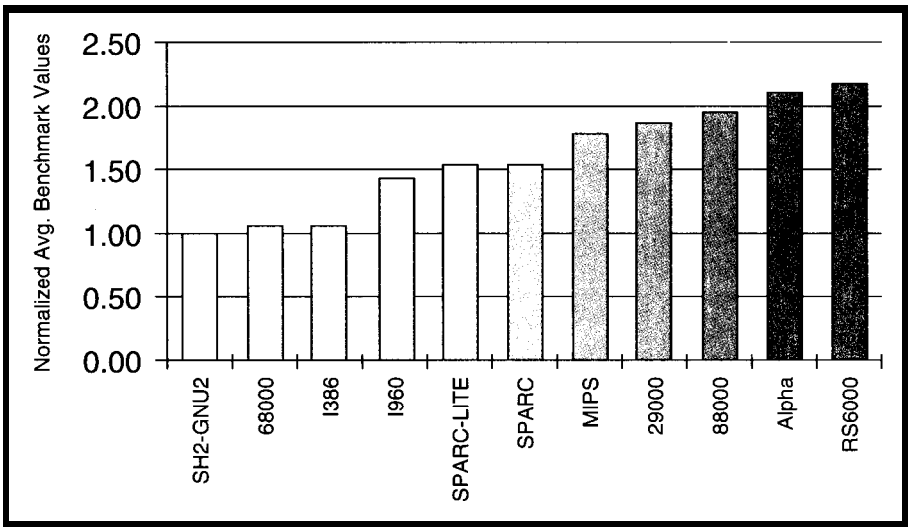


Figure 7—*The* Hitachi *SH* series R/SC processor uses a *K-bit, fixed-length* instruction to *dramatically* improve code *density.*

critical path, a VLIW should be able to run faster.

Finally, and probably most importantly, the compiler, using black magic techniques like trace scheduling, memory disambiguation, and directed acyclic graphs (DAGs), can search the entire program-not just a tiny superscalar dispatch window—and move code across basic blocks in the quest for ultimate parallelism.

So, watch for more VLIW activity, notably including an announced effort by Intel and HP to somehow combine VLIW techniques with 'x86 compatibility.

Speaking of Intel brings up the "maybe" answer to the "Is RISC Dead" question. At Intel, RISC really means "any non'x86 chip" and thus, in the PC context, refers to the Power Mac, PREP, MIPS, and Alpha versions of Windows NT, and so on.

No one knows the answer better than "The King," Bill Gates. I suggest an alternative question leading to the same answer is "if and when will native versions of Microsoft applications like Word and Excel be available."

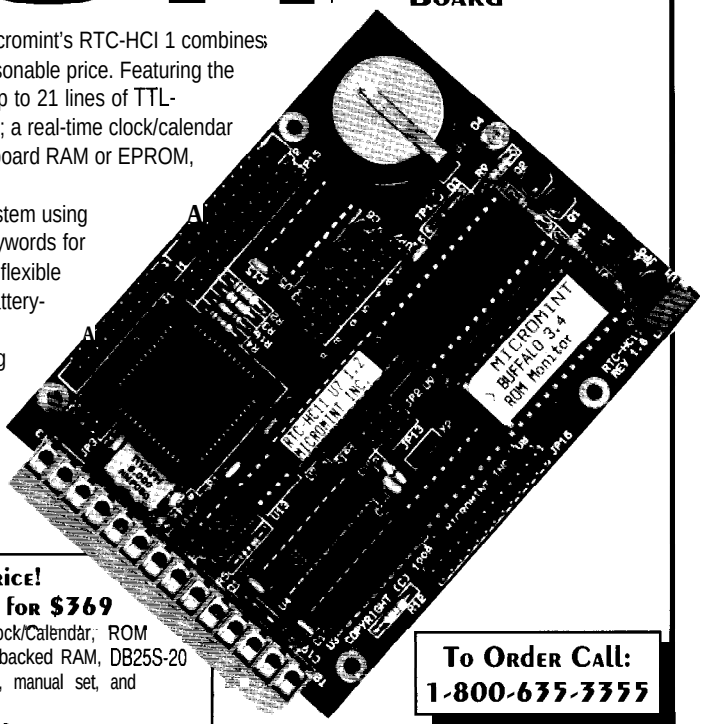
You may pose this question the next time you stop by the Microsoft booth at a trade show. If you happen to talk to a "strategic" type, you'll be reassured that the arrival of native mode applications for any or all non'x86 machines is imminent. On the other hand, "tactical" types are more likely to promulgate a philosophy that only machines with a giant installed base deserve support. Only "The King" knows for sure.

Silicon Valley is a very desktop-centric place. Most of these folks wouldn't recognize a nondesktop embedded micro if it bit them on the

RTC-HC11 Processor Board

Offering an exceptional value in a single-board embedded controller, Micromint's RTC-HCI 1 combines all of the most-asked-for features into a compact 3.5" x 4.5" package at a reasonable price. Featuring the popular Motorola MC68HC118-bit microcontroller, the RTC-HC11 gives you up to 21 lines of TTL-compatible I/O; an b-bit, d-channel analog-to-digital converter; two serial ports; a real-time clock/calendar with battery backup; 512 bytes of nonvolatile EEPROM; and up to 64K of on-board RAM or EPROM, 32K of which can be battery backed.

Software development can be done directly on the RTC-HCI 1 target system using BASIC-i 1, an extremely efficient integer BASIC interpreter with dedicated keywords for I/O port, A/D converter, timer, interrupts, and EEPROM support. In addition, a flexible configuration system allows a BASIC program to be saved in the on-board, battery-backed static RAM, and then automatically executed on power-up. Micromint also offers several hardware and software options for the RTC-HCI 1 including the full line of RTC-series expansion boards as well as an assembler, ROM monitor, and a C language cross-compiler.



Additional features include:
- Asynchronous serial port with full-duplex RS-232 and half-duplex RS-485 drivers
- 1 -MHz synchronous serial port
- CPU watchdog security
- Low-power "sleep" mode
- 5-volt-only operation
- RTC stacking expansion bus

Special Development System Price!
RTCHC11-DEV a $437 value for $369
Board w/8-bit ADC, EEPROM, 8K RAM, Clock/Calendar, ROM monitor, BASIC-11 in EPROM, 32K battery-backed RAM, DB25S-20 serial cable, utilities diskette (PC compatible), manual set, and HCTerm software.
Other configurations starting at $239

To Order Call:
1-800-635-3355

MICROMINT, INC. 4 Park Street • Vernon, CT 06066 • (203) 871-6170 • Fax (203) 872-2204
in *Europe: 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194 • Distributor Inquiries Invited!*

#119

nose. Thus, it's ironic that the embedded world is where RISC is poised for takeoff, not burial.

So far, high-end embedded RISCs like the Intel '960 and AMD 29k have been confined to pricey (e.g., >$1000) equipment such as laser printers, LAN hubs, avionics, and so on.

However, as chip prices inexorably fall, watch for yesterday's high-end chips to migrate into tomorrow's low-end applications. Notable examples include next year's wave of 64-bit video games and automotive engine control (Ford plans to challenge GM's CISCs with an IBM Power-derived RISC).

Further, watch for new "reality-based" RISCs to join the until now unchallenged ARM on the small-die-size, low-power front.

Consider the Hitachi SH series (the SH2 is shown in Photo 1) which achieves good (although not super-duper) performance without a lot of system design headaches or sticker shock. Running at less than 30 MHz, the SH won't win any drag races. But, on the other hand, you won't need any 5-ns SRAMs and there is actually a chance your gadget will pass FCC inspection.

Put away your fans and heat sinks, not to mention "active" (e.g., thermocouple, liquid] cooling techniques. The SH consumes only 0.5 W, an order of magnitude less than a truly *Hot* chip. Why, it even works in a low-cost plastic package.

