

**SPECIAL BONUS SECTION: HOME AUTOMATION & BUILDING CONTROL**

# CIRCUIT CELLAR

**INK<sup>®</sup>**

THE COMPUTER APPLICATIONS JOURNAL

#60 JULY 1995

## GRAPHICS & VIDEO

Using Color in  
Scientific  
Visualization

Digital Video Resizing

MIPS for the Masses

Transient  
Circuit Protection  
Explored

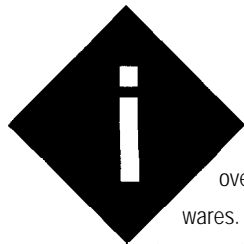


\$3.95 U.S.  
\$4.95 Canada



# EDITOR'S INK

Habitech95



Imagine walking onto a show floor and finding over 80 exhibitors showing their home automation wares. Turn around, walk down the hall, and you find dozens of home automation classes and seminars dealing with topics such as equipment selection, installation, customer service, and marketing. Such was the recent scene in Atlanta, the site of Habitech95.

Habitech is the home automation industry's only trade show. It hosts virtually every key player in the HA arena and is Mecca to HA dealers, installers, enthusiasts, and those just thinking of getting their feet wet. It's also the place to size up the state of the **industry** and see what's new.

The majority of exhibitors displayed independent subsystems that control a specific aspect of the home. Lighting systems, drapery controllers, audio/video distribution, HVAC controllers, security systems, and even central vacuum setups were represented. However, as Greyson Evans points out in his article in the HABC insert, for the **industry** to truly grow, we need a unified method of communication between these subsystems to enable them to interoperate.

There are a number of contenders wanting to facilitate that communication, and the question in everyone's mind continues to be, "Will it be X-10, CEBus, or LonWorks?"

X-10 continues to hold the lead in terms of price. There were even new X-10 products that address some of the shortcomings people have complained about for years. Powerline Control Systems (PCS) has a number of X-10 offerings that gradually brighten from off (rather than going to full on first, then dim), preset dim, and microdimming. While pricey, I think they'll be popular among X-10 diehards. They are also shipping now.

As for CEBus versus **LonWorks**, the jury is still out. Both camps had pavilions showing products with support for each built in. Greyfox presented a CEBus box that provides Node 0 functionality for coax and twisted pair in addition to routing services between power line, coax, twisted pair, and RF. Such a box serves as the core of any complete CEBus installation. I hadn't expected to see one produced for at least another year or two.

I certainly don't have room here to get into all of what was shown, but there is a Web page that offers extensive home automation information, including how to contact the companies I've mentioned here. Point your browser at <http://www.hometeam.com/> and be prepared to spend some time reading. It's good stuff.

Next year's show cohabits with CES Orlando: The Digital Destination and takes place May 23-25. Mark your calendar now.

# CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR  
Steve Ciarcia

PUBLISHER  
Daniel Rodrigues

EDITOR-IN-CHIEF  
Ken Davidson

PUBLISHER'S ASSISTANT  
Sue Hodge

TECHNICAL EDITOR  
Janice Marinelli

CIRCULATION MANAGER  
Rose Mansella

ENGINEERING STAFF  
Jeff Bachiochi & Ed Nisley

CIRCULATION ASSISTANT  
Barbara Maieski

WEST COAST EDITOR  
Tom Cantrell

CIRCULATION CONSULTANT  
Gregory Spitzfaden

CONTRIBUTING EDITOR  
John Dybowski

BUSINESS MANAGER  
Jeannette Walters

NEW PRODUCTS EDITOR  
Harv Weiner

ADVERTISING COORDINATOR  
Dan Gorsky

ART DIRECTOR  
Lisa Ferry

PRODUCTION STAFF  
John Gorsky  
James Soussounis

CONTRIBUTORS:  
Jon Elson  
Tim McDonough  
Frank Kuechmann  
Pellervo Kaskinen

CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0696-6965) 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 Circuit Cellar INK Subscriptions, P.O. Box 696, Holmes, PA 19043.9613 or call (600) 269-6301.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P O Box 698, Holmes, PA 19043.9613.

Cover photography by Barbara Swenson  
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

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

MIDWEST  
Nanette Traetow  
(708) 357-0010  
Fax: (708) 357-0452

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8bits, 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 responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK*.

Entire contents copyright © 1995 by Circuit Cellar Incorporated. All rights reserved. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

**1 4** **The Use of Color in Scientific Visualization**  
by *Mike Bailey*


**2 4** **Virtual Reality Position Tracking**  
by *Herschel1 Murry & Mark Schneider*


**3 0** **Digital Video Resizing and Compression**  
by *James Goel*

**3 6** **A PIC-based Motor Speed Controller**  
by *Chuck McManis*




**OUR BONUS SECTION: HOME AUTOMATION & BUILDING CONTROL  
BEGINS ON PAGE 47**

**7 4**  **Firmware Furnace**  
journey to the Protected Land: The Mystery of Scan Code Set 3  
by *Ed Nisley*

**8 2**  **From the Bench**  
Sacrifice for the Good of the Circuit  
Strengthening the Weak Link  
by *Jeff Bachiochi*

**8 6**  **Silicon Update**  
MIPS for the Masses  
by *Tom Cantrell*

**9 2**  **Embedded Techniques**  
Circuit Protection  
by *John Dybowski*

# INSIDE ISSUE 60

**2** **Editor's INK**  
Ken Davidson  
Habitech95

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

**8** **New Product News**  
edited by Harv Weiner

**100** **ConnecTime**  
Excerpts from  
the Circuit Cellar BBS  
conducted  
by Ken Davidson

**112** **Steve's Own INK**  
Your Computerized  
Future

**81** **Advertiser's Index**



# READER'S INK

## PNEUMONICS HELP MEMORY?

I think you folks need to take better care of Steve. All that overdosing on Papa Gino's pizza has affected his vocabulary.

Steve said he was writing pneumonics. My dictionary defines "pneumonic" as an adjective which refers to the pulmonary system or affected with pneumonia.

It seems Steve could use a mnemonic, as in ni-mo-nik, to keep his lungs clear. I hope so! Somewhat later he did write that he was "rationalizing these few pneumonics." Did that help him feel better? I sure hope so, because you produce a great magazine!

Joe Craig  
Ellicott City, MD

*While Steve readily admits that his spelling is about as good as any other engineer, this is certainly one we should have caught on the very first reading. His favorite programming language may be solder, but he really does know what a mnemonic is. Honest.-Editor*

## Contacting Circuit Cellar

We at Circuit Cellar *INK* 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, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.

**Phone:** Direct all subscription inquiries to (800) 269-6301.

Contact our editorial offices at (203) 875-2199.

**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](mailto:jeff.bachiochi@circellar.com). For more information, send E-mail to [info@circellar.com](mailto:info@circellar.com).

# ALL ELECTRONICS CORP.

QUALITY PARTS . DISCOUNT PRICES . FAST SERVICE . HUGE SELECTION

## INCREDIBLE SAVINGS! 30' CABLE

A GREAT DEAL!



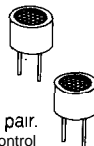
Originally made to connect Digital Audio Tape machine to a remote controller, this "snake" cable consists of four separate 9 conductor plus drain wire, foil-shielded cables in one jacket. The conductors are stranded 24 AWG wire. Each end of each smaller cable is terminated with DB-9P connectors. The cable is well-made and quite flexible for its size. Snake cable nominal O.D. is 0.52". Interior cable O.D. is 0.15". The outer jacket could be slit and removed if only the 9 conductor cable is required. DB-9 connectors include thumbscrew hold-downs. If you are using multiconductor shielded cable this is a great deal.

\$5.50 each

CAT# CBL-3

2 for \$10.00

## ULTRASONIC TRANSDUCER SET



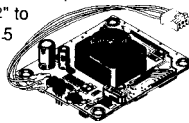
Murata-Erie #MA40A3R & MA40A3S  
40 Khz transmitter and receiver, matched pair.  
Band width: 4K Hz+. Ideal for remote control systems, burglar alarms, flow rate detectors, etc.  
0.64" diameter X 0.47" high.

CAT# UST-40

\$2.50 per pair

## MINIATURE B/W CAMERA

High-resolution, ultra-compact pc board CCD camera, 1.56" X 1.79" X 0.64". Equipped with pinhole lens which can view objects from 2" to infinity in extremely low light (.5 Lux @ F 1.6). Infrared sensitive works in total darkness with an infrared light source. Auto-iris for automatic light compensation. 12vdc (10-15vdc) @ 150 ma. operation.



\$159.00 each

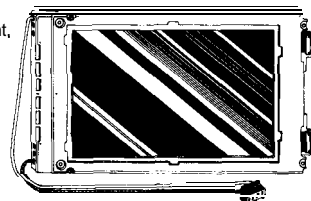
CAT# VC-200P

## 640 X 480 LCD PANELS WITH BUILT-IN DIGITIZER

Two sizes available, both originally designed for laptop computer/note pad. Built-in digitizer to be used with a stylus (not included) for hand written notations. Onboard drivers. Operates on 5 vdc (logic) and 18 vdc (LCD). Full documentation on both units.

### SHARP LM64P90

Built-in CCFT backlight, Overall dimensions: 10.19" x 7" x 0.35" Viewing area: 7.86" X 6" Dot size: 0.27mm X 0.27mm.

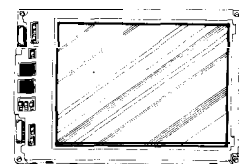


White dots on black background. CAT # LCD-31

\$40.00 each

### SHARP LM64194F

Overall dimensions: 3" x 5.75" x 0.25" viewing area: 6" X 4.5". Dot size: 0.21mm X 0.21mm Black dots on white background.



CAT # LCD-32

\$40.00 each

CALL, WRITE or FAX for a FREE 64 Page CATALOG Outside the U.S.A. send \$2.00 postage.

MAIL ORDERS TO:  
ALL ELECTRONICS CORPORATION  
P.O. Box 567  
Van Nuys, CA 91408  
FAX (818)781-2653

ORDER TOLL FREE  
**1-800-826-5432**  
CHARGE ORDERS to Visa, Mastercard or Discover

TERMS: NO MINIMUM ORDER. Shipping and handling for the 48 continental U.S.A. \$5.00 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include local state sales tax. Quantities Limited. NO COD. Prices subject to change without notice.



# NEW PRODUCT NEWS

Edited by Harv Weiner

## BASIC LANGUAGE PROGRAMMABLE CONTROLLER

Sylva Energy Systems introduces a low-cost, full-featured BASIC-programmable controller board suited for data logging, home automation, and industrial-control applications. The **BAC552** controller uses a Philips 80C552 microcontroller with a 14.7-MHz clock and is packaged on a 9" x 6" fully socketed board. The controller is application ready with onboard I/O and screw-terminal plug-on connectors.

The controller provides ten 5-A relay outputs, three LED outputs, sixteen optoisolated DC inputs, eight 10-bit analog inputs (4-20 mA or high impedance), two 8-bit analog outputs (0-5 V), and an RS-232 or RS-485 communications port. As an option, the board may be ordered with an X-10 interface controller, which provides full two-way power-line communications ability.

System expansion via the I<sup>2</sup>C bus offers up to 224 additional I/O points. An I<sup>2</sup>C operating system provides full master/slave communications between multiple BAC controllers interconnected with I<sup>2</sup>C bus extenders.

The user BASIC program resides in approximately 29.4 KB of SRAM, backed by a lithium battery and a MAX691. A 32-KB EPROM socket is available for permanent application program storage. A BASIC command generates an Intel hex file of the program in SRAM for EPROM programming. Full floating-point BASIC is expanded with statements supporting discrete-I/O-control-based applications. Interrupt sources include 8 of the 16 inputs, timer interrupt (1-65,535 s) and a communications interrupt.

Enhanced embedded controller functions include a system watchdog timer, real-time clock/calendar module, power-loss-duration calculation, warm-or-cold-boot determination, auto-program load on powerup or reset, error trapping, and a queued P R I N T statement for improved BASIC execution speed.

The BAC552 controller sells for \$299.95 U.S. with a comprehensive user's manual and the X-10 controller option.

Sylva Energy Systems  
519 Richard St.

Thunder Bay, ON • Canada P7A1R2 • (807) 683-6795 • Fax: (807) 683-6485

#500

## IDE/PARALLEL PORT INTERFACE

Palmtech announces two new interface chips that connect IDE (hard disk) drives to 8-bit processors using a minimum of space while providing additional I/O and interrupts. Originally developed for a Z180-powered single-board computer, the chips have a multitude of applications.

The **PT IDE802/803** are single-chip IDE interfaces for 8-bit CPUs. The chips incorporate an additional 8-bit unidirectional or bidirectional parallel port plus 10 (11 for the IDE803) handshake or extra I/O lines suitable for a printer port. The chips also include a controller for three interrupts (two for the '803). The 8-to-16-bit data conversion is fully transparent with an IDE sector accessed as 512

8-bit bytes rather than 256 16-bit words. The PT IDE80x accepts any IDE drive conforming to the CAM ATA standard and does not require external buffering to the drive or printer. It can be coupled to most 8-bit CPUs with little or no extra glue logic. The chip is available in a 68-pin PLCC package, requires 100 mA at 5 V, and adds only 75 ns to the IDE drive access time. Also included on the

chip is a speaker or general-purpose output.

The PT IDE80x sells for approximately \$41 in single quantities.