The SH designers stuck with the RISC concept of fixed instruction length-they just made it 16 CISCs and as much as two times better than desktop RISCs. Remember, better code density not only stretches your memory dollars, but also multiplies the effectiveness of on-chip cache and memory.

All this dieting adds up to small die size, with the SH consuming only ¼ the silicon of the big-shot RISCs. This translates into low prices, prices approaching the magical $10 mark that separates technical curiosities from high-volume (>1M units/month) chips.

As the embedded market trend toward ever fancier C programs and bigger data sets bangs up against the 64 KB barrier, it's likely that a RISC is in your future.

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.

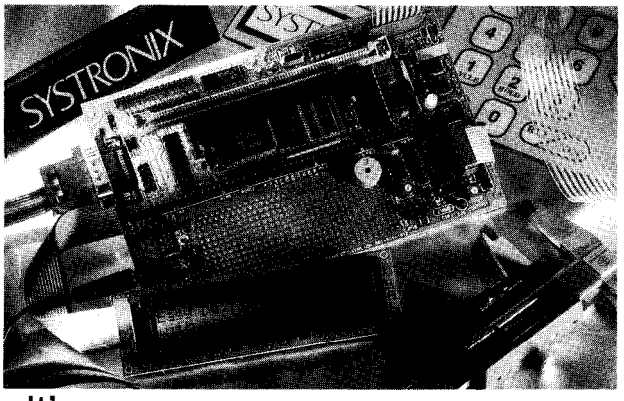
CONTACT

Hot Chips
c/o Dr. Robert G. Stewart
1658 Belvoir Drive
Los Altos, CA 94024
(415) 941-6699
Fax: (415) 941-5048

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

NEW! UNIVERSAL DALLAS DEVELOPMENT SYSTEM from $199!

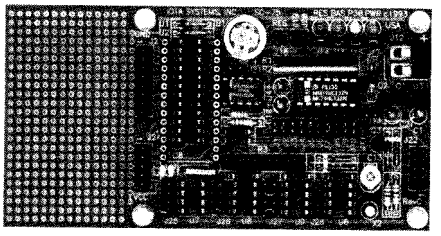


- It's a complete single board computer!
- One board accommodates any 40 DIP DS5000, 40 SIMM DS2250, 40 SIMM DS2252, or 72 SIMM DS2251, 8051 superset processor! Snap one out, snap another in.
- Programs via PC serial port. Program lock & encrypt.
- LCD interface, keypad decoder, RS232 serial port, 8-bit ADC, four 300 mA 12V relay driver outputs.
- Power with 5VDC regulated or 6-13 VDC unregulated
- Large prototyping area, processor pins routed to headers
- Optional enclosures, keypads, LCDs, everything you need
- BC151 Pro BASIC Compiler w/50+ Dallas keywords $399

SYSTRONIX® TEL: 801.534.1017 FAX: 801.534.1019
555 South 300 East, Salt Lake City, UT, USA84111

#135

Speed Your Development Process
By Using Our Con trollers!



We offer an array of controller boards and software tools for the 8051 and 87C751 families of microcontrollers. Complete packages are available to help you develop your projects. We also have a selection of add-on peripherals such as LCD and keypad interfaces.

Features:

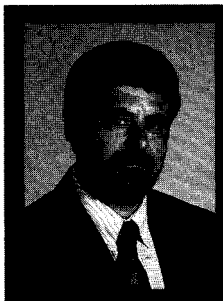
- Breadboard area
- Flexible I/O arrangement
- Powerful controller BASIC for the 87C752
- Simulators

Ph: (702) 83 I-6302 • Fax: (702) 83 I-4629
Iota Systems, Inc.
POB 8987 • Incline Village, NV 89452-8987

#136

Heavy Duty Hammers

Beef up the 8052 with the DS87C520



John's been spending time talking about the new DS80C320 processor, but what happens when you want a true high-speed microcontroller with onboard ROM? Bring on the DS87C520.

EMBEDDED TECHNIQUES

John Dybowski

Judging by the response I've received about my ec.32 project, there are a lot of people out there who think souped-up 8-bit processing makes sense for a lot of applications. I wouldn't be foolish enough to dispute that a significant number of applications demand a level of performance only attainable from advanced 16- and 32-bit processors.

But, it all boils down to using the right tool for the right job. I once heard if the only tool you have is a hammer, everything starts looking like a nail. Conversely, if all you've got to drive are nails, then the tool of choice should be obvious.

Looking back to the ec.32, it is evident that the system is broadly composed of two basic components: the ec.32 hardware and the Dunfield debug firmware or software. It is through the close coupling of these elements that the system can serve equally well as a general-purpose, single-board computer, a low-cost development system, and an evaluation vehicle suitable for realistically test driving the DS80C320.

Notably, this philosophy carries over to the new ec.52. Although this new system addresses an entirely different set of design goals, it maintains compatibility with the ec.32 and older 803 1 designs.

VERY HIGH-SPEED PROCESSING

Based on the Dallas Semiconductor's 8-bit DS87C520 controller, the ec.52 single-board computer ups the processing ante to unexpected levels. Running at 33 MHz, a minimum instruction cycle now checks in at 120 ns resulting in a truly impressive throughput of 8 MIPS.

As usual, the MIPS are made up of little instructions that excel at boolean operations and various bit-manipulation functions. But, this is the stuff many real-time systems are made of.

On the other hand, a different class of functions can be performed by simply combining a bunch of these small and seemingly inconsequential instructions. All it takes is enough time or bandwidth.

The 87C520, shown in Figure 1, builds on the basic features contained in the 80C320 and adds capabilities which include on-chip program and data memory. Special features have been added to keep power consumption in check during periods of reduced throughput-few applications need to run at full bandwidth continuously.

Since it contains on-chip memory, the 87C520 is capable of stand-alone, single-chip operation. This is made possible with the inclusion of 16 KB of EPROM and 1 KB of on-chip external (MOVX) RAM. This memory is in addition to the usual 256 bytes of directly addressable internal RAM.

To attain the required flexibility, the ec.52 does not use the 87C520 in single-chip mode, but instead uses external high-speed, nonvolatile RAM to provide a combined 32-KB program and data space. The internal EPROM is used to hold the resident debugging kernel and miscellaneous utilities and drivers that support onboard and offboard peripherals.

Regardless of the peculiarities of the specific peripheral, data is only a function call away. This (sort of) BIOS lets you access any of the system peripherals in a consistent and straightforward manner regardless of any device idiosyncrasies and interface complications. All these routines only consume only about 2 KB, which leaves the remaining 14 KB available for other purposes.

The peripheral set is rounded with the inclusion of 8 CMOS inputs, 8 CMOS outputs, a real-time clock-calendar with RAM, and an 8-channel, 12-bit A/D converter. A fully CMOS design combined with the 87C520's

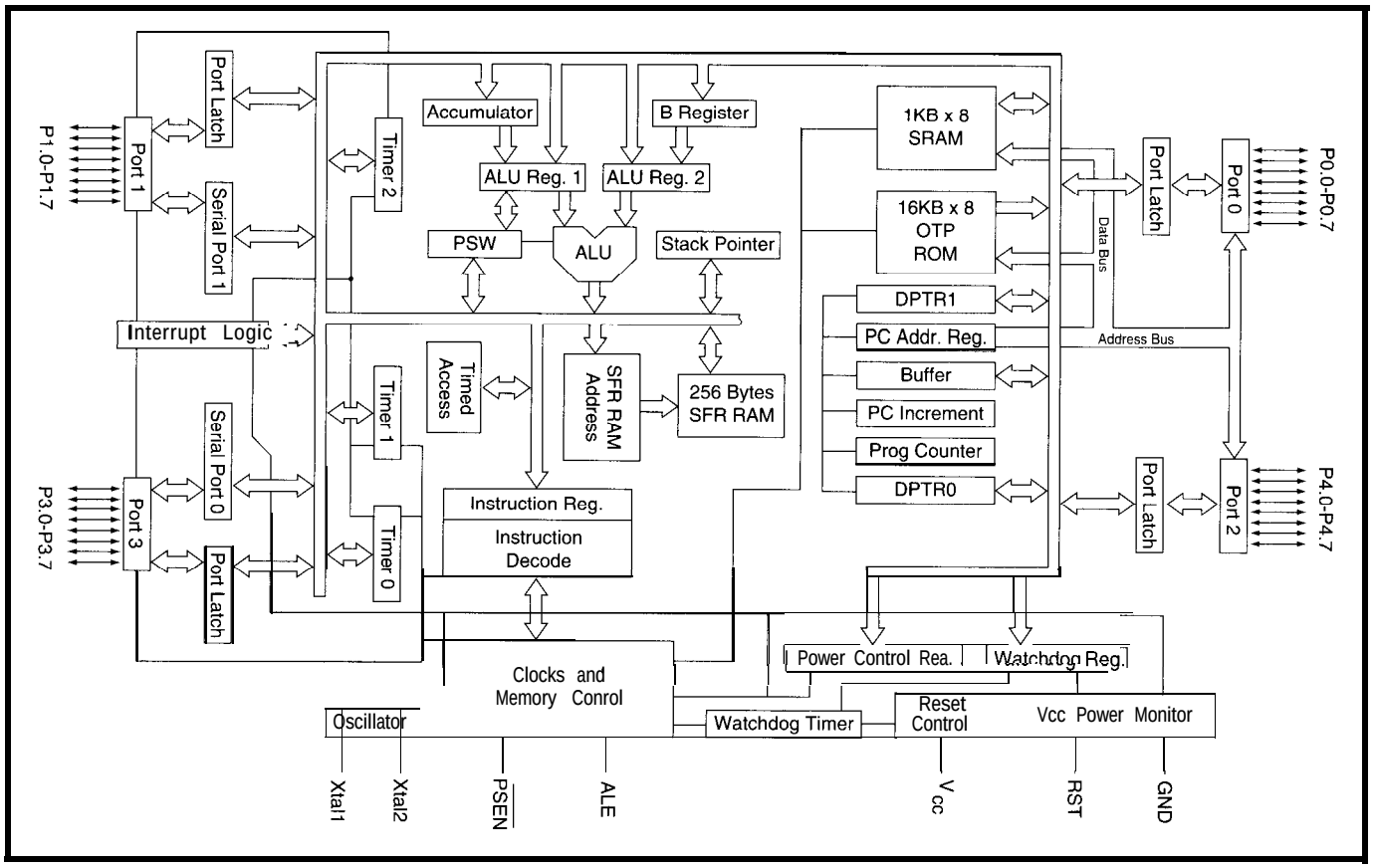


Figure I--The *DS87C520 microcontroller includes so many peripherals on the single chip that if starts looking like a complete system*

power management modes presents a system suitable for battery-powered applications and allows the use of a low-cost pass regulator. An I²C port is provided to support a variety of external peripherals. All this is packed on a board that measures only 4" x 4". The entire ec.52 schematic is depicted in Figure 2 and the circuit card is shown in Photo 1.

Although the ec.52 runs with virtually the same PC debugger and resident debugging kernel as the ec.32, the overall system organization is quite different. This isn't a result of any inherent dissimilarities between the 80C320 and 87C520 controllers. Instead, the different compositions simply exist because the two systems are intended to serve different applications. It's not even a performance issue since there's no reason the ec.32 can't be upgraded to 33 MHz.

The ec.52 processor's address and data bus is used to directly interface with the external 32-KB program memory, the data RAM, and the parallel I/O. While the 87C520 offers the option of inserting stretch cycles into MOVX instructions (external memory references), program references run at full speed.

The only way to gain headroom here is to use a lower-frequency crystal. For certain applications, doing this could be just the right thing. The system would certainly be less expensive, consume less power, and run much quieter. But, since I really want to see my old 8031 code run even faster than it does on the ec.32, this would defeat the whole purpose-at least for the moment.

FAST RAM/SLOW RAM

You may recall my tirade on system timing when I kicked off the ec.32 project (*CAJ* 49). The points I made are equally valid when applied to an 87C520-based system, except that timing margins grow ever smaller as the operating frequency inches upwards. Even with the use of fast AC logic, the access times mandate the use of relatively high-speed RAM.

The requirement for nonvolatile operation dictates the placement of a RAM controller into the chip-select timing path. The DS1210 significantly taxes the timing budget with its 20-ns (maximum) chip-enable (\CE) propagation delay. Of course, alternate methods of protecting RAM invoke less of a delay penalty, but due to a twist of fate, I ended up with RAM that allows more than enough slack to take up such a delay with no problem.

Taking a cursory look at the 87C520's instruction-fetch timing reveals that at 33 MHz, valid data must be available 70 ns after port 0 emits the low-order address (t_{AVDV1}). If we account for the 10-ns travel time through the 74ACQ373 address latch, this value shrinks to about 60 ns. The corresponding timing for valid data from high-order address at port 2 is indicated as 81 ns (t_{AVDV2}). Since A15 is inverted for use as the chip-select signal for the RAM, the 8-ns delay through the 74AC04 inverter and the 20 ns lost through the DS1210 must be accounted for. This leaves only about 53 ns to complete the data transfer.

It would be relatively easy to shave off a few nanoseconds which would allow the use of a 55-ns RAM.

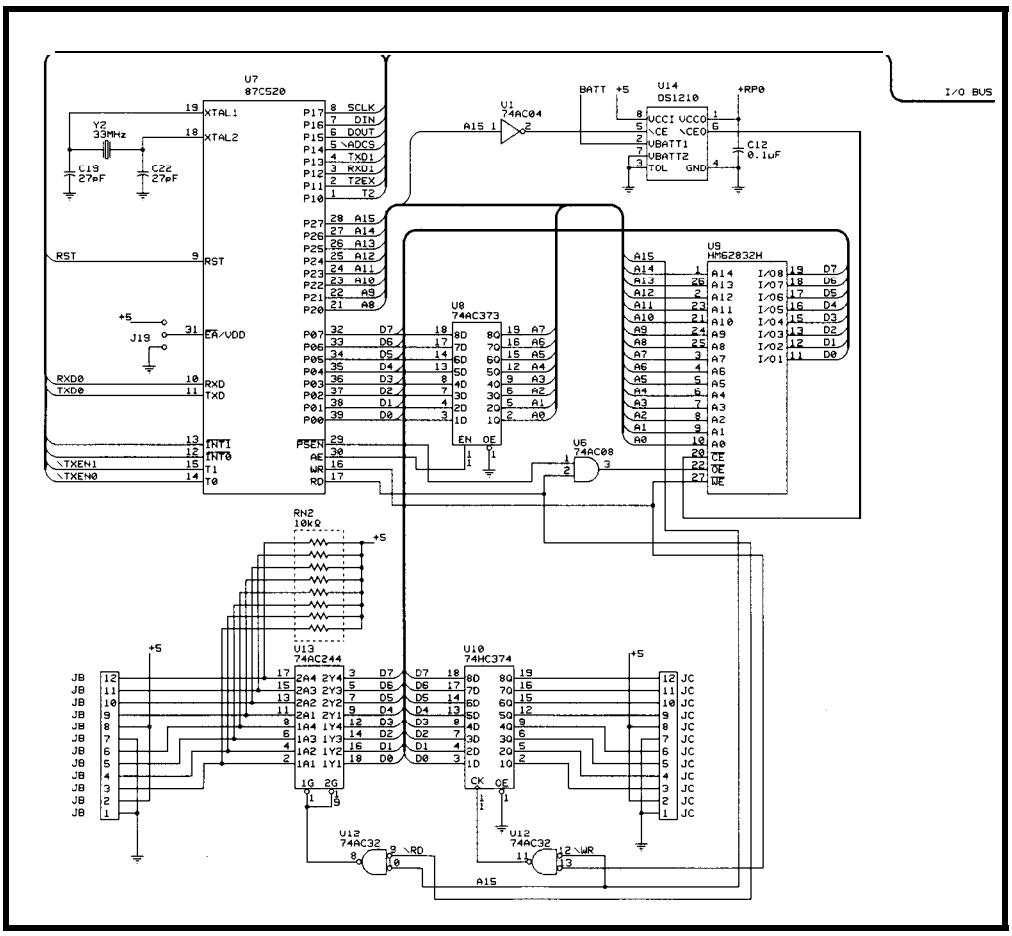


Figure **2a**—*The ec.52 includes battery backup for its RAM plus basic parallel I/O.*

However, this exercise turns out to be unnecessary since 55 ns puts us into the middle ground that falls between slow and fast RAM types.