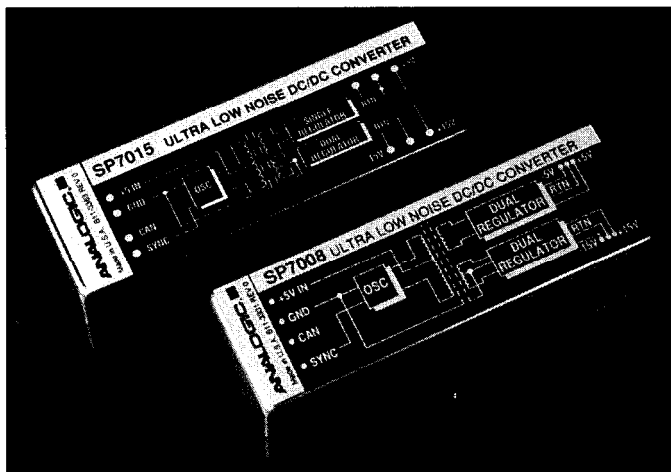
**Palmtech**  
Moonah & Will St.  
Bouliia, QLD 4829  
Australia  
+6177 463-109  
Fax: t6177 463-198

#501

# NEW PRODUCT NEWS

## LOW-NOISE DC/DC CONVERTER

Analogic announces a family of state-of-the-art exceptionally low-noise DC/DC converters designed specifically for high-performance data-acquisition applications. The family, models SP7005, SP7008, and SP7015, provides isolated  $\pm 15$ -V and +5-V analog supplies from a +5-V input and has excellent line-and-load regulation. In addition to data acquisition, these converters can be used in mixed signal circuits and in telecommunications.



This family of DC/DC converters features low noise and ripple of 5-mV peak-to-peak under a full load with a line-and-load regulation of  $\pm 0.2\%$ . Models

SP7005 and SP7008 offer additional -6-V and -5-V supplies, respectively, and provide 6 W of power. Model SP7015 supplies up to 6.75 W of power. Each

product is packaged in a 1" x 3" x 0.5" fully shielded module with input-to-output isolation of 10 M $\Omega$  and 500 VRMS. An optional sync input is available to blank switching during analog-to-digital conversion.

Models SP7005, SP7008, and SP7015 are priced at \$148.50 each in 100-piece quantities.

Analogic Corp.  
8 Centennial Dr.  
Peabody, MA 01960  
(508) 977-3000 Ext. 2170  
Fax: (617) 245-1274

#502

## PORTABLE DATA ACQUISITION SYSTEM

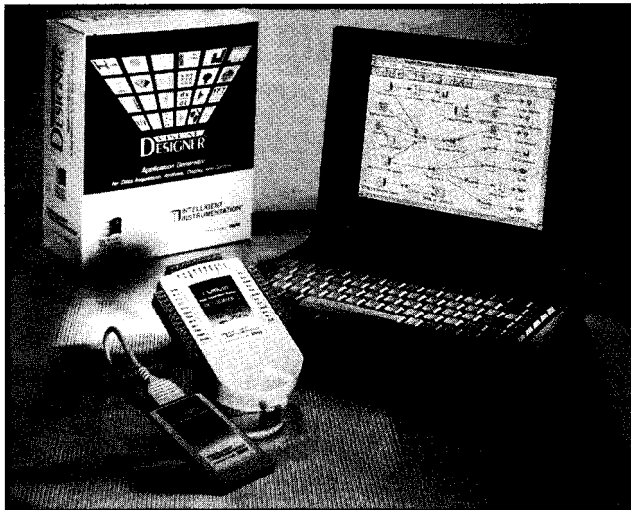
I/Ocard, a portable data acquisition system for the PCMCIA bus, has been launched by Intelligent Instrumentation. Designed for notebook PCs, the system requires only 1 W of power, making it ideal for remote applications. I/Ocard connects to a PCMCIA type II slot and consists of a PC card and a termination pad.

The card features 8 differential analog inputs with 12-bit resolution, 30-kHz throughput, and external triggering. Gains and input ranges are software selectable. Both unipolar and bipolar input ranges are available.

Two models of I/Ocard are available. Model PCI-460P-1 provides gains of 1, 10, 100, and 1000 for extremely low-level signal measurement. PCI-460P-2 provides gains of 1, 2, 4, and 8. The system also features four TTL inputs, four TTL outputs, cold-junction compensation for direct thermocouple connection, and a voltage-reference output for powering sensors

I/Ocard is fully supported by Visual Designer, Intelligent Instrumentation's Windows-based application-generator software. Users can easily develop their own applications by drawing block diagrams (flowgrams) rather than coding the applications with a language such as C, Pascal, or BASIC. Sliders, switches, numeric inputs, and user prompts control the execution of the application. Displays include fully customizable plots, instrument panels, and control panels.

The I/Ocard portable data acquisition system sells for \$595. Additional termination pads are available for \$225.



Intelligent Instrumentation, Inc.

6550 S. Bay Colony Dr., MS130 • Tucson, AZ 85706 • (520) 573-3504 • Fax: (520) 573-0522

#503

# NEW PRODUCT NEWS

## DIGITAL VIDEO ENCODER

Philips introduces a digital MPEG-compatible video encoder. The **SAA7185** encodes digital YUV data to an NTSC or PAL CVBS and S-video analog signal to be displayed on a TV or recorded on a VCR. The SAA7185 is designed for use in video-processing equipment such as computers, video servers, and video CD players. Because it accepts 16-bit YUV data or 8-bit CCIR 656-compatible YUV data in MPEG format, it is ideal for CD playback in PCS.

The SAA7185 is a highly flexible and easily programmable 5-V CMOS device. It is controlled via an I<sup>2</sup>C serial interface or an 8-bit microprocessor port and can be synchronized as master or slave to external devices.



The chip produces European PAL and either the U.S. or Japanese version of the NTSC signal. NTSC-M and PAL B/G standards and substandards are supported. The SAA7185 also provides 8-color onscreen display and provides closed-caption encoding. The chip contains cross-

color-reduction circuitry and IO-bit oversampled DACs to improve image quality.

The SAA7185, in 68-pin PLCC packages, sells for less than \$7 in volume.

Philips Semiconductors  
811 E. Arques Ave. • Sunnyvale, CA 94088-3409  
(408) 991-3737 • Fax: (708) 635-8493

#504

## A Serious Imaging Solution



IMPACT Professional is a complete image analysis system that includes a broad range of cross-discipline tools grouped into eight separate processing environments. These modules include:

- Image Processing System
- 2-D Analysis System for Edge, Shape and Surface Characterization
- Object Counting, Analysis and Sorting System
- Neural Network Graphic Recognition and Classification System
- Image Comparator and Anomaly Detection System
- 3-D Mapping System including Stereo- and Random-dot Projections
- Fractal Analysis System for Point, Line, Edge and Surface Processing
- Color Mapping System which handles Greyscale, Pseudo and 24-bit RGB

**SYSTEM REQUIREMENTS:** PC/AT or compatible, 386, 486 or Pentium, with at least 16 MB of RAM and a hard disk DOS 3.1 or higher. Uses a flat memory model with its own extender and Virtual Memory Manager capable of addressing 4 gigabytes of memory. A super VGA video card.

TARDIS Systems **FREE DEMO** Phone: (505) 662-9401  
P.O. Box 1251 Fax: (505) 662-6780  
Los Alamos, NM 87544 U.S.A. Technical Support: (505) 662-5623

#105

## \$129.95 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

6051 CODE

COMPATIBLE.

ADD A KEYPAD

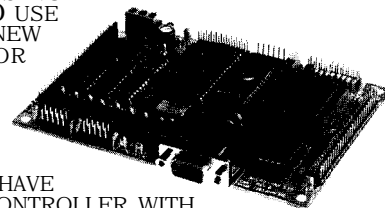
AND AN LCD

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
- OPTIONAL 4 CHANNEL, 8 BIT D/A
- UP TO 4, 16 BIT TIMER/COUNTERS WITH PWM
- UP TO 3 RS232/485 SERIAL PORTS
- BACKLIT CAPABLE LCD INTERFACE
- OPTIONAL 16 KEY KEYPAD & INTERFACE
- 160K OF MEMORY SPACE, 64K INCLUDED
- 8051 ASSEMBLER & MONITOR INCL., BASIC OPT.



1985-1995  
10  
YEAR  
ANNIVERSARY

**EMAC, inc.**

618-529-4525 Fax 457-0110 BBS 529-5708  
P.O. BOX 2042, CARBONDALE, IL 62902

#128



# NEW PRODUCT NEWS

## PLUG-AND-PLAY MICROCONTROLLER

Silicon Systems introduces the **73M2918**, an 8052-compatible microcontroller with virtual 550 UART and built-in hardware which supports the emerging Plug-and-Play ISA standard. This high-performance microcontroller has all the attributes of an 8052 8-bit microprocessor, including instruction cycle time, UART, timers, interrupts, 256 bytes of RAM, and programmable I/O.

The 73M2918 also includes an HDLC packet-generation unit

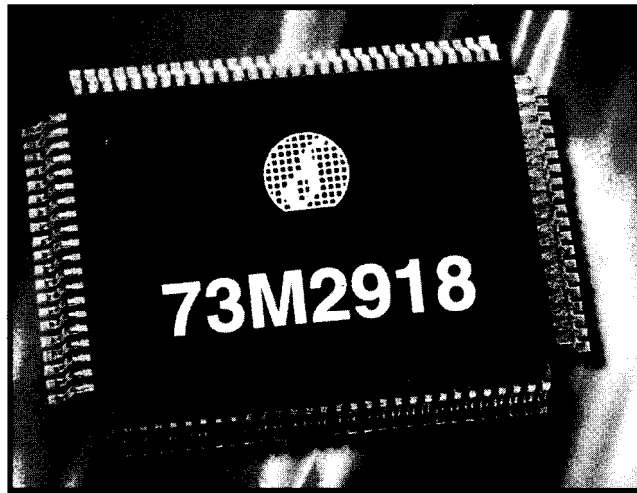
along with the registers and hardware required to facilitate Plug and Play. The device features 32 pins of additional user programmable I/O. Programmable bank- and chip-select logic reduces the need for board-level glue logic. The unit has two buffered clock outputs to support peripheral devices such as UARTs and modems and two general-purpose input pins with programmable wake-up capability.

The device operates at a speed of 33 MHz at 5 V. An optional version, the **73M2918A**, operates at 44 MHz for high-speed applications. The device is offered in a small-form-factor 100-

lead QFP package. In quantities of 1000, the 73M2918 is priced at \$10.24 and the high-speed 73M2918A at \$14.92.

Silicon Systems  
14351 Myford Rd.  
Tustin, CA 92680  
(714) 573-6200  
Fax: (714) 573-6914

#505



Don't bang your head and bust your development tool budget with inferior tools. Think smart and use **Paradigm LOCATE**. With today's compressed development cycles, you can't afford not to have the most powerful, full-featured locate utility speeding your '186 or V-series design to market.

Faster, more capable and supporting all Borland and Microsoft C/C++ compilers, only **Paradigm LOCATE** can jump the embedded system debugging chasm with access to **Paradigm DEBUG**.

Don't be caught flat-footed when you can be fast and nimble. Get Paradigm for all the tools and development support you need.

## PARADIGM

Nuff said.

Proven Solutions for Embedded C/C++ Developers

**1-800-537-5043**

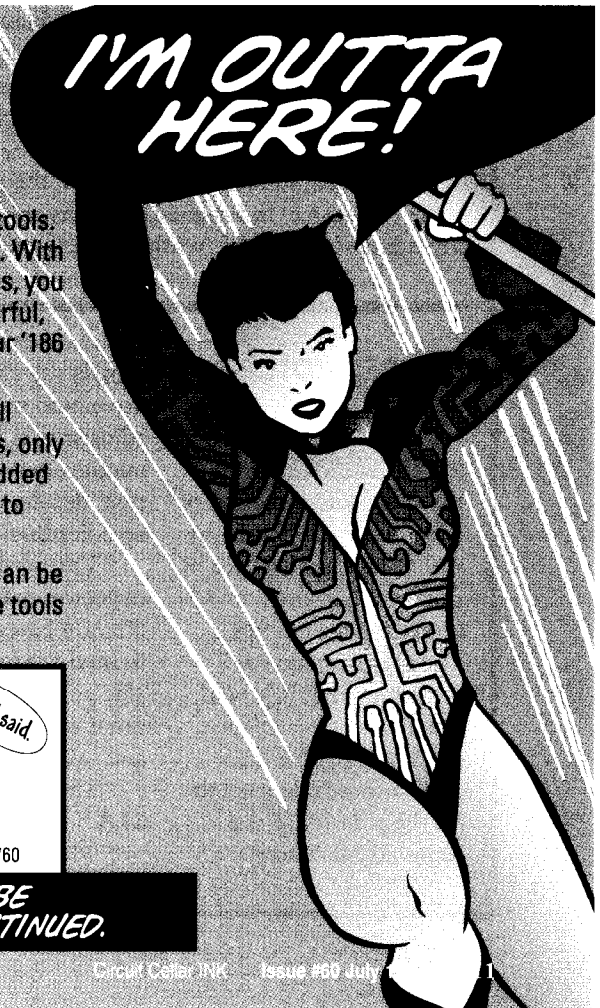
Paradigm Systems  
3301 Country Club Road, Suite 2214, Endwell, NY 13760  
(607) 748-5966 / FAX: (607) 748-5968  
Internet: 73047.3031@compuserve.com

**TO BE  
CONTINUED.**

#106

©1995 Paradigm Systems, Inc. All rights reserved.