Ironically, this distinction between fast and slow RAM has less to do with speed than with circuit optimization and parametric tradeoffs. As you would expect, the point at which a RAM is considered fast is continuously changing as technology develops. Not too long ago, RAMs with a 70-ns access time were considered fast. Now, 35 ns is fast. So, despite the fact that a 55-ns access time may seem mighty fast to some of us, by definition it is slow. This is good news since slow RAMs are significantly less expensive than fast RAMs. The bad news is that availability of 55-nsRAMs is spotty at best; multiple sourcing is somewhat of a problem.

To take advantage of the rapidly changing RAM scene, the ec.52 can accept either slow or fast RAM. Interestingly, though it's currently easier to get fast RAM that achieves the ec.52's access requirements, it won't be long before slow devices are widely available with under-55-ns access times. Nonetheless, the ability to accommodate slow and fast RAM enables the ec.52 to be detuned. The system can run at less than maximum clock frequency for applications that need its special features, but not full-bore 33-MHz operation or cost.

To accommodate the different RAM devices, the ec.52 circuit card accepts either a fast RAM that is usually housed in a 300-mil package or a typical 600-mil slow device.

Flexibility in the nonvolatile backup power source is also provided. The backup power can be derived from a 0.22-F supercap; a 2.4-V, 60-mAh NiCd battery; or even a BR1225 lithium coin cell. Each device has its advantages and disadvantages.

The supercap is at its best in very low-drain applications. It won't ever leak and never requires maintenance. The NiCd, although capable of much longer playing times and capable of multiple recharge cycles, is still a battery which will eventually need to be replaced. The lithium cell has the longest life under extremely light loads, but once its energy has been depleted, it must be discarded. I prefer to use a supercap to a battery whenever possible. The environmental ramification of dead batteries decaying in the landfills is, frankly, somewhat frightening.

To promote the viability of supercap backup scheme, the RAM I selected for the ec.52 is not only fast (by definition), but also possesses exceptionally good data retention characteristics. Hitachi's HM62832-HLP-35 delivers an access time of only 35 ns, but is capable of retaining data all the way down to 2 V with a maximum data retention current of 50 µA at 3 V. The typical value at room temperature is about 1 µA. This part's familiar nomenclature and benign DC specifications might make you feel like you're on familiar ground. But, this is no standard RAM. I guarantee its price will snap you back to reality faster than its access time.

The only other devices residing on the high-speed parallel bus are digital I/O ports. These ports are provided using an 74ACQ244 buffer and an 74HC374 latch. The inputs are pulled up to +5 V which enables them to be used with CMOS, TTL, or open-collector drivers. The outputs are raw HC outputs and, as such, can drive directly into CMOS, TTL, or other low-level loads.

ACCESSING I/O

At 33 MHz, data must be available 40 ns from the falling edge of \RD. This is the critical path for I/O reads. Since an 8-ns delay is incurred by the 74AC32 strobe gate, the time remaining for the transfer demands the use of a fast buffer such as the 74ACQ244.

You may recall that the 87C520 is capable of introducing stretch cycles that ease external memory access timing. Unfortunately, you can't designate stretch cycles to operate over only certain memory regions; they're either on or off.

Since I've already made a considerable investment in fast RAM capable of full-bandwidth operation, it makes sense to allow full-speed I/O as well. This way I don't have to monkey with stretch cycles on the fly when accessing I/O. Write-cycle timing is not particularly restrictive and allows the use of a standard 74HC374 latch.

The chip select for the digital I/O is derived directly from A15, which implies that these ports should be read and written at location 0 in the memory map-a reasonable assumption that's not entirely true.

Here's what happens. The 87C520 contains 1 KB of RAM located at location 0. When this RAM is enabled, it affects how the processor handles its I/O pins. On powerup, the 87C520 boots with its built-in external (MOVX) RAM disabled by default. This feature accommodates existing systems which have their lower data memory area already populated with RAM or peripheral devices.

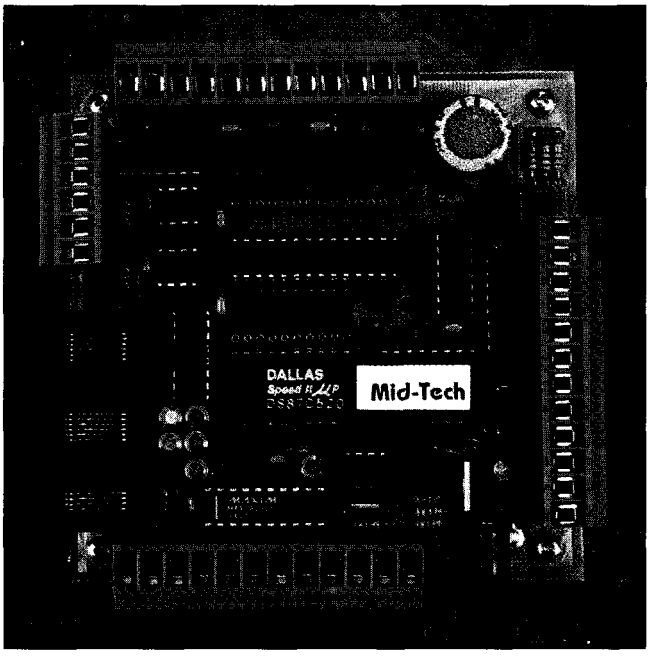


Photo I--The ec.52 very high-speed sing/e-board computer sports a truly impressive throughput of 8 MIPS.

The ec.52 enables this 1-KB RAM when the resident kernel takes control following reset. Accesses into the lowest 1024 bytes of data memory are directed to the on-chip RAM and nothing is emitted from the controller's I/O ports. This is as you would expect since the 87C520 is basically operating in single-chip mode and all ports are available as general I/O. This means the ec.52's externally mapped I/O ports must be accessed at some location above the 87C520's l-KB on-chip RAM. The ec.52 begins addressing the I/O ports at 400h, which is where the on-chip RAM leaves off.

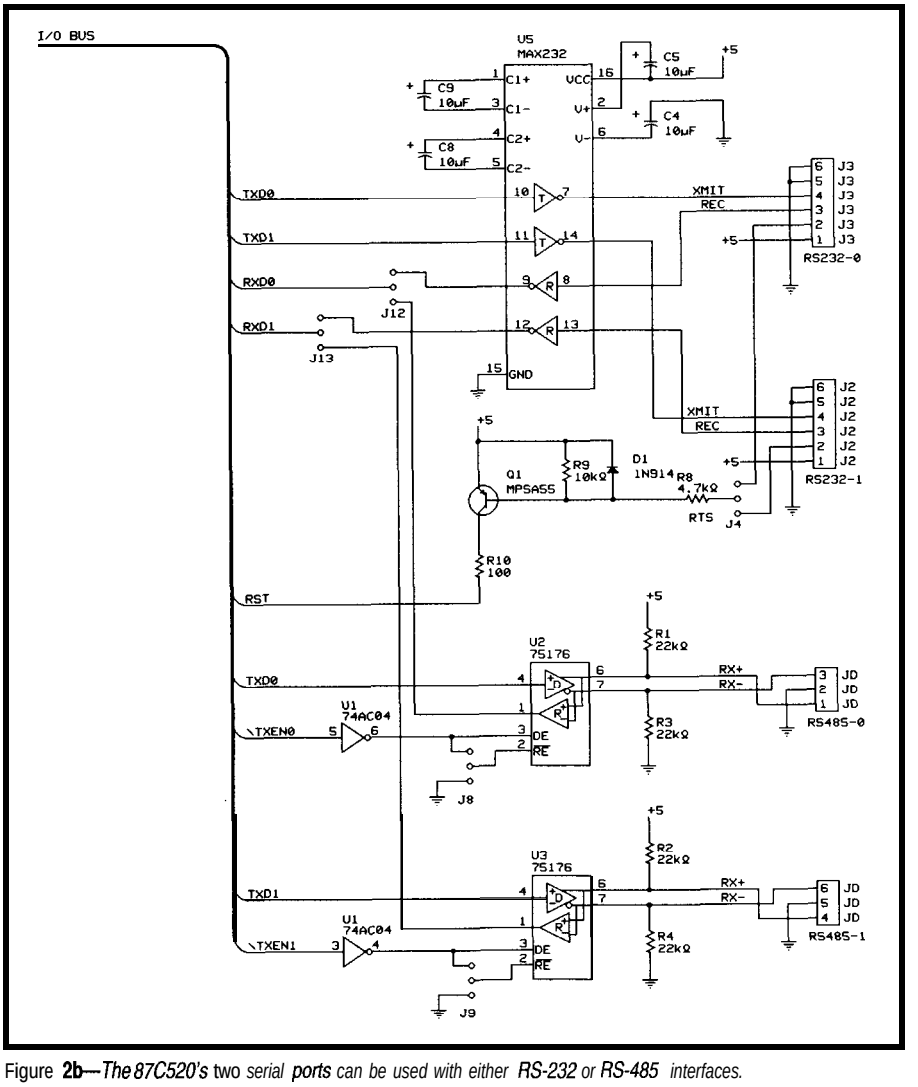


Figure 2b—*The 87C520's* two *serial ports can be used with either RS-232 or RS-485 interfaces.*

ANOTHER SERIAL SUBBUS

To save board space and interconnects, and to avoid unnecessary loading of the high-speed data bus, the remaining peripheral devices are interfaced to the processor serially. Although it's no secret I like using the I²C for my serial peripherals, I opted in this case to go with a more conventional Microwire interface.

The Microwire standard is based on a three-wire scheme consisting of DIN [data in), DOUT (data out), and SCLK (serial clock). Microwire devices that don't transmit and receive simultaneously connect the two data pins together, thereby allowing a two-wire interface. Unfortunately, even

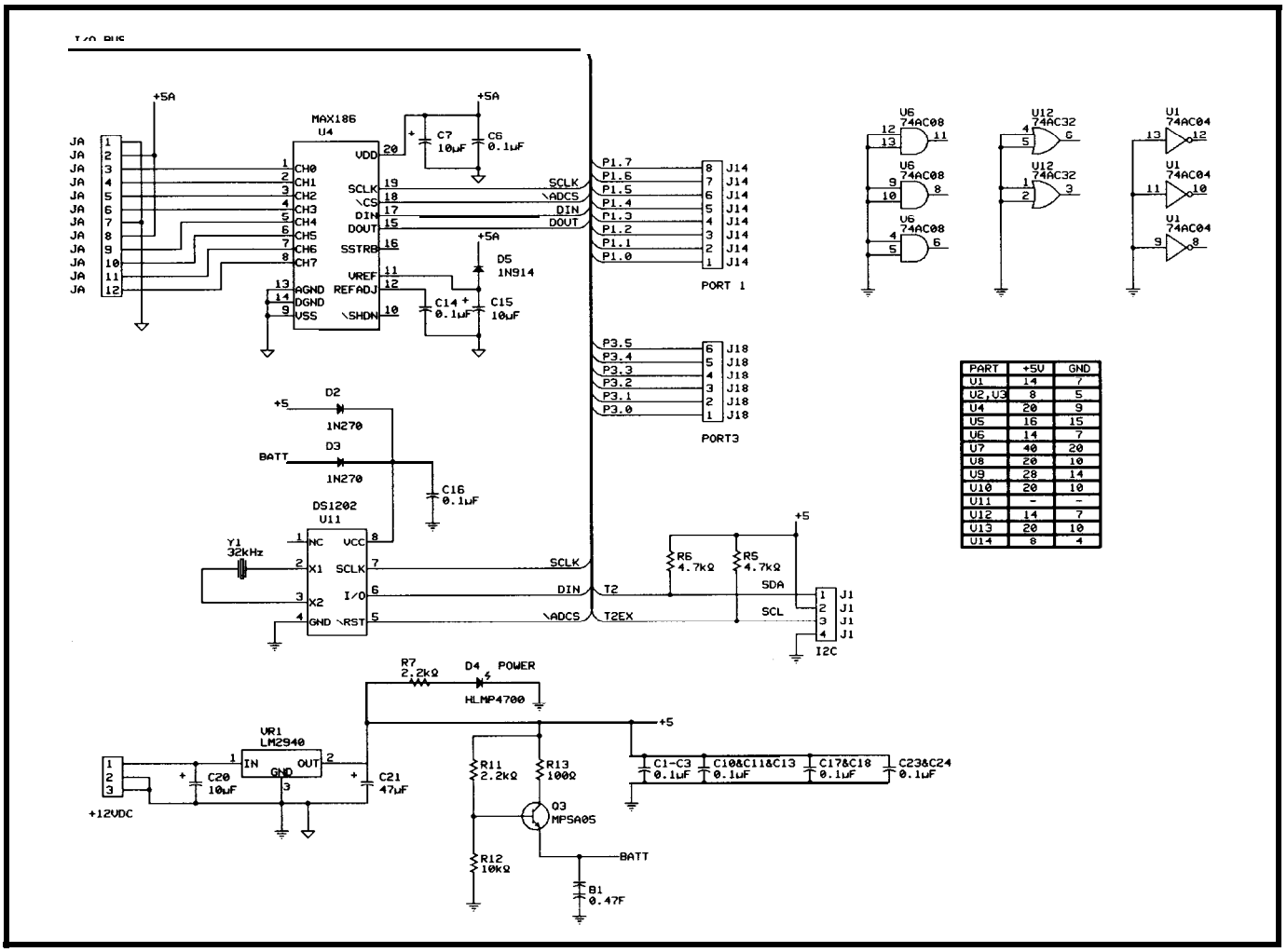


Figure **2c—***The ec.52* includes an *8-channel,12-bit* serial A/D converter and a serial *clock* calendar. The *power supply section is very* ***simple*** *due to the board's minimal power requirements.*

though the basic interface is carried over two or three wires, they generally don't tell you that each individual I/O device needs an independent chip-select line.

So much for serial...

Luckily, with just two peripherals to support, Microwire serves reasonably well. It's a fast interface; you can clock data around just about as fast as you want even with a 33-MHz processor. And ironically, its somewhat loose protocol is what gives it the flexibility needed to handle different types of peripherals, word lengths, and formats. But, don't think I'm about to abandon the I²C. As with the ec.32, an RJ1 **1** I²C tap is available for outboard devices.

Although I've been talking generically about National Semiconductor's Microwire, the parts I'm using are not National's. The system A/D converter, a Maxim MAX1 86, actually adheres to the Maxim serial interface standard. In its simplest form, the Maxim serial standard is very similar to Microwire and Motorola's SPI. The other serial peripheral is Dallas's DS1202 RTC which, other than the fact that it has data and clock lines, has less in common with these standards.