Circuit Cellar 11K Issue #60 July



# NEW PRODUCT NEWS

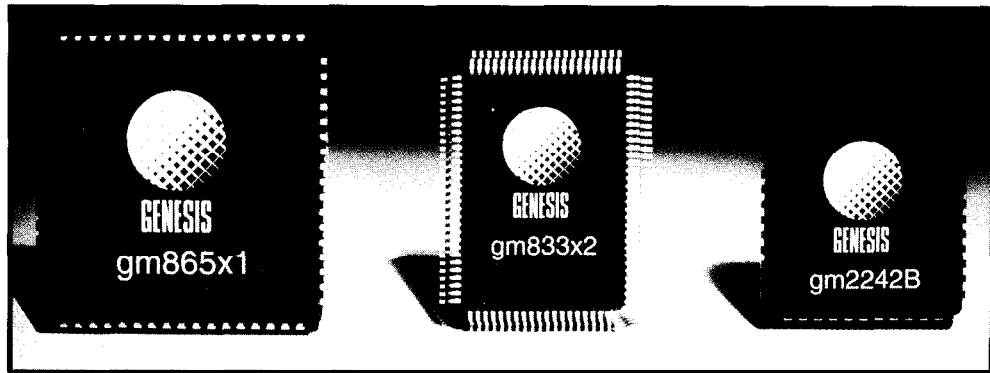
## VIDEO/IMAGE DSP CHIPS

Genesis Microchip announces full-volume production of three video/image DSP chips: the **gm865x1** and **gm833x2** video/image resizing engines and the **gm2242B** half-band filter.

The gm865x1 and gm833x2 provide a revolutionary breakthrough in image-resizing technology and are part of the Acuity Resizing series of real-time, 2D filtering engines. Acuity Resizing devices produce the high-quality scaled images while minimizing the undesirable aliasing, artifacts, and distortion often created during digital-video scaling. Both chips use patented algorithms and architectures and implement advanced interpolation and finite-impulse-response (FIR) filtering. In reduction mode, all memory required for FIR filtering is provided on-chip.

The gm865x1 offers up to 65-tap vertical and horizontal filtering independently in both directions. As a top-of-the-line part, it produces the highest quality resized images possible. Applications benefiting from the chip's performance include medical imaging, LCD projection systems, and high-end broadcast equipment which can take advantage of the chip's dynamic horizontal resizing for special effects.

For more cost-sensitive systems, the gm833x2 (an up to 33-tap device) is designed for many applications including videographic workstations, multimedia



systems, and projection and scan-conversion equipment. Both chips perform shrink-and-zoom operations.

The gm2242B half-band filter is fully compatible with the industry-standard TMC2242B part and offers more features at significantly less power and substantially less cost. Its unique features include a user-selectable  $\sin x/x$  compensating filter and a handy pass-through mode. Constant data latency, available in all operating modes, is also available. Half-band filters double or halve digital-signal sampling rates and simplify ADC and DAC subsystem design. Applications ranging from broadcast and teleconferencing systems to digital compression and encoder equipment often use half-band filters.

Genesis Microchip, Inc.  
200 Town Centre Blvd., Ste. 400  
Markham, ON  
Canada L3R8G5  
(905) 470-2742 • Fax: (905) 470-2447

#506

## STEPPER MOTOR CONTROLLER

The **ServoStep** from MicroKinetics controls the speed and direction of a stepper motor from any  $\pm 10$ -VDC signal source or from a single slide or rotary potentiometer. The step output is a frequency that is proportional to the magnitude of the input. The direction output is dependent on the polarity of the input signal. These

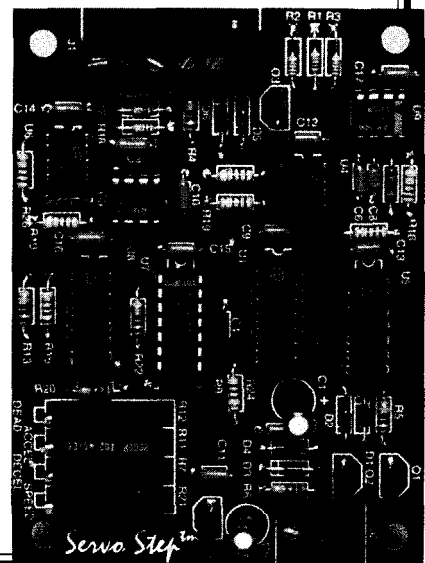
signals run the stepper driver and motor. Onboard adjustments include maximum speed, acceleration, deceleration, and a zero deadband adjustment.

The outputs are open collector, making them directly compatible with industry-standard drivers. The power supply requirement is +12 VDC at 100 mA and -12 VDC at 50 mA.

The ServoStep sells for \$199.

MicroKinetics Corp.  
1220-J Kennestone Cir.  
Marietta, GA 30066  
(404) 422-7845  
Fax: (404) 422-7854

#507



# FEATURES

**14** The Use of Color in Scientific Visualization

**24** Virtual Reality Position Tracking

**30** Digital Video Resizing and Compression

**36** A PIC-based Motor Speed Controller

# FEATURE ARTICLE

**Mike Bailey**

## The Use of Color in Scientific Visualization

Getting color right is no easy task! After reviewing a little physiology, Mike looks at how we define colors in computers. Final tips ensure information does not get construed by color choice.

**O**o some, a discussion of color in scientific visualization seems unnecessary. But, experience shows that some ways of using color communicate information more effectively than others. Used incorrectly, color detracts, providing less information than if it had not been used at all.

This article does not explain everything there is to know about color physics and human vision. Instead, after an overview of some these aspects, I go right to the qualitative issues of what color is and how it should be handled.

### COLOR FREQUENCIES AND WAVELENGTHS

The electromagnetic spectrum is infinitely large. At the low frequency

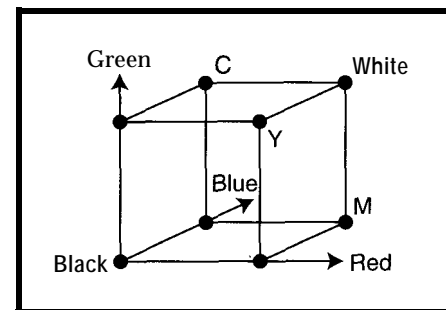


Figure 1—RGB color space is represented as an orthogonal axes.



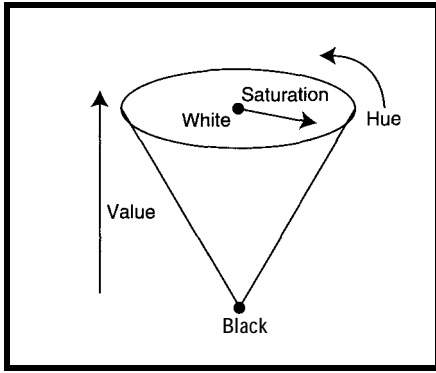


Figure 2—The Hue-Saturation-Value color cone is easier for humans to understand and work with.

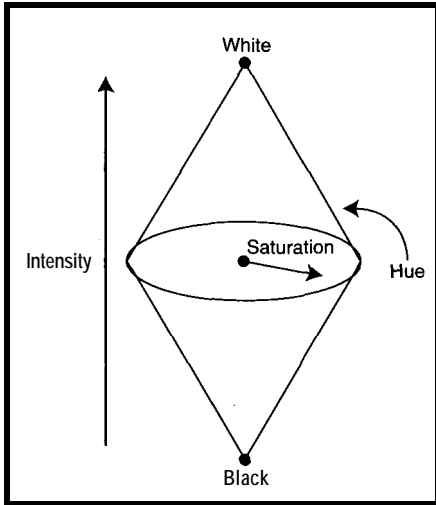


Figure 3—in the Hue-Saturation-Intensity cone, color intensity is also accounted for.

end, it contains radio waves and microwaves. At the high end, it contains x-rays and gamma rays. Rather than speak of frequency, electromagnetic waves are usually discussed in terms of wavelength,  $\lambda$ , which is inversely proportional to the frequency,  $f$ :

$$\lambda = \frac{c}{f}$$

where  $c$  equals the speed of light (i.e.,  $3 \times 10^8$  m/s). Our eyes happen to be sensitive to the range of wavelengths within 380-780 nm. We see the long wavelengths around 780 nm as red and the short wavelengths around 380 nm as deep blue-purple.

## THE EYE

The human eye is a marvelous input device, designed to meet a certain set of everyday demands. Like many parts of the human body, the eye is not a single component. Rather, it has several parts, each with its own

area of specialization.

Rods are retinal sensors that detect grayscale and low levels of light. A typical human retina has approximately 115 million rods, mostly sensitive in the 500 nm (-green) range. Because there are so many, rods are much better at detecting high spatial frequencies than cones.

Interestingly, rods are concentrated near the retina's periphery. Peripheral vision is therefore much more sensitive to small light changes than straight-on vision. As a result, some people detect CRT flicker only out of the corner of their eye. Straight-on, the flicker is no longer there.

Cones enable us to see color. Approximately 8 million cones concentrate near the center of the retina (the fovea), where their density is about 150,000 per square millimeter. Thus, color vision is far more sensitive to objects directly in front of us.

Cones are categorized by the wavelengths they are sensitive to. Low, medium, and high frequencies are viewed by L, M, and H cones, which achieve maximum sensitivity at 570, 550, and 440 nm, respectively. These wavelengths loosely correspond to the red, green, and blue portions of the spectrum.

It would be nice if the brain approximated red, green, and blue signals in a 24-bit (or more) frame buffer. Un-

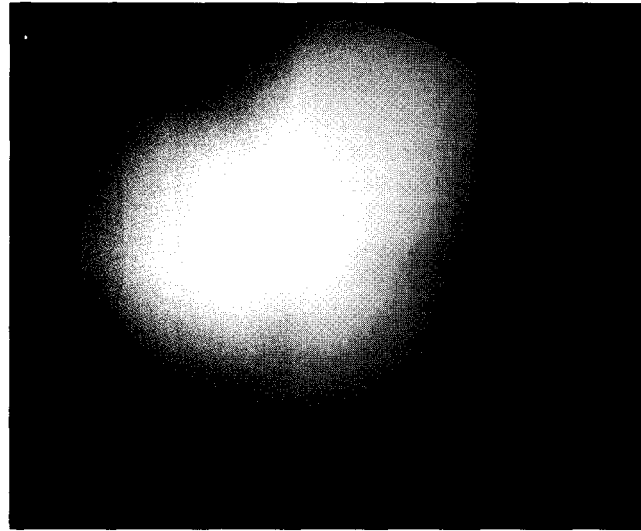


Photo 1—Here's the top circle of the Hue-Saturation-Value color cone.

fortunately, it is more complicated than that.

Three signals do go to the brain but as combinations of the L, M, and H cone signals. Specifically, they are:

1. L - M
2. S - (L + M)
3. L + M

If we permit ourselves the luxury of relating L, M, and H to R, G, and B (even though this is not exactly right), the three signals become:

	R	G	B	Y
Black	0.0	0.0	0.0	0.00
White	1.0	1.0	1.0	1.00
Red	1.0	0.0	0.0	0.30
Green	0.0	1.0	0.0	0.59
Blue	0.0	0.0	1.0	0.11
Cyan	0.0	1.0	1.0	0.70
Magenta	1.0	0.0	1.0	0.41
Orange	1.0	0.5	0.0	0.60
Yellow	1.0	1.0	0.0	0.89

Table 1—Using the luminance equation, it is relatively easy to determine the luminances of standard colors.

	Black	White	Red	Green	Blue	Cyan	Magenta	Orange	Yellow
Black	0.00	1.00	0.30	0.59	0.11	0.70	0.41	0.60	0.89
White	1.00	0.00	0.70	0.41	0.89	0.30	0.59	0.41	0.11
Red	0.30	0.70	0.00	0.29	0.19	0.40	0.11	0.30	0.59
Green	0.59	0.41	0.29	0.00	0.48	0.11	0.18	0.01	0.30
Blue	0.11	0.89	0.19	0.48	0.00	0.59	0.30	0.49	0.78
Cyan	0.70	0.30	0.40	0.11	0.59	0.00	0.29	0.11	0.19
Magenta	0.41	0.59	0.11	0.18	0.30	0.29	0.00	0.19	0.48
Orange	0.60	0.41	0.30	0.01	0.49	0.11	0.19	0.00	0.30
Yellow	0.89	0.11	0.59	0.30	0.78	0.19	0.48	0.30	0.00

Table 2—Luminance differences offer a scientific measurement of contrast. The greater the luminance difference, the greater the contrast. A good contrast is 0.40 luminance difference.

1. red-green
2. blue-(red+green)
3. red+green

If yellow is substituted for red+green, the second signal becomes blue-yellow. Thus, the eye-brain system distinguishes chromaticity by the red/green and blue/yellow difference.

Knowing that blue only contributes about 11% to the overall appearance of brightness of a color, color scientists treat the third quantity red+green as luminance. So, the third signal becomes overall luminance.

The red-green signal is where most color deficiency occurs. For example, people with red/green color blindness do not correctly produce the red-green signal. Although blue, yellow, and overall luminance is understood, the person cannot determine if something is more red than green or vice versa.

## DEFINING COLOR DIGITALLY

Too often, we ask computers for color in a way suitable to them: red, green, blue (RGB). Integers are fed to the digital-to-analog converters of the color guns.

This method uses a rectilinear color space where red, green, and blue are the principal axes. Black is at the origin and white is at the other end of the major diagonal. As Figure 1 illustrates, the complementary colors of

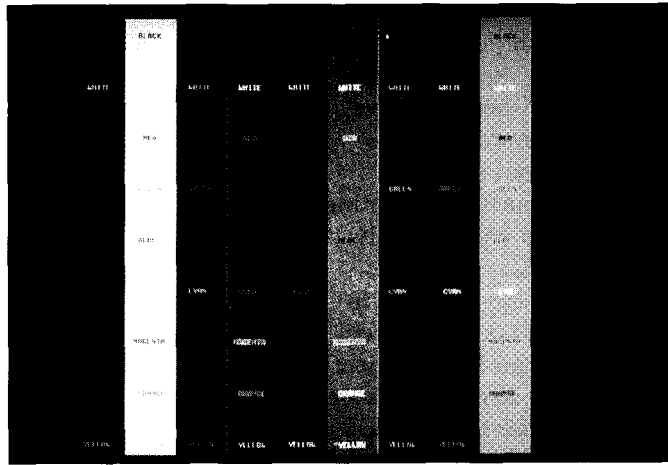


Photo 2—Luminance differences shown in Table 2 take on a clearer meaning when shown graphically.

cyan, magenta, and yellow are at the remaining corners.

Although this is convenient for computers, humans don't think this way. It's easier to specify color with the Hue-Saturation-Value (HSV) system, an inverted cone shown in Figure 2. Black is at the bottom tip and white forms the base's center. The circumference includes colors with at most two components. Colors with three components are found within the cone's volume.

If you start on the base circumference and increase the missing color

component, the color moves closer to white and saturation decreases. Decreasing one or two components decreases the value or brightness. Traveling around the circumference changes the hue. Photo 1 shows the top circle of the HSV color cone.

The Hue-Saturation-Intensity system is another fairly easy way to specify color (see Figure 3). The two cones join at their bases with black being at the bottom tip and white at the top. Equatorial colors have at most two components (e.g., red or red + green = yellow). Colors with three

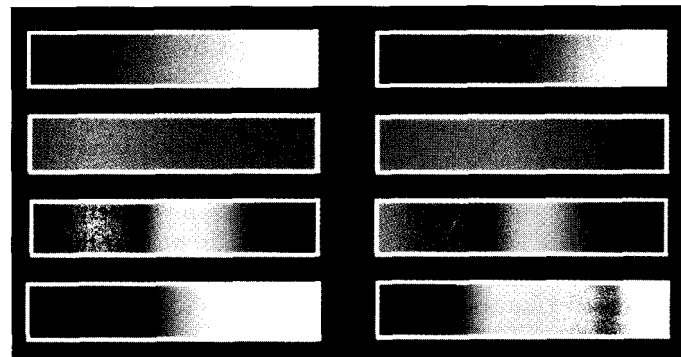


Photo 3—A gallery of color interpolation functions. Grayscale, intensify, saturation, two-color, rainbow, rainbow with luminance modifications, heated object, and optimal are represented starting in the upper left and moving left to right and top to bottom.

components are found in the volume of the double cone.

Doing a conversion from HSV or HSI to RGB is reasonably straightforward [1].

## THE LUMINANCE EQUATION

When displaying information on top of other information or a background (e.g., with text and graphs), it is important to get good intensity contrast. The luminance equation determines good contrasts:

$$Y = 0.30 \times \text{red} + 0.59 \times \text{green} + 0.11 \times \text{blue}$$

Table 1 shows standard colors and their luminances. Individual color component intensities have been normalized from 0.0 to 1.0, instead of the more familiar 0-255.



Photo 4—Accidental meaning is added by quantizing the colors in a visualization display.

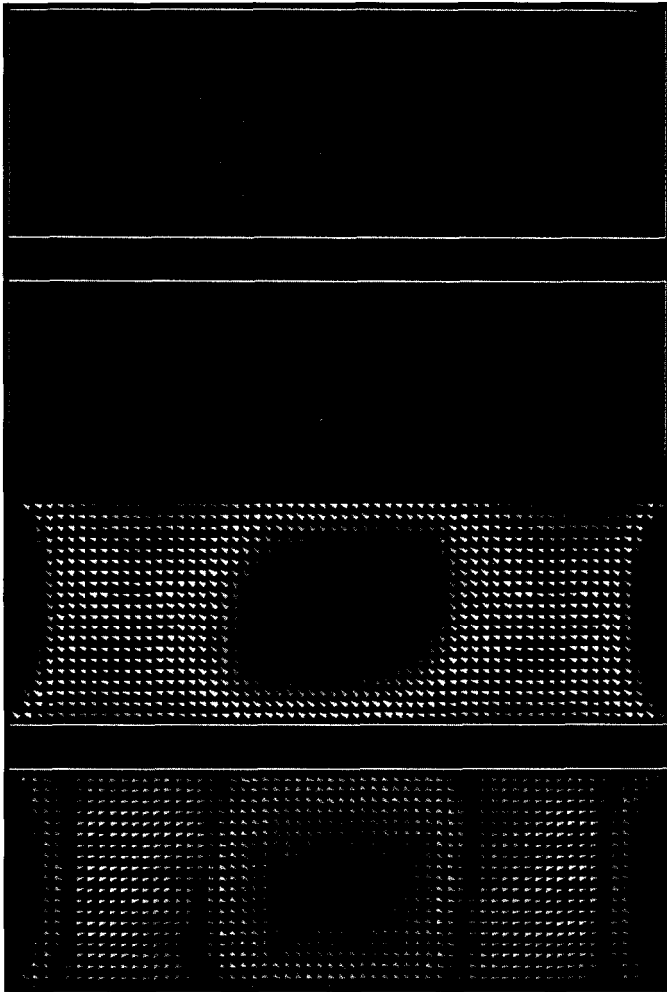


Photo 5-In the first photo, arrows indicate the direction of the magnetic field. The second photo shows the same arrows with color added using a rainbow scale keyed from the sine of the arrow angle.

Table 2 offers a first-order approximation of contrast by taking the difference between the luminances of a foreground and background color. The greater the Aluminance, the greater the apparent contrast. A good threshold value for contrast is a Aluminance of about 0.40. This number, of course, varies between people and lighting conditions. Photo 2 depicts a color version of Table 2.

As the luminance equation implies, RGB color space is not perceptually uniform. A green of (0,1,0) looks brighter than a blue of (0,0,1).

Neither the HSV nor the HSI color space is perceptually uniform either.

On the circumference of the color cone, green looks brighter than blue. If perceptual uniformity is important, it is better to ask for color in the CIELAB or CIELUV color space. [2]

## COLOR ATTRIBUTES

In choosing color, be aware of established cultural or professional meanings for certain colors. Any visualization portraying set qualities is less effective if it fights the colors most viewers associate with these qualities.

For example, red indicates stop, on, off, hot, dangerous, high stress, oxygen, shallow, and money loss, depending on the application.

## COLOR INTERPOLATION

Just as color represents ranges of scalar values such as temperature or stress, good ways must be chosen to interpolate colors in an intuitive and

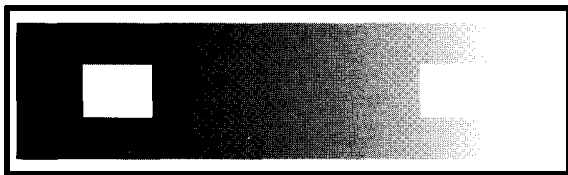


Figure 4-In this simultaneous contrast illusion, the white square on the left looks lighter than the white square on the right. In fact, they are both the same. Only the surrounding colors differ.

# RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

CONNECTS TO RS-232

**AR-16 RELAY INTERFACE (16 channel)..... \$ 69.95**  
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, 10 amp)..... \$ 44.95**  
**RD-6 REED RELAY CARD (6 relays, 10 VA)..... \$ 49.95**  
**RH-6 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/8 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 A/D configurations and 12 bit converters (terminal block and cable sold separately).

**ADC-8E TEMPERATURE INTERFACE\* (6 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.

**IPS-4 PORT SELECTOR (4 channels RS-422)..... \$ 79.95**  
Converts an W-232 port into 4 selectable RS-422 ports.  
CO-498 (RS-232 to RS-422/RS-485 converter)..... \$ 44.95

\*EXPANDABLE...expand your interface 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 900 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 43215.5436



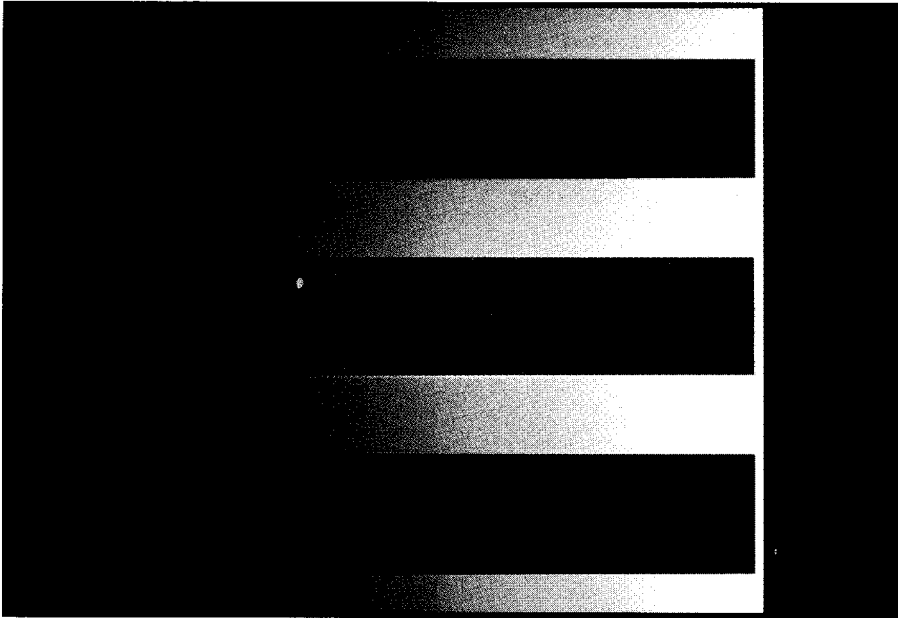


Photo 6—In this simultaneous contrast illusion, you can see how red, green, and blue change as background luminance changes.

unambiguous way. There are many ways to do this.

- grayscale interpolation

The simplest way to interpolate color is through grayscale, which is easy to produce—mix equal amounts of red, green, and blue. However, many resist grayscale interpolation because it is not sufficiently flashy.

But, grayscale interpolation has a major advantage. It faithfully reproduces a range of scalar values without assigning any preconceived ideas about the order. Physicians using computer graphics dislike color interpolation for x-ray photographs since color can imply meaning where there is none. For example, a bright red area draws atten-

tion even though it is no more significant than any other color.

Grayscale's major disadvantage is that the eye only sees a limited number of shades per hue. Multiple hues offer more scalar values.

- HSV and HSI (rainbow) interpolation

To get a reasonable color range for displaying scalar values, interpolate hues in the HSV or HSI color spaces. You can hold the saturation and value or intensity constant and linearly interpolate the hue.

Typically, this interpolation begins at blue, passes through green, and ends at red. The blue-green-red path is popular because it approximates the color order of the electromagnetic

spectrum that everyone sees in a rainbow. The direction blue-green-red or red-green-blue is determined by the inherent meaning of those colors in that particular application.

However, this method has problems. The best-looking colors on the monitor are fully saturated, but saturated colors cause problems with hardcopy and video devices. Typically, saturation should be held at 0.80 or less during the interpolation.

As well, different hues are perceived to have different intensities. Yellow is seen as the brightest, most important color (not red). If this distortion is a problem, use the luminance equation with the HSV or HSI equations to achieve a constant luminance instead of constant value or intensity.

This interpolation method also leaves large ranges of scalar values mapped to similarly perceived colors. You can make this clearer by having every nth scalar value map to black or white, creating contour lines to distinguish scalar values. This technique is especially effective in dynamic displays, but double check that you don't accidentally add meaning through color discontinuities.

- saturation interpolation

This method gives a color scale from unsaturated gray to a fully saturated color and is convenient when the hue carries other informational meaning and cannot be modulated.

Fully saturated areas are most colorful and draw most of our attention. Map the most important scalar

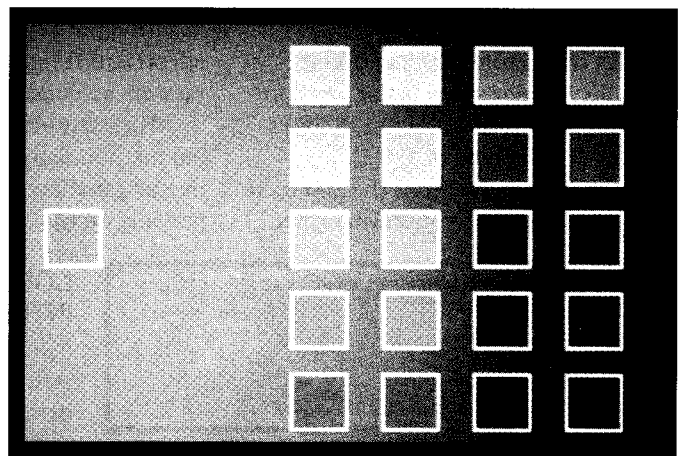
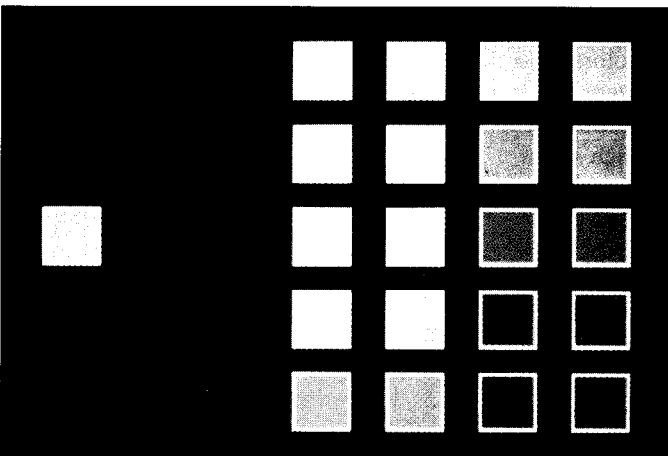


Photo 7—Because of simultaneous contrast, identical color which are further apart spatially are more likely to appear different. Although it is fairly easy to identify the colored square's match on the right in the first photo, it is next to impossible in the second.

values here. For example, information on forest fires could overlay an existing colored map. If saturation is inversely proportional to the amount of burn, severely burned areas are gray or nearly gray and untouched areas retain their full map color.