See what I mean about loose protocols?

DATA ACQUISITION ON A CHIP

When pressed for space, it pays to look to semiconductor manufacturers for highly integrated answers to your real estate problems. This makes sense not only from a packaging standpoint, but also for protection. It is wise to encapsulate as many sensitive analog functions as possible.

In this respect, the level of integration attained in the Maxim MAX186 is truly impressive. Thanks in part to a reduced pin count made possible by using a serial interface, the MAX1 86 provides a complete data-acquisition system on a 20-pin IC. The combined functionality includes a 12-bit data converter, 8-channel multiplexer, high-bandwidth track and hold, and a built-in 4.096-V reference. The converter can be set up to operate with eight single-ended or four differential channels. A block form of the MAX186 is shown in Figure 3.

The MAX186, a successive-approximation converter, requires a conversion clock to drive the analog-to-digital conversion steps. This clock can either be derived from the SCLK or can be internally generated by the MAX186. The ec.52 uses the external conversion clock in which SCLK not only shifts data in and out, but also drives the A/D conversion sequence. Following the receipt of the control byte, successive-approximation bit decisions are made and appear at

DOUT on each of the next 12 SCLK falling edges.

Using external clock mode eliminates the need to sample the SSTRB pin to synchronize the processor to the internal free-running conversion, but some restrictions do apply. The conversion must be allowed to complete in a certain minimum time. Otherwise, droop in the sample-and-hold capacitors may lead to degraded conversion results. The clock period must not exceed 10 µs and overall conversion must be complete within 120 µs. Also, the duty cycle must be held to 45-55 %. Using the ec.52's built-in function calls guarantees these conditions are met.

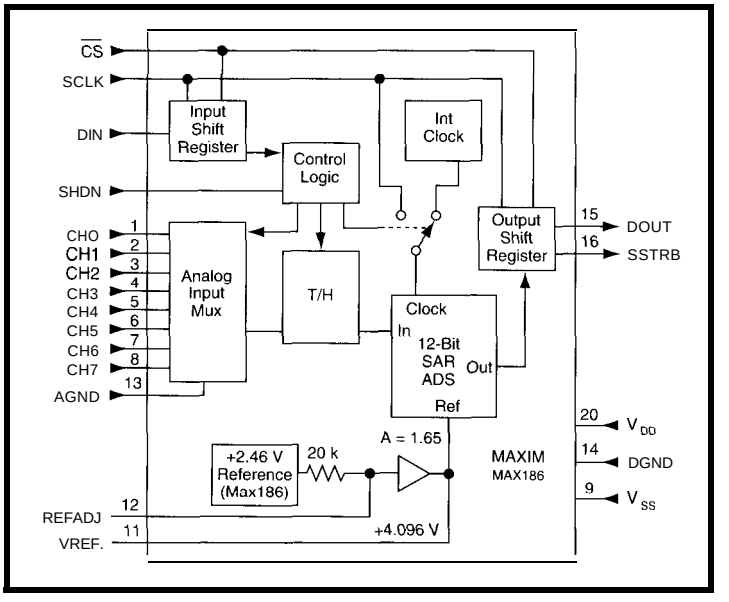


Figure 3—*The MAX786 is an B-channel, 12-bit successive-approximation A/D converter The processor interface is serial to reduce the chip's pin count.*

SYSTEM TIMEKEEPING

The only other peripheral to share this Microwire interface is a serial timekeeping chip. The DS1202, shown in Figure 4, contains a real-time clock-calendar and 24 bytes of static RAM. It counts the usual intervals from seconds to years and automatically adjusts for months with fewer than 3 1 days and for leap years. In other words, it does the things you expect an RTC chip to do. The DS1202's claim to fame is that it does all of this while consuming less than 300 nA. This, in fact, is the maximum current the clock will draw at 2 V with its oscillator running and its counters counting.

The DS1202's interface, although similar in principle to the MAX1 86, differs in several details. Instead of having two data pins, the DS1202 has a single bidirectional pin. Instead of an active-low chip enable, the DS1202 has an active-low reset. This effectively amounts to an active-high chip enable that facilitates tying the same signal to both chips. One will always be enabled, but this has no effect since a specific sequence of data and clock bits is required to cause a reaction.

Like the MAX186, you can essentially move data about as fast as you can clock it. [The maximum clock rate is 2 MHz.) Individual clock and RAM locations can be read and written. A burst mode also exists where the entire contents of the clock or RAM can be transferred in a single operation. A write-protection capability is provided as well for added security.

You may be interested to know that Dallas has a new and improved version of this RTC called the DS1302. It has separate pins for t5 V and a battery, optional trickle charge capability to the battery supply pin, and seven extra bytes of RAM. Although in some applications these could be valuable features, they are unnecessary for the ec.52

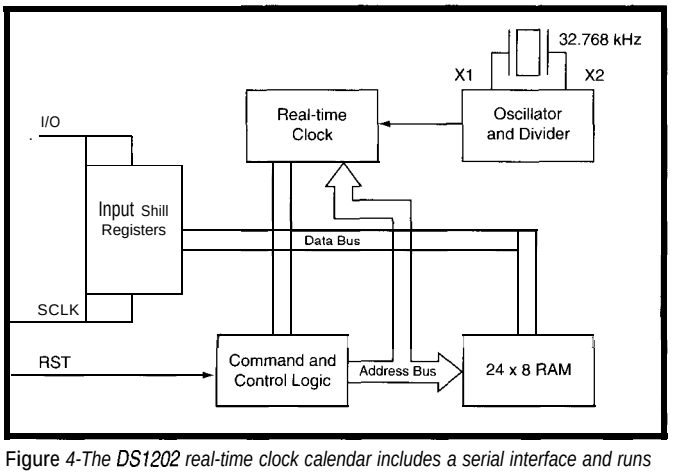


Figure *4-The DS1202 real-time clock calendar includes a serial interface and runs on almost zero power (<300 nA* max.).

JUST LIKE REFORM SCHOOL

Some of the ec.52's other features include two hardware UARTs that can be set up for RS-232 or multidropped RS-485, three 16-bit timers including a timer-capture system, and some general-purpose parallel I/O and interrupt lines.

With that, I think we've got enough hardware to last a while now. Next month, I'm going to put a controller behind bars.

John Dybowski is an engineer involved in the design and manufacture of embedded controllers and communications equipment with a special focus on portable and battery-operated instruments. He is also owner of Mid-Tech Computing Devices. John may be reached at (203) 684-2442 or at john.dybowski@circellar.com.

SOURCES

For elements of this project, contact:

Mid-Tech Computing Devices
P.O. Box 218
Stafford Springs, CT 06075-0218
(203) 684-2442

Individual chips are available from:

Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 27613
Phone/fax: (9 19) 676-4525

I R S

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) **871-1988—Four** incoming lines
Internet E-mail: **sysop@circellar.com**

This *month's messages include something that those who frequent BBSs are likely familiar with, but* **others may never have seen: message quotes.** *There are many situations* **when** *someone reading a group of messages may not have the entire thread to refer back to. It can be very confusing to read a rep/y without the benefit of being able to read the original question.*

In such situations, the person writing the rep/y will often "quote" portions of the original message so that the reply will make sense even if the original is unavailable. I usually edit the messages used in ConnecTime to eliminate quoting, but this month I came across a **Msg#:23706** *left in. The quoted p usually have a ">" character at the start of each line to make them easy to pick out.*

You'll notice another use for quoting is to make answering a multipart question much easier. Rather than try to phrase the answer to indicate which part of the original it's answering, simply repeating the question before the answer makes things **obvious.**

The first thread this month covers everything you ever wanted to know about tantalum capacitors and their failure conditions. They aren't necessarily the panacea some designers make them out to be.

Finally, in the second and last discussion, we fake a quick look at some alternatives to varying the speed of an AC motor.