- intensity interpolation

You can also hold the hue and saturation constant to interpolate the intensity, which produces a color scale running from a dark version of a color to a light version of the same color (e.g., black to white).

This method is seldom used in scientific visualization because too often the difference in intensity is mistaken for 3D light-source shading (2D images with significant intensity variations tend to look 3D).

Use this interpolation method carefully and only with good reason!

- two-color interpolation

Sometimes, it's an advantage to show the variation of a scalar variable by interpolating between two colors. For example, with a terrain map, a forest (green) and a desert (brown) uses natural colors to correspond to scalar values. Interpolating the brown and green indicates forested land.

- heated-object color scale

When an object is heated, its color passes through a range of frequencies from red to yellow to white otherwise known as the *heated-object color curve*. It is used, among other things, to determine the temperature and motion of stars. That range of colors quite effectively encodes scalar information [3].

- optimal color scales

Levkowitz and Herman [3] also use a color scale passing through red, yellow, and light blue while moving from black to white. This color scale optimizes the maximum number of perceptually equal changes from black to white and distinguishes the maximum number of scalar values in a many-valued image such as a CAT scan or satellite image.

Photo 3 shows a gallery of color interpolation functions. However, it is

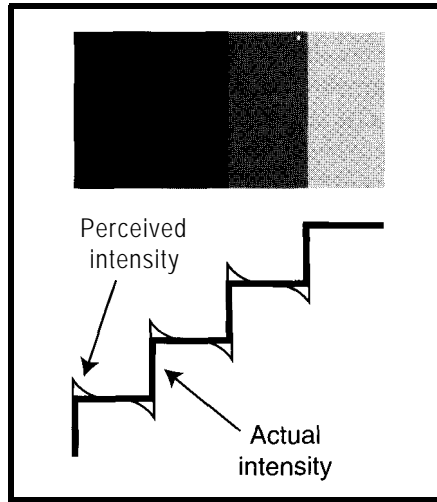


Figure 5—With the mach banding illusion, the eye sharpens each edge by making each band appear brighter on the left and dimmer on the right. In fact, the intensity is constant within each band.

sometimes necessary to interrupt the continuity of color interpolation because your system has only 256 colors or you need to add black or white colors to create contour lines. In either case, a less-than-continuous change of colors might be interpreted as a dramatic change in the scalar values.

Photo 4 shows what can happen when 16.7 million colors are quantized to 256. Note that the boundary between scalar values in the quantized image appears much sharper than it really is.

## REINFORCING INFORMATION

Color has an enormous impact on the way people perceive information. Here are some principles that you need to bear in mind while using color.

- color indicates patterns

Because color can be perceived as a global pattern, using color to reinforce smaller details often reveals new patterns in the data.

For example, a large vector field could be displayed as a collection of arrows where each arrow's size and direction shows the value of the vector field at that point. It is difficult to look at such a display and understand the overall pattern of vector magnitudes. Color coding the vectors (e.g., low velocity marks one end of the color

# Byte Craft



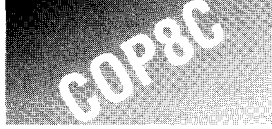

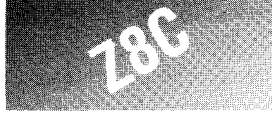

## C Compilers of choice

- Fast, efficient optimizing compilers
- Chip specific
- Built-in assembler
- Integrated Development Environment
- Linker, libraries

Optimizing C compilers for your single chip designs.

We respond to your C compiler needs!

**BYTE CRAFT LIMITED** Byte Craft Limited  
421 King St. N., Waterloo, Ontario  
CANADA N2J4E4  
(519) 888-6911  
Fax: (519) 746-6751  
BBS: (519) 888-76;

spectrum and high velocity the other) reveals more of the overall pattern.

Photo 5 enables viewers to better understand the distribution and direction of a magnetic field.

- limit the number of colors

If viewers need to discern information quickly or for absolute color discrimination, limit the total number of colors. Too many colors cause a viewer to overanalyze the information (at best) or get it wrong (at worst). Studies show that approximately seven colors is the optimal number for easy memorization and discernment.

- surrounding change colors

Simultaneous contrast is a well-known effect which states that our perception of a color is tainted by surrounding colors.

In Figure 4, the white square on the left looks lighter than the white square on the right. Photo 6 shows another example of the simultaneous contrast illusion. Here, you can see how red, green, and blue change as background luminance changes. These examples reinforce the argument for using fewer colors when quick discrimination is needed.

- compare adjacent colors

Because of simultaneous contrast, the farther apart two colors are spatially, the more likely it is that intervening colors cause them to appear different, even if they are identical.

Photo 7 shows a pathological example of this. A colored square is meant to be matched to one of the squares on the right. With a solid background, it is not hard to do. In the second photo, the background has been replaced with one of changing color. The differences in background, combined with the distance between the colors to be matched, make comparison more difficult.

- colors change with area size

Perception of a color depends on its area. In particular, our ability to discriminate colors diminishes with size. This is especially true with saturated blues, which should be avoided for small objects.

Color standards know about this effect and thus fix the size of the color area presented to test subjects. Typically, this is around 2" of subtended arc (2" is about how large your thumb appears at the end of your outstretched arm. A full moon subtends an arc of about 0.5").

If you don't believe that perception of color changes with the size of the colored area, remember the paint swatch you liked and the painted wall you didn't.

- color changes with ambient light

High ambient light tends to desaturate the appearance of colors, particularly yellows.

- the ability to discriminate colors changes with age

You're not going to change this much—just use fewer colors.

- beware of afterimage anomaly

With all-green calligraphic CRTs, there was a phenomenon known as *pink eye*. After staring at a green screen for a while, the victim would rise, look around, and see nothing but pink. This afterimage of a particular color is determined by its complement in either the Red-Green or Blue-Yellow pair.

While this is fun to experiment with, it can seriously jeopardize conclusions in scientific visualization and reinforces the need for fewer colors. The more difference between display colors, the less likely an afterimage will cause one color to be mistaken for another.

- beware the anomaly of mach banding

The human visual system tries to automatically increase edge sharpness. Nowhere is this more obvious than in a series of intensity bands shown in Figure 5.

Even though a spectracolorimeter shows that the intensity is constant within each band, the eye sharpens each edge by making each band appear

brighter on the left and dimmer on the right. We perceive a cusping effect as would be seen in a Greek column.

Mach banding also shows up when smooth-shading polygons. Even though the intensity is interpolated within each polygon, at the polygon borders there is a first-derivative discontinuity in intensity so that bright or dark lines appear along polygon boundaries.

Since mach banding interferes with the eye's ability to discriminate intensity differences, don't expect crucial decisions to be made based on

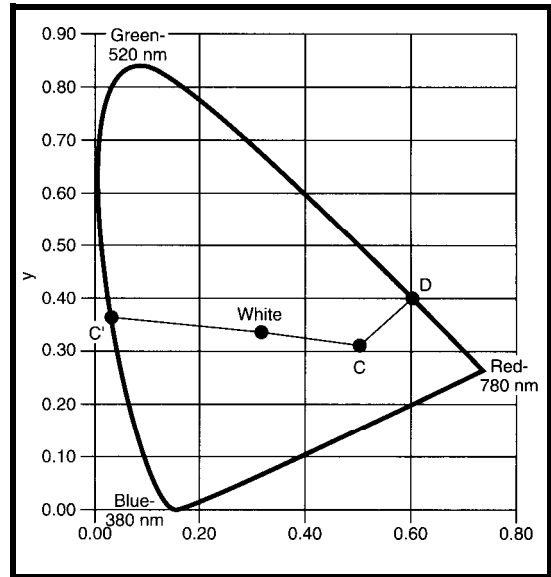


Figure 6—The interior of the CIE chromaticity diagram represents the total color set that humans perceive.

displays where there is abrupt intensity or intensity-slope changes.

- be aware of color-recognition deficiencies

The phrase "color blindness" is a misnomer, better described as a color-recognition deficiency. Many people (-10% Caucasian men, -4% non-Caucasian men, and -0.5% women) have some form of color-recognition deficiency.

The most common deficiency is the inability to discern red versus green. It is a good rule of thumb to redundantly display important information (e.g., colored and outlined).

Because many have problems recognizing colors quickly, don't have color recognition as the single point of failure in crucial operations of interac-

tive systems. Also, because hardcopies get photocopied, color information is eliminated. Duplicate information through shape, fill pattern, outline pattern, outline thickness, character strings, fonts (including bold and italics), and symbols.

- outline boundaries

Two colored areas adjacent to each other is common in scientific visualization (e.g., when two countries abut on a map or areas of stress concentration are depicted contiguously).

Usually, the shape of the border is important since cones, which detect color, are not good at detecting boundaries. Rods detect boundaries well, especially if the line between adjacent colors is black or white.

- avoid saturated blues for fast-moving items or fine detail

Only about 10% of your cones are the S-type, which means that your

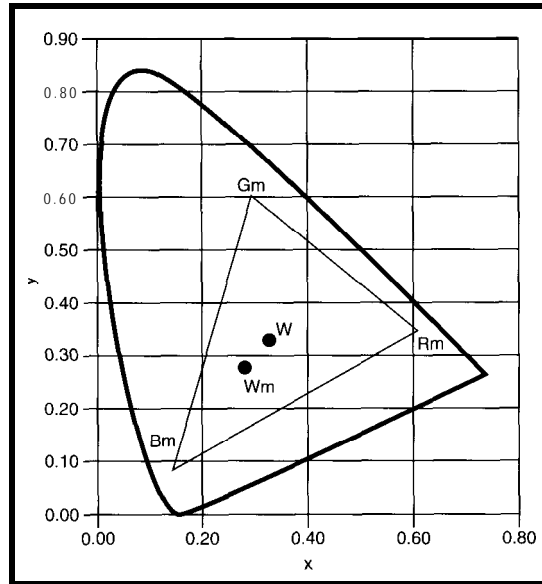


Figure 7—In the color gamut for a Silicon Graphics monitor, the white of the monitor ( $W_m$ ) is more blue than the overall white point ( $W$ ).

sensitivity to blues is reduced. In other words, the visual system processes blues less effectively than other colors. From the luminance equation, we know that only about 11% of overall luminance come from blue.

Less-saturated blues work much better because other wavelengths are mixed in, thus stimulating the more effective M- and L-type cones.

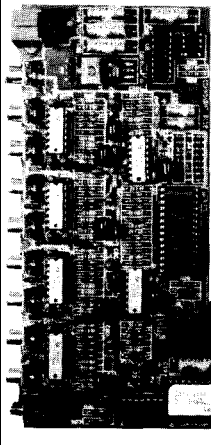
- avoid mixing saturated reds and blues together

Do not place saturated reds and blues next to each other. In the eye, low color frequencies (reds) focus in front of high frequencies (blues). The difference in focal locations, known as *chromostereopsis*, appears to the eye as a change in depth. The red portion appears closer to the viewer than does the blue portion.

In advertising, this effect is often used to draw attention. In scientific visualization, however, it can be disconcerting. Although the viewer knows that the entire display is at a single depth, chromostereopsis fights this knowledge.

If reds and blues must be adjacent, desaturate them so that other color

# CIRCUIT CELLAR KITS



## HAL-4

EEG Biofeedback Brainwave Analyzer

The HAL-4 kit is a complete battery-operated 4-channel electroencephalograph (EEG) which measures a mere 6"x7". HAL is sensitive enough to even distinguish different conscious states—between concentrated mental activity and pleasant daydreaming. HAL gathers all relevant 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 frequency, amplitude, and phase components, the results are graphically displayed in real time for each side of the brain.

HAL-4 kit.....\$179.00 plus shipping

## Sonar Ranging Experimenter's Kit

Targeting ♦ Ranging ♦ Machine Vision

The Circuit Cellar TI01 Ultrasonic Sonar Ranger is based on the sonar ranging circuitry from the Polaroid SX-70 camera system. The TI01 and the original SX-70 have similar performance but the TI01 Sonar Ranger requires far less support circuitry and interface hardware.

The TI01 ranging kit consists of a Polaroid 50-kHz, 300-V electrostatic transducer and ultrasonic ranging electronics board made by Texas Instruments. Sonar Ranger measures ranges of 1.2 inches to 35 feet, has a TTL output when operated on 5V, and easily connects to a parallel printer port.