Tantalum capacitor mystery

Msg#:23654
From: BARRY KLEIN To: ALL USERS

I was wondering if any of you have any insight into the scenarios that might cause tantalum capacitors to catch fire. Typically, this occurs on computer peripherals, such as tape and disk drives. It occurs very, very infrequently, but when it does the end user or OEM wants an explanation—immediately! Typically, these peripherals are run with the common +5/+12-V computer switching supplies.

I have access to several manufacturer's disk drives and the large majority seem to design in these caps without any form of transient or reverse-voltage protection. Ratings on the caps are typically 16 V for the 5-V and 25 V for the 12-V caps. As these devices typically have 4-pin Molex power connectors, there is the possibility of applying power in reverse (the pins touch) or with a floating ground. Sometimes after this is done, the peripheral will still function after power is applied normally. A few questions:

1. Will raising the voltage rating on the caps help anything in this regard?

2. Is there a good way to test to see if a cap has been damaged by power reversal or whatever?

3. One manufacturer I contacted could design a transorb-type device into the capacitor that he thinks would cost less than a higher-voltage-rating cap. Would this help or would the transorb _cause_ the cap to catch fire if reverse voltage was applied long enough?

Msg#:23706
From: JAMES MEYER To: BARRY KLEIN

> 1. Will raising the voltage rating on the caps help
> anything!

Probably not, since most of the problems come from reverse voltage, and even if the normal voltage rating goes up, the reverse voltage rating never goes over one volt.

> 2. Is there a good way to test to see if a cap has been
> damaged?

The leakage current of the cap (when it's biased normally) should go up by a good deal if it has been damaged.

> 3. One manufacturer I contacted could design a
> transorb-type device into the capacitor that he
> thinks would cost less than a higher-voltage-
> rating cap. Would this help?

I don't think so. It would be better to prevent the reverse voltages that damage the caps in the first place. Adding a fuse (or even just a narrow place in the PC board trace) in series with the incoming power and I-amp or better diode (reverse connected) across the power input to the circuit would protect *all* the capacitors on the board at the same time. Except for the caps that got installed backwards when the board was put together at the factory. You *have* checked for that, haven't you?

Msg#:24199
From: BARRY KLEIN To: JAMES MEYER

Yes, they are installed correctly-although I suppose they could have been labeled backwards. One additional

question: What if the power supply was oscillating at a high frequency? Could this cause damage to the cap! It is suspected the problem occurs when power is applied by the Molex connector (hot). Can the typical PC power supply oscillate with no load?

Msg#:24813
From: JAMES MEYER To: BARRY KLEIN
> What if the power supply was oscillating at a
> high frequency? Could this cause damage to the cap?

Possibly. Tantalum caps have a large capacitance value for their size. They also have a somewhat higher ESR (Effective Series Resistance) than some other types of caps. The leakage factor and ESR for tantalums increase as the caps get hot. The high ESR would mean that tantalums would begin to heat if large amounts of AC current were forced through them. Since they're small and can't get rid of heat very well, the heat would make them leak more and get hotter in a vicious circle that could end in lots of smoke and maybe some flames.

Although I *do* use them for DC power supply bypass filters, tantalum caps should never be used in a critical application when there is a possibility that large amounts of AC current will be passed through them.

Take a look inside a *real* IBM PC power supply sometime. There are filter caps everywhere, but I can't spot even *one* tantalum, they're all aluminum.

> Can the typical PC power supply oscillate
> with no load?

There is no such thing as a "typical" PC power supply. Some early switching supplies would shut down if there wasn't at least a minimum load and I guess they could try to start again only to shut down in a cycle, but I wouldn't call this a real oscillation.

Msg#:25589
From: BARRY KLEIN To: JAMES MEYER

Thanks for your input. We kinda got on this subject a while back when I had a personal interest in the failure modes of PC supplies. Now I am asked to take a look at this at work. I have applied +12/+5 in reverse to see effects, etc. The only thing I see so far is that if you apply the voltage either correct or reverse polarity, but float the supply ground from the peripheral, a negative voltage appears on the caps of about 0.5 V. Probably restricted to that by internal diodes in the ICs on the board. Most specs will allow "temporary" negative voltages of this level though. So I suspect something is funny with the supply and that's the avenue I'm taking next.

Msg#:26459
From: JAMES MEYER To: BARRY KLEIN

I would rate the supply as pretty low on the list of suspects.

I have seen some of those epoxy-dipped tantalum caps that were marked backwards for polarity. Those type caps are constructed from tantalum-based powder compressed into a cylinder. There is a wire lead running the length of the center axis of the cylinder and another lead soldered onto a layer of silver that's plated on the outside of the cylinder. The center lead is the positive connection and the outer lead is the negative one. Once the whole thing is dipped in epoxy, it's often hard to tell by just looking at the cap which lead is which.

If a cap burns up, though, the wire leads are usually left attached to the PC board. If you get the burnt remains before somebody disturbs them, you can usually determine which lead was which even though the tantalum part of the cap might be ashes.

IMHO, the most likely culprit will be defective, mislabled, or misinstalled caps. Any over or reverse voltage applied to a whole board should result in more than the caps going "fritz."

Msg#:27184
From: BARRY KLEIN To: JAMES MEYER

Well, even if mislabeling was a problem it wouldn't be the cause. The caps are installed by surface-mount machines. They don't care about labeling.

I think what the real problem is is that some people are hot plugging the drives. The dv/dt is too great and can inflict damage. The specs for tantalum caps specifically discourage you from using them in any applications where extremely low source impedances exist-like nickel hydride or cadmium powered applications or switching circuits. I took some current probe measurements and I think this is the culprit. Surface-mount electrolytics are just coming on the market and should be better for these locations if they fit on the board. Anyway, thanks again.

Msg#:27491
From: JAMES MEYER To: BARRY KLEIN
> Well, even if mislabeling was a problem it wouldn't
> be the cause. The caps are installed by surface-mount
> machines. They don't care about labeling.

No they don't care about labeling. They simply rely on the manufacturer to put the little suckers into the carrier tubes or onto the tape reels so that they're all pointing in the same direction. If one got turned around, would your pick-n-place machine know the difference?

I still think that they're getting installed backwards.

CONNECTIME

Msg#:30003
From: PELLERVO **KASKINEN** To: BARRY KLEIN

While I cannot give an absolute solution to your question, I have some relevant experience that I want to share here. For the first, there are at least two quite different technologies used for making the tantalum capacitors. One is a wet slug type, which you typically find packaged in tubular metal cases. The other one is the dry type, which typically appears in the epoxy drop-shape packages. They have slightly different characteristics, but share one common feature: very high volumetric efficiency (small cases for a given CV product [capacitance and operating voltage]).

The high volumetric efficiency can and does have a drawback: the small volume or rather a small surface area results in a minimal power dissipation capability. In other words, if the ESR-generated power becomes large, then the small capacitors will become _very_ hot. Becoming hot is the primary cause for starting a fire..

When will the ESR cause this problem in a system component? Simple, it happens whenever there is too much ripple current through any particular capacitor. That again is likely to happen when some _other_ capacitor is out of the normal duty of contributing to the smoothing action. So, in your case, probably the aluminum electrolytic capacitor in the power supply has failed open due to a soldering defect or something similar. Then the poor tantalum capacitor in some plug-m component may try to carry the entire ripple current and fail miserably.

There _can_ be another problem. The one capacitor may happen to resonate at a ripple frequency, which may not be fixed. Actually, a capacitor needs something inductive in the wiring to get into the dangerous resonance, but there could be certain amount of inductance in the wiring or there can be a choke for intentional inductance. I don't have any estimate about the likelihoods of these kinds of resonances in a PC, but I have seen all kinds of unpleasant resonances on the switching motor drives that I have built.

One last possibility. The chopping of the load current that a disk or tape drive may be imposing on the supply rail would normally not do too much harm, but if we assume that the supply is in current limit or there is a bad contact somewhere along the line, then all the ripple current goes predominantly through the local capacitor. Again, I do not know how bad a contact would need to be in order to cause this overheating and not cause an immediate failure to operate or an overheating at the bad contact site. But a current limit in the power supply could easily become serious enough. Too many peripherals pulling current at the same time could even lead to energy swings between the different peripherals (their local capacitors).

Enough of speculations for this time. Just a side note or two. There was and still is a small company in Nashville, TN, making very specialized tantalum capacitors. They are designed, dimensioned, and tested for extreme volumetric efficiency and simultaneous reliability in power supply use. The volumetric efficiency and long life expectancy at high temperatures are the key parameters for their main use: in NASA spacecraft.

Actually, NASA has experienced enough slowdown and changes in personnel to cause the inevitable: New engineers specified something else into their systems and had failure after failure. Finally they started asking questions about how the same things were done before and found that only this obscure company in Nashville had ever successfully made those critical capacitors. So, they dug out the old files and ordered some more of these tantalum capacitors. One problem solved-the old workhorses still worked fine.

For about half a year before I moved to NC, I had the opportunity of seeing certain phases of this manufacturing process while I helped the owner in some research about the dry-type tantalum capacitors. I also have seen the nice colors that the anodized slugs exhibit. In fact, you can tell the capacitance variation in the batch from the variation of the rainbow colors! But the important detail here is that for highest volumetric efficiencies, the surface area of the tantalum powder has to be maximized. That means using finer and finer powder. That again, after anodizing, eats its share of the particle-to-particle conductive path, which tends to increase the ESR and cause some catastrophic failures at higher currents due to thermal expansion and hot spots. But we handled that part already, didn't we? I try to keep myself from getting too far in the esoteric details. After all, there are plenty of trade secrets in there.

Msg#:30042
From: BARRY KLEIN To: JAMES MEYER

Well thanks to both of you for your input. We did look into fused tantalums but they are way too expensive for such a high volume application. You know, the failure rates are so low that they approach the specified failure rate of the part. It's just that when they go it's very obvious! So using a higher-voltage-rated part may have the same failure rate and result.

Msg#:30231
From: JAMES MEYER To: BARRY KLEIN

I concur. I haven't ever seen an aluminum electrolytic catch fire. If you've got the real estate, you might want to think about switching.

Msg#:36567
From: **C.D.** PRITCHARD To: JAMES MEYER

We had an electrolytic capacitor ignite in an smallish UPS in a computer room. Aluminum cap was on AC side

CONNECTIME

and about 6" high x 3" dia. Ruptured cap with charred and burnt plastic jacketing on the cap and heavily blackened area above the cap on the underside of the UPS's metal top. The smoke tripped the halon system which discharged and disconnected power to the UPS. Had the fire protection not been there, the fire might have spread to other materials. UPS manufacturing was tight-lipped on what caused the cap to fail.

AC motor speed control

Msg#:36330
From: DAVID WHITE To: ALL USERS

I need to slow down a swimming pool pump that I use on a KOI fish pond. It runs at full speed during the summer, however in the winter months when fish activity slows down I need to slow it down. It runs on 120 volts at 8 amps.

Can I just put a diode in the hot lead, say a 600-volt,10-amp diode, without any problems and will this slow the pump down about half? Any help any of you might have would be appreciated. Thanks in advance.

Msg#:36997
From: ED **NISLEY** To: DAVID WHITE

Nope, an AC motor requires an AC input. Converting it to pulsed DC will fry the poor thing.

How about adding a little plumbing around the motor so it happily pumps water in a closed loop with a little flow through the pond? That might be easier on the motor than restricting the flow through the pump.

Msg#:37191
From: JAMES MEYER To: DAVID WHITE

I know of no AC-driven motor of the size that you obviously have that would 'not' be damaged by placing a diode in series with it.

Motors like you probably have, were designed to run at one speed only.

If I had your requirements, I'd add a second, smaller pump and motor combination to the system. A switch could select which motor would get power, and a valve could isolate the working pump from the idle one.

One other idea: if the motor is connected to the pump with pulleys and a belt, add another set of pulleys to reduce the pump speed while letting the motor run at its normal speed. Look at a drill press to get an idea of what I'm talking about.

Msg#:38841
From: GEORGE NOVACEK To: DAVID WHITE

No way. You will fry it. Most of the pool pump motors are asynchronous motors which will allow some degree of speed control. I'm no motor expert, but by decreasing the voltage, the slip will increase and the speed will drop somewhat. I've seen motors controlled with transformers and rheostats as well as triacs. The best controller I saw used zero crossing and modified the number of 60-Hz cycles to control the speed. I don't think the controller will be cheap no matter what you do.

Personally, I wanted to save some energy when my pool is not being used, so I control the duty cycle of the pump using an X-10 interface. During weekdays (when nobody's home), 10 minutes on, 10 off. At night, it's 10 minutes on, 20 minutes off. On weekends or on command, continuous.

Msg#:38556
From: DAVID WHITE To: GEORGE NOVACEK

Thanks all for the responses. After I left the message I went back and scanned the messagebase for AC motors and found the same answers. It looks like the best way to handle this problem is with a separate smaller pump for winter use. Thanks again for the response. This is the best BBS there is.

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 360 KB IBM PC-format disk for only $12.

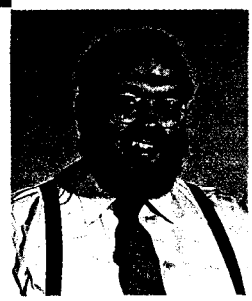
To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 8752199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

I R S

425 Very Useful 426 Moderately Useful 427 Not Useful

STEVE'S OWN INK

A Majority Gains Control



A couple months ago Ken and I commented in our editorials about the *Computer Applications Journal*'s future commitment to home automation and building control. Until we can underwrite an independent dedicated magazine on the subject, *CAJ* plans extensive coverage through quarterly supplements.

To further establish, in our own minds as well as those of our advertisers, that our readers are both receptive and ready, I offered *FREE* the printed-circuit board and the software for the Circuit Cellar HCS II-DX to *CAJ* subscribers. (You can still take advantage of this free offer by getting a copy of *CAJ* 50 or faxing us for a copy of the qualification card. The offer is only good until 12/31/94, so don't delay.) My invitation generated an overwhelming response.

There is nothing quite as exhilarating as coming in after a quick business trip and finding a pile of a hundred DX-offer cards on your desk. In fact, by the time the first *CAJ* Home Control supplement hits the stands in January, there will be more than 2,000 HCS owners feverishly waiting for substantive technical presentations! I view this as an astounding affirmation of your interest in home control.

However, it does bring up the question about whether HCS users are content to follow an industry or whether they want to lead. Even with the prodigious advertising of alternative automation control systems, I suspect that their total sales are mediocre by commercial standards. I further estimate that HCS II owners will eventually be a majority. Such a user base can't be ignored either editorially or by the advertisers. When I see this much interest, I visualize a plethora of application articles and a bonanza of sensor and support merchandise offerings.

Ok, ok, I know my pet interests are getting me ahead of myself. Of course, the more of you who get off the fence and join me in home control, the less it will seem like "Steve's pet interest" and more like catering to the majority.

Finally, I want to thank all of you who participated in emptying the Circuit Cellar of all my old manuscripts and prototypes. Your enthusiasm was such that everything is gone, and I now have a few spare shelves. Surprisingly, the response I've gotten back from those who've receive project boxes is amazement. They're astonished that I actually did what I said I would. Has the world really gotten to that?

Well, people, if there's one thing I hope you'll remember from our association, it's if I say I'm going to do something, I do it! There are at least 75 people, including one guy in Louisiana with a $12.000 Mandelbrot Generator and another in South Africa with a pile of Trump cards, who can attest to that.