TI01 Sonar Ranger kit.....\$79.00 plus shipping

## CHECK OUT THE NEW CIRCUIT CELLAR HOME CONTROL SYSTEM

- ♦ Expandable Network
- ♦ Digital and Analog I/O
- ♦ X-I/O Interface
- ♦ Voice
- ♦ IR Interface
- ♦ Remote Displays
- ♦ Telephone Interface

Call and ask about the HCS II

To order the products shown or to receive a catalog, call: (203) 875-2751 or fax: (203) 872-2204  
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 that HAL be battery operated only.

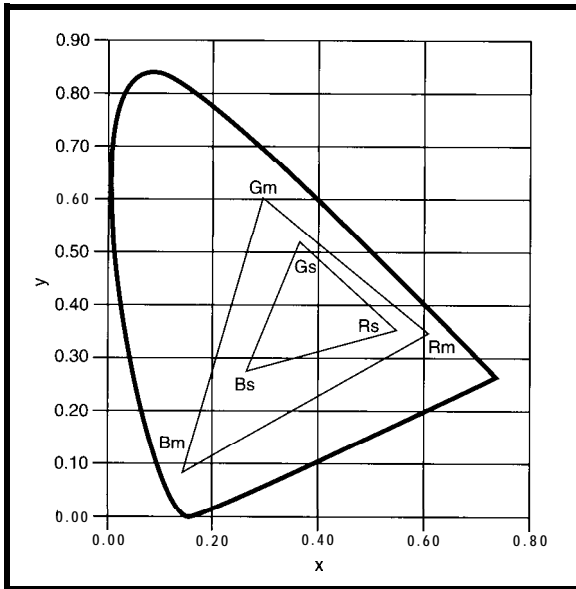


Figure 8—The slides are projected before taking the CIE measurements so they are viewed in the environment they will most often be seen in. Thus, this color gamut is a result of both the film's color response and the projector light bulb's color bias.

frequencies mix in and the frequency difference is not as dramatic.

- do not display high spatial frequencies in color

High spatial frequencies (e.g., closely spaced thin parallel lines) are a lot easier to recognize in black and white than in color. Rods have higher spatial discrimination than cones.

## COLOR GAMUTS

A color is a color, right? Wrong.

The color a device is asked to display and the color that comes out are usually different. The range of colors that a graphics device is capable of displaying is referred to as a *color gamut*.

Before discussing color gamuts, we need to introduce the *CIE chromaticity diagram*. Figure 6 shows this diagram which was created in 1931 by the Commission Internationale d'Éclairage (CIE) and was the result of experiments in human color vision.

Although it is interesting to delve into how the graph was created or why it is the shape it is, the important thing to know is that the interior of the graph represents the total color set humans perceive. The horseshoe represents the electromagnetic spectrum from 380 nm (blue) in the lower-left corner to 520 nm (green) at the top and

780 nm (red) in the lower-right corner. The equal-energy white point lies at the coordinate (0.33, 0.33). The distance from the white point measures saturation, and the angle around the white point is considered the hue.

Colors on the outline of the horseshoe are *pure colors* (i.e., composed of a single wavelength). The exception to this rule is the diagonal straight line from blue to red, known as the *magenta line*. The line is pictured because display monitors mix blue and red to get magenta, so it is handy to have some way of showing this, even though the electromag-

netic spectrum does not.

For the color C in the interior, the shortest distance to the outline of the horseshoe produces the color's dominant wavelength, *D*. A line through the white point (*W*) produces the color's complementary color *C'*.

The CIE diagram provides a standard measurement methodology for comparing color on different devices. CIE color is measured with an instrument called a *spectracolorimeter*. Different devices (CRTs, paper, film) have different color gamuts.

For example, Figure 7 shows the color gamut for a particular Silicon Graphics monitor (measured by a Photo Research PR-650 spectracolorimeter [4]). Note that the white of the monitor (*Wm*) is more toward the blue part of the gamut than the overall white point (*W*). Monitor vendors have discovered what laundry detergent makers have known for some time: humans perceive a slightly bluish "white" than pure white.

Obviously, there are many colors the eye de-

fects that the monitor cannot display, primarily in the green areas. This problem is most critical in generating realistic images, but in scientific visualization, color usually represents something else.

It does, however, become a major consideration when moving scientific visualization images from the monitor to another display device (e.g., a screen dump of a monitor display which sends the RGB values directly to a slide film recorder). Figure 8 shows the color gamut of projected slides.

Not surprisingly, Figure 8 shows a good range of color reproducibility, but with a distinct bias toward the yellow portion of the spectrum. This bias is a result of the yellowish color coming from the bulb in the projector. This drift mostly hurts the reproducibility of blues in the original image.

The biggest color gamut problems come in subtractive color printing.

Figure 9 shows the color gamut for a Canon CLC-500 color printer. As you can see, the color gamut for the CLC-500 is much smaller than that of the SGI monitor.

Besides having a smaller gamut, this example shows the more insidious problem of color rotation. Lines drawn from the white point through each of the two green points show a considerable rotation around the white point. As a result, not only are the saturated

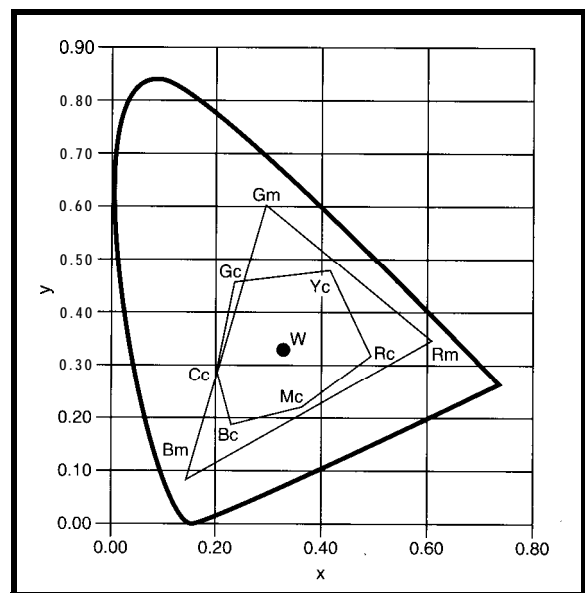


Figure 9—The biggest problems with color gamuts occur in subtractive color printing. As you can see, there can be quite a difference between the color gamut of a monitor and a color printer.



monitor colors not able to be printed, but attempting to print them results in a considerable hue change.

Much work has been done on the use of color theory to match the colors of different display devices [5],[6], and [7].

## LIMITATIONS OF NTSC VIDEO

Frequently, scientific visualizations are recorded to videotape. It would be nice if this was an automatic process (i.e., what you see on the monitor is what you get on the video).

Sadly, this is not even close to true. NTSC [North American] video has a set of idiosyncrasies that must be understood to make the production of scientific videos hassle free. The following are some general guidelines when going to NTSC video from a display:

- Don't wait until the last minute to consider video issues. The best time to begin thinking about them is when you start to design your display. The colors, layout, and amount of fine detail all come into play when you eventually go to video.
- NTSC does not handle saturated colors well. Use a saturation of 0.80 or less.
- Use two or more pixel thicknesses (no single-pixel thicknesses).
- The resolution of NTSC video is approximately 640 x 480. Keep this in mind when deciding how large to make text, graphs, and so on.
- NTSC is encoded with 267 intensity cycles occurring per scanline. Any more than this does not show up.
- NTSC is encoded so that 96 cycles of orange-blue and 35 cycles of purple-green occur per scanline. (Orange-blue is emphasized more than purple-green because this range includes flesh tones.) Much less detail can be encoded in color than can be encoded in intensity because of the sensitivity of the human eye. Fine detail should be displayed in black and white, not color.
- Most workstation video must be passed through a standards converter before it is NTSC compatible. It is important to understand the characteristics of your standards converter

and how it downsizes the image. For more information, see Blinn [8].

## PREVENT COLOR POLLUTION

Finally, avoid overuse. Just because you have  $2^{24}$  colors, doesn't mean you have to use them all. □

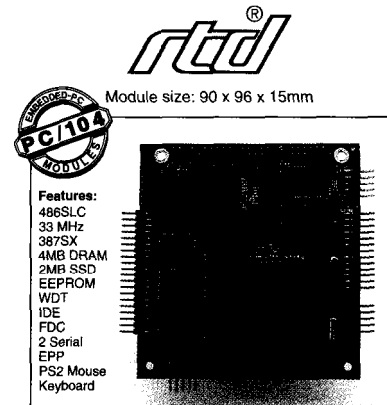
**Dr. Mike Bailey is a senior staff scientist at the San Diego Supercomputer Center and an associate professor at the University of California at San Diego. Mike specializes in scientific visualization and computer-aided engineering. He may be reached at [mjb@sdsd.edu](mailto:mjb@sdsd.edu).**

## REFERENCES

- [1] Jim Foley, Andy van Dam, Steve Feiner, and John Hughes, **Computer Graphics Principles and Practices**, Reading, MA, Addison-Wesley, 1990.
- [2] Roy Hall, **Illumination and Color in Computer Generated Imagery**, Springer-Verlag, 1989.
- [3] Haim Levkowitz and Gabor Herman, "Color Scales for Image Data," **IEEE Computer Graphics and Applications**, 72-80, January 1992.
- [4] Photo Research, PR-650 **Spectra-Colorimeter**, Specification Sheet.
- [5] Maureen Stone, William Cowan, and John Beatty, **Color Gamut Mapping and the Printing of Digital Color Images**, Xerox Report EDL-88-1, 1988.
- [6] Maureen Stone, **Color Printing for Computer Graphics**, Xerox Report EDL-88-5, 1988.
- [7] Maureen Stone, William Cowan, and John Beatty, "Color Gamut Mapping and the Printing of Digital Color Images," **ACM Transactions on Graphics**, Vol. 7, No. 4, 249-292.
- [8] Jim Blinn, "NTSC: Nice Technology Super Color," **Computer Graphics and Applications**, 17-23, March 1993.

401 Very Useful  
402 Moderately Useful  
403 Not Useful

## REDUCE THE STACK! Use fully integrated PC/104 CPU and DAS modules from



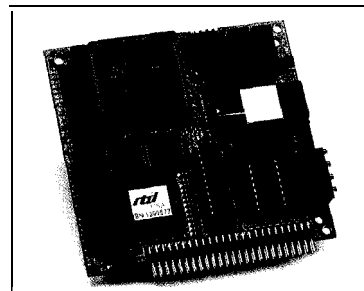
**PC/104 Compliant PC-AT SBC  
CM1486SLC-1: \$597 /100 pieces  
2MB DRAM & SSD software included.**

### 6 PC/XT/AT Single Board cpuModules™:

- 486DX, 486SLC, 386SX, F8680 & V30 CPUs
- 80387SX math coprocessor socket on-board
- 2MB, 4MB or 8MB DRAM installed
- Two 32-pin SSD sockets & support software
- IDE, floppy & CGA controllers
- RS-232/422/485 serial ports
- Bidirectional parallel, keyboard & speaker ports
- Keypad scanning & virtual device support
- Power management & single +5V supply

### 7 utilityModules™:

- SVGA, Ethernet & intelligent GPS modules
- PCMCIA carriers for Types I, II & III cards



**PC/104 Compliant 200 kHz Analog I/O Module  
DM5408-2: \$498 /100 pieces**

### 17 DAS dataModules®:

- 12 & 14-bit A/D conversion up to 200 kHz
- Random scan, burst & multiburst
- Pre, post & about triggers
- 1 K channel-gain scan memory with skip bit
- 1024 sample A/D buffer with data marker
- 12-bit analog outputs
- Bit programmable digital I/O with Advanced Digital Interrupt modes
- Incremental encoder interfaces
- 4-20 mA current loop source
- opto-22 compatibility
- Low power & single +5V power supply

For technical specifications and data sheets on PC/104, ISA bus and Eurocard products, call  
RTD USA Technical FaxBack®: 1 (814) 235-1260  
RTD USA BBS: 1 (814) 234-9427

**rttd Real Time Devices USA**

200 Innovation Boulevard • P.O. Box 906  
State College, PA 16804-0906 USA  
Tel: 1 (814) 234-8087 • Fax: 1 (814) 234-5218

**RTD Europa RTD Scandinavia**  
Fax: (36) 1 212-0260 Fax: (358) 0 346-4539

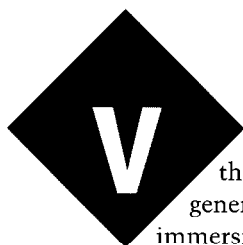
RTD is a founder of the PC/104 Consortium and the world's leading supplier of PC/104 CPU and DAS module:

# Virtual Reality Position Tracking

Until recently, magnetic trackers existed solely in the workstation domain. But, with the popular advent of VR, trackers have joined the PC world. Here, Herschell and Mark introduce us to tracker technology.

## FEATURE ARTICLE

Herschell Murry & Mark Schneider



Virtual reality is the science of generating an immersive, synthetic environment about a person. It simulates a real environment where we can move in any direction and automatically get different views of our world. We could generate a similar environment in a virtual world if space, time, and money were no object *and* we had the USS Enterprise and its holodeck.

Failing that, we don a helmet with displays right before our eyes and move around in a cyberworld. Displayed scenery moves as we move. A computer can do this kind of simulation quite nicely if we can tell it our movements so it knows where to scan the scene. And, this is where motion trackers enter into the virtual reality cybersystem design.

The head tracker determines the position and orientation (P&O) of the head, which the computer then uses to generate the appropriate scene for display. Trackers also are used on instrumented gloves (e.g., the DataGlove) for enabling physical interaction in a virtual world.

Since virtual-reality tracking systems are likely to become affordable in the near future, now is a good time to get a jump on understanding them. A thorough explanation of trackers requires a sizable text. We'll explain enough about tracker technology, its use, and interface requirements to give you a working knowledge. Hopefully, it'll be enough to keep you out of trouble.

## TRACKING FUNDAMENTALS

The best current tracking systems are based on the AC electromagnetic coupling between two sets of three-axis dipole antennas (i.e., three mutually perpendicular coils of wire about a common center). A typical low-cost tracker has a transmitter (the reference set of antennas), a receiver (the tracked set of antennas), and a PC-compatible electronics board (see Figure 1). One of the lowest cost PC-compatible trackers available today, the Polhemus Insidettrak, is shown in Photo 1.

The tracker's PC-compatible electronics can be separated into analog and digital portions. Three receiver antennas are connected to three analog input channels as pictured in Figure 2. These channels are composed of three low-noise, high-gain amplifiers whose outputs are multiplexed into a single fast, high-resolution A/D converter.

The three transmitter antennas are driven by three high-power analog outputs. These outputs are digitally generated via low-resolution DA converters.

No doubt, you realize that other configurations are possible: separate A/D converters representing each channel, a multiplexed D/A converter, and so on. The tradeoffs between the different configurations affect overall price and performance of the tracker.

Analog inputs and outputs are under control of a microprocessor. In recent designs, a DSP has become common since it better performs digital-filtering functions. The DSP is responsible for generating the transmitter signals, collecting the multiplexed input data, processing the raw input data into P&O data, and interfacing to the I/O bus.

These processes occur during a typical tracker measurement cycle. During the data-acquisition time, the transmitter antennas are driven and the induced receiver voltages are measured. This process takes most of the measurement cycle.

Calculation of P&O from the collected data occurs next and uses a smaller part of the cycle. The remaining time is dedicated to performing I/O over the PC bus (or buses like RS-232/422 or IEEE-488 for external tracker

designs). Once a cycle completes, a new measurement cycle begins.

The equations defining the antenna couplings are well known and are solved to give the P&O of the receiver with respect to the transmitter. The main fault with this type of tracker is exhibited when it is used near conductive materials.

Because trackers generate AC magnetic fields, nearby conductive materials receive and retransmit these fields as well. These additional fields are not accounted for and do not fit the assumptions of the tracking algorithm.

The advantage, however, is that magnetic trackers do not require an unobstructed view between receiver and transmitter like an infrared or ultrasonic system. And, when designed correctly, they are reasonably immune to interference from other electronic gear, but more on that later.

## TRACKER SPECIFICATIONS

One of the problems in tracking real-time movement is the interval of time called *latency*. Latency is measured from the midpoint of the time period during which data collection occurs (i.e., the midpoint of the motion during this interval) to the start of the output of a P&O solution.

You might wonder about the data taken before and after the midpoint. The crafty tracker designer, using linear prediction, adjusts the collected data from the ends of the data acquisition toward the middle. However, this only covers *tracker* latency. When a tracker is integrated into a VR system, the overall [or system] latency needs to be reckoned with (more on this later).

Other tracker specifications that need definition are:

- update rate
- accuracy
- resolution
- repeatability

Update rate is the interval of time between P&O solutions. If the mea-

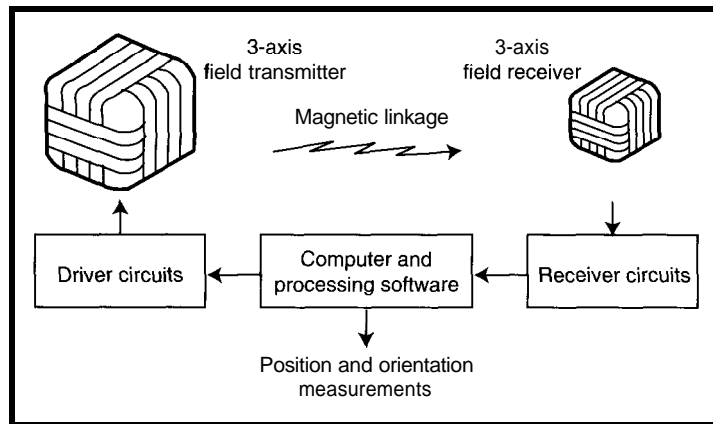


Figure 1—A magnetic motion tracker system typically includes a transmitter, receiver, and a PC-compatible electronics board.

surement cycle is 33.3 ms long and the tracker computes a new P&O solution each measurement cycle, the update rate is 30 Hz.

Accuracy indicates how well the computed P&O matches the actual position and orientation. Accuracy is affected by many of the overall system-engineering design considerations including the size of the receiver and transmitter antennas, mechanical assembly of the antennas, drive signal, and signal-to-noise ratio (SNR), among others. SNR itself is affected by a combination of factors including tracker design, its electromagnetic environment, and the range between transmitter and receiver.

A typical contributor to environmental noise is the display. Although many are electrically quiet LCDs, TVs and display monitors are CRTs which generate magnetic fields during refresh. The relatively low-frequency

vertical refresh can easily overload the tracker's sensitive front end. The higher-frequency horizontal refresh has become less of a problem recently as these rates have increased beyond the tracker's passband.

As noted earlier, the other environmental threat stems from conductive materials. Every effort should be made in system design to use non-conductive materials like

plastic, wood, and glass. If that is impossible, poor conductors such as cast iron and some types of stainless steels can often be tolerated with small effects. The worst materials to use are good conductors like copper, aluminum, and brass.

The signal strength that combats such noise drops off as the cube of the transmitter-receiver separation since the magnetic field must flood the 3D space. In other words, if the range of separation is doubled, the signal drops to one-eighth of its earlier value (i.e.,  $1/2^3 = 1/8$ ). Obviously, SNR is always an issue in trackers. You'll note the same problem occurs with resolution.

Resolution is the smallest change a tracker can detect in position and orientation. As with accuracy, both system and environmental factors affect resolution. Internally, the resolution of the A/D converter and P&O computation, the circuitry SNR, and a

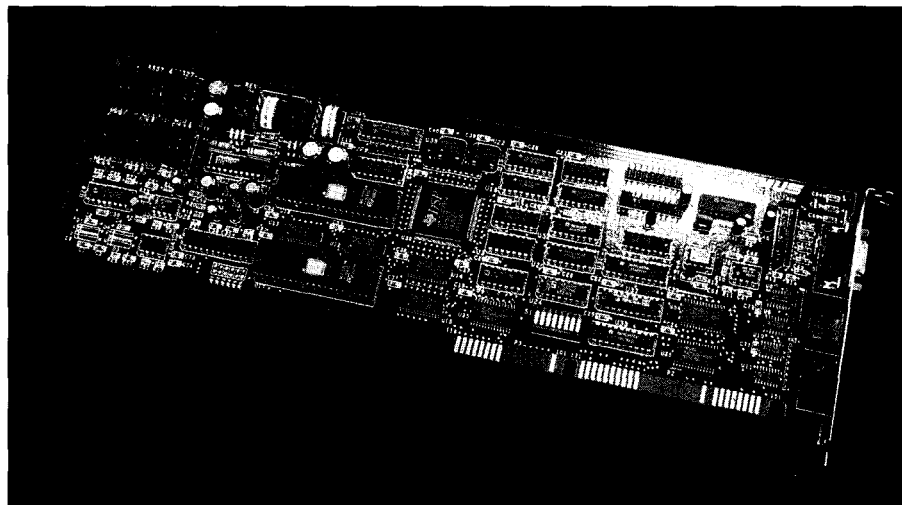


Photo 1—The Polhemus Insidetrak board is one of the lowest cost PC-compatible trackers available today.

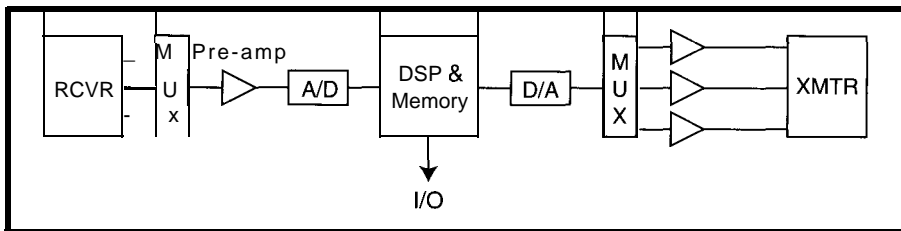


Figure 2—The motion tracker links three antenna coils carrying signals coupled from three transmit coils that are being driven simultaneously. This communication with itself is an ideal DSP application.

magnetically noisy environment degrade the final resolution.

The final and ultimate limiter of resolution is the separation between the receiver and transmitter. A VR tracker might have a working range of 1-5'. Because the strength of the magnetic field varies at  $1/\text{range}^3$ , the signal

not normally specified, it obviously is related to both accuracy and resolution. If the accuracy varies due to uncorrected drifting of the circuitry, repeatability suffers. Likewise, if the resolution is so gross that you can easily straddle the demarcation line between two adjacent ADC readings, repeatability again suffers.

### TRACKER INTERFACING

Now that we know how a tracker works and what its key definitions are, it's time to get to what an application buff might call the "good stuff." A low-cost tracker board is usually based on the 16-bit ISA bus standard for maximum utility (it should work on an EISA bus also). Input commands, tracker status, and data are 16 bits long. A jumper block or switches for setting a bus address are available. Assuming you've read the tracker manual, you have set the correct address to a value different from anything else on the bus.

Being able to access tracker data directly on an internal bus is a plus. Considering that quite a few bytes of data are transferred per P&O answer at

least 30 times a second, a great deal of real-time data must be handled. Many trackers on the market today use a serial bus such as RS-232, -422, or -485. However, these units not only must run at a high baud rate, their system software must navigate considerable overhead. As you know, overhead delays getting the data.

Typically, one would like that data directly in the binary representation of the computer. This facility usually is not the case with external interfaces, adding a formatting or conversion task to the software, and thereby adding more delay.

So, with a PC, having the tracking data available right on the ISA bus is a marvelous feature. It makes the tracker appear faster than it really is compared to external trackers.

### SYSTEM ALIGNMENT

Probably one of the hairiest parts of setting up a motion tracker is getting the transmitter and receiver coordinate systems figured out and coordinated with the video. Of course, everything in a magnetic motion tracker is referenced to its own signal source—the transmitter.

This reference frame typically is called the *space frame*. The coordinate framework for the receiver is typically called the *body frame* since it usually is located on your body.

Two processes need to be understood to get these frameworks to cooperate with your video world: transmitter alignment and receiver boresight.

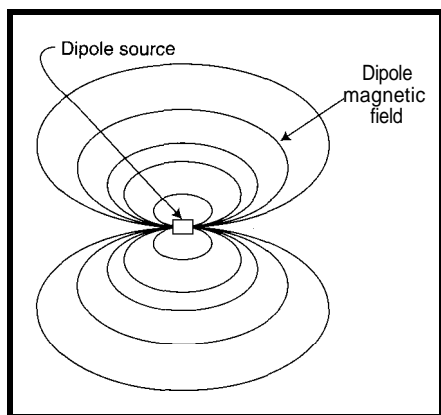


Figure 4—The dipole field is symmetrical around any axis drawn through its center.

drops rapidly with separation.

The maximum signal is designed to occupy the full range of the A/D converter. As the signal decreases, the number of bits required to represent it also decreases so that only the least-significant bits are left taking readings.

Positional resolution is therefore specified as a function of range. It can be expected to be on the order of 0.0003–0.001 inch per inch. Orientation resolution, being based on ratios of measurements, is independent of range (except that it's still dependent on data which is represented by a smaller and smaller range of bits) and is usually something like 0.1-0.5".

Repeatability refers to the tracker's proficiency at providing the same P&O output when the receiver is placed in exactly the same position and orientation with respect to the transmitter over and over again. While

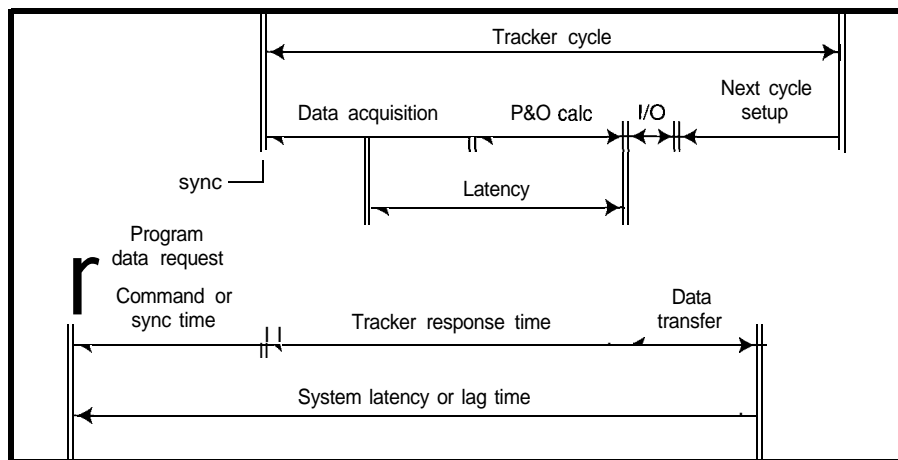


Figure 4—The tracker transmits and acquires data and then calculates position and orientation. Tracker latency extends from the motion acquisition midpoint to the start of the output. Data out continues at the pace of the host computer as the tracker prepares to start the next cycle.

Alignment of the transmitter can be done mechanically, but it is much harder than the mathematical method provided in the tracker. Aligning the transmitter is similar to setting up a camera to pan in the horizontal plane. If you turn left and see nothing but sky and turn right and see nothing but ground, you know the camera is tilted.

Similar processes are available in the tracker to carry out transmitter alignment through collecting data and making mathematical adjustments. Going through the whole process is beyond this article's scope, but your tracker user's manual is a lot of help.

Once alignment is done, the receiver(s) can be similarly adjusted, but it is much easier. Remember the receiver navigates relative to the transmitter's space-frame coordinates.

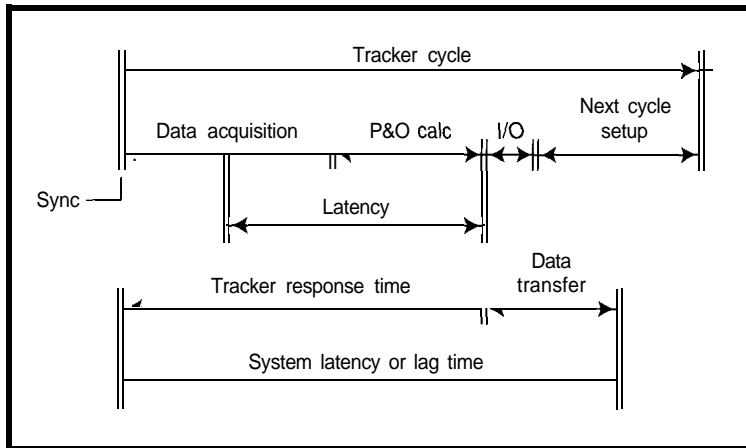


Figure 5—If you compare this figure to Figure 4, it is obvious that the continuous mode results in much greater efficiency.

But, the body frame can be referenced simply by orienting the receiver in a certain direction and executing a boresight command. This position becomes the (0, 0, 0) reference in the body frame. However, if the transmitter has not first been aligned, don't expect a boresight to align it. The boresight only helps with the receiver.

With a magnetic motion tracker, hemisphere is also important. Refer to

the dipole field shown in Figure 3. It is easy to see that it is symmetrical about any axis drawn through its center. This symmetry means there are identical answers 180° apart. Hence, the user must define which hemisphere obtains the right translational output.

Determining this is usually quite easy because you are walking either on the floor or on the ceiling, sitting inside the cockpit or outside the canopy.

And, the tracker has means for defining the hemisphere in its manual.

However, be cautious with distortion and hemispheres. Sometimes distortion can create an answer that throws the reading into the opposite hemisphere. Needless to say, it's easy to detect, can be very disconcerting, and should be accounted for in system design.

# Mid-Tech USA

Very high performance C-programmable controllers and development software

The ec.52 8031-compatible 8 MIPS controller  
Call for fill-line hardware/software product catalog

**Mid-Tech Computing Devices USA**  
P.O. Box 218 • Stafford, CT 06075 • voice/fax: (203) 684-2442

## CADPAK & PROPAK for Windows

### PROFESSIONAL SCHEMATIC CAPTURE AND PCB SOFTWARE FOR WINDOWS NEW!

# LOW PRICES!

Professional schematic & PCB design Software for Microsoft Windows 3.1

- Automatic wire routing & dot placement.
- Exports diagrams to other applications.
- Full control of drawing appearance.

PROVIDES all the Convenience of Windows & the POWER of CAD Software

- Advanced routing
- Libraries, including SMT Gerber, Excellon & DXF
- Includes Gerber Viewing
- Fast & Easy to use.

**R4 SYSTEMS INC.**  
1100 GORHAM ST. SUITE 111 B-332  
NEWMARKET, ONTARIO  
CANADA L3Y 7V1  
905 898-0665 FAX 905 898-0683

**FREE DEMO  
WRITE OR CALL  
TODAY**

**BBS 905 898-0508 (9600,8,N1)**



## SYSTEM LATENCY

In Figure 4, you see a complete application cycle, including system lag from cycle initiation to receipt of data.

Of course, you can't afford to wait for data after kicking off a tracker cycle, but it is important to determine what the latency time period is. Once you know the interval, you can go about other tasks and come back for the P&O answer when it is ready. By doing this, little time is lost.

It is even more efficient to set the tracker into continuous tracking operation. By syncing up with it, you only need to return when an interrupt indicates output is available (Figure 5). Hence, each P&O answer precedes display computation. The display frame updates while the tracker comes around with the next cycle.

You can also reduce system latency by using only the parameters you need and obtaining them in the most efficient format for you. You might want binary data rather than ASCII or your application may need only orientation angles and not posi-

tion. Besides saving memory space, such cuts save code run time, I/O transfer time, and formatting.

## TRACKER EFFICIENCY PERKS

Even with an efficient internal bus like the ISA, the user can realize even more efficiency by

- requesting only the parameters needed
- choosing a compact, directly usable data format
- integrating the system cycle with the tracker cycle through proper sync generation
- metering out the system cycle in the most logical and computationally sensible sequence □

*In his 25-year career, Dr. Herschell Murry has served as chief scientist, program manager, and project engineer for companies such as Andrew Government Systems, Rockwell International, and Martin Marietta Aerospace. As vice president of engineering at Polhemus, Herschell*

*oversees product design and development.*

*In his 15 years at Polhemus, Mark Schneider has led the design and analysis for commercial products. This year he received the Academy Award for Technological Achievement from the Academy of Motion Picture Arts and Sciences, Mark serves as Polhemus's manager of newproduct design. Mark may be reached at (802) 6553159, ext. 290.*

## SOURCE

Insidetrak  
Tom Jones  
Polhemus, Inc.  
P.O. Box 560  
Colchester, VT 05446  
(806) 655-3159, ext. 234  
Fax: (802) 655-1439

## IRS

404 Very Useful  
405 Moderately Useful  
406 Not Useful

**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.



**CALL 1-800-635-3355**

**MICROMINT, INC.**

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



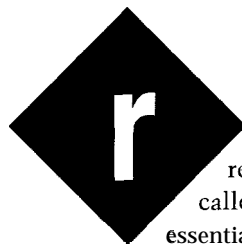
in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194 • Distributor Inquiries Welcome

# FEATURE ARTICLE

James Goel

## Digital Video Resizing and Compression

Powerful processors and imaging techniques have made computer video possible. Here, James introduces Genesis's resizing chips. They reduce, bypass, or zoom using FIR filters with up to 65 taps.



Real-time image resizing (often called scaling) is essential in modern video systems. Powerful workstations combine graphics and scaled digital video to create today's hottest computer animation. Everything from huge video walls in trendy night clubs to video-in-a-window teleconferencing on a desktop PC requires digital-video resizing. With the invention of modern graphical user interfaces, people expect to control how and where their video data is displayed—they're no longer content watching fixed-sized video.

Multiple resizable video windows are the next evolutionary progression for modern computers. In fact, the upcoming release of Windows 95 contains integrated video playback software for applications like video conferencing, advanced computer-aided learning, and CD-ROM games.

But, before a digital nirvana of cost-effective, high-quality integrated video arrives, a number of difficult problems must be solved. The first is a lack of storage and transmission bandwidth. Video data is too massive to be stored or transmitted without compression, though this situation has improved with video compression standards like

the Moving Pictures Experts Group (MPEG-1 and 2), H.261, and the Joint Photographic Experts Group (Motion JPEG). These standards are widely accepted and use various algorithms to achieve compression ratios of 100: 1 and beyond.

The second problem is a lack of quality scaling. Efficient video compression is just one link in a chain. It must be coupled with effective resizing to create an image worth viewing. Until recently, systems relied on the "cheap and nasty" *nearest neighbor* or the slightly better *linear interpolation* algorithms for video scaling.

Unfortunately, these algorithms produce video images suffering from aliasing (visually evident as absent image detail or extraneous artifacts). Until now, resulting picture quality has ranged from barely acceptable to poor. Fortunately, new cost-effective, high-quality scaling algorithms have been developed.

Distortion-free scaling is crucial in applications like video production, teleconferencing, navigation instrumentation, and medical imaging because these applications demand high-quality output. Modern systems chain devices together to manipulate digital video during production. Throughout this process, quality video scaling must be used to prevent contamination of final video output.

Understanding high-quality scaling requires a little background information. First, I'll cover antique resizing algorithms which set the stage for new scaling technology. However, old or new, the basic concept of all video scaling is determining how to generate and position output target pixels in relation to the input source pixels.

By the way, the techniques mentioned in the next section only

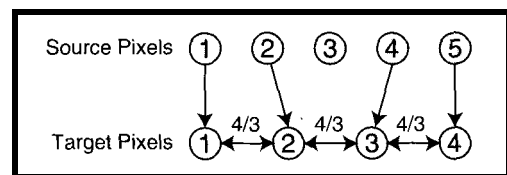


Figure 1 a--The *nearest-neighbor resizing algorithm* is fast and cheap but results in poor-quality output. In this example, the third pixel in the source is dropped from the final image.

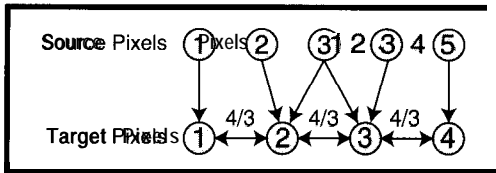


Figure 1 b—The linear interpolation algorithm simply averages the closest source pixels to create the required output pixels. At most, each output pixel is generated from a maximum of two input pixels.

involve resizing of horizontally spaced pixels. Vertical scaling is already covered since all algorithms can perform vertical resizing if entire video lines are substituted in place of single pixels.

### NEAREST NEIGHBOR

The nearest-neighbor algorithm could also be called “keeping up with the Joneses” because each target pixel finds its closest neighbor and copies it. This cheap and simple algorithm runs quickly, but produces a distorted output.

The technique begins with first determining the closest neighbor. The space between target pixels is calculated in relation to the space between source pixels. Let’s call this value *TarInc*. *TarInc* describes the output grid spacing on which new target pixels must be placed:

$$TarInc = \frac{\text{Number of Source Pixels} - 1}{\text{Number of Target Pixels} - 1}$$

The subtraction of 1 in both the numerator and denominator is required because the spaces between pixels are important, not the pixels themselves.

Figure 1a illustrates an example where five source pixels are reduced to four target pixels. In this figure, *TarInc* turns out to be  $\frac{4}{3}$ . All target pixels are separated by a space equal to *TarInc*. The exact position of each target pixel in relation to the first source pixel is now known.

The next step is to calculate the value of each target pixel. In this algorithm, source pixels are mapped directly to the target pixels that are spatially close. In our example, source pixel 1 maps directly on top of target pixel 1, and target pixel 2 moves  $\frac{1}{3}$  to

the right of source pixel 2 and  $\frac{2}{3}$  to the left of target pixel 3. Thus, source pixel 2 maps to target pixel 2 because it is nearest.

As you can see, after all pixels are mapped, source pixel 3 does not contribute to the target image. Remaining unmapped source pixels are automatically thrown out.

The problem with this algorithm is that source pixels are removed if they do not closely neighbor target pixels. Entire lines and columns drop from the original image. These visual distortions are obviously unacceptable for medical applications where doctors require the highest image fidelity possible for accurate diagnosis.

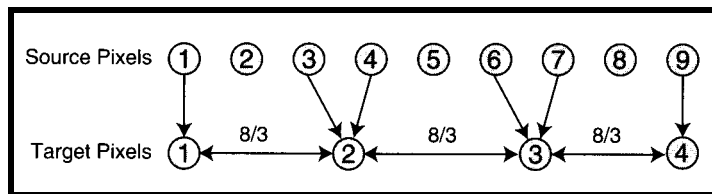


Figure 1 c—In this case, linear interpolation has degenerated into the nearest-neighbor algorithm. Source pixels 2, 5, and 8 make no contribution to the output target pixels.

Schematics and line drawings are also susceptible to the harsh aliasing effects of the nearest-neighbor algorithm.

### LINEAR INTERPOLATION

Linear interpolation improves on the nearest-neighbor algorithm by calculating the value of each target pixel as the weighted average of the two closest source pixels. This technique prevents pixel and line dropping for certain resize factors above 50%, but aliasing still remains a problem.

In Figure 1b, although the spatial positioning of the target pixels is the same as the nearest-neighbor method of Figure 1a, the value for each pixel is calculated differently. The first source pixel maps directly to the first target pixel, but the second target pixel is calculated as a weighted average between source pixels 2 and 3.

Target pixel 2 is  $\frac{1}{3}$  to the right of source pixel 2 and  $\frac{2}{3}$  to the left of source pixel 3. Thus, the following equation calculates the value of target pixel 2:

$$\text{Target Pixel [2]} = \text{Source Pixel [2]} \times \frac{2}{3} + \text{Source Pixel [3]} \times \frac{1}{3}$$

It is important to note that since target pixel 2 is closer to source pixel 2, it is made of  $\frac{2}{3}$  more of source pixel 2 than source pixel 3. All other target pixels are calculated by determining their position in relation to two surrounding source pixels.

This technique produces poor results for high-quality images with sharply defined lines and barely passable results for nonquality-sensitive images. Figure 1c illustrates this problem. Source pixels 2, 5, and 8 do not contribute anything to the target pixels causing the same line and column dropping discussed earlier.

### “TEXTBOOK CORRECT” VIDEO RESIZING

Correct video resizing must be performed for all resize factors without informa-

tion loss through line or pixel dropping. To achieve this, each source pixel must contribute appropriately to the output. Scaling video correctly results in the highest quality images, though it comes at increased complexity and cost.

Figure 1d illustrates an example of textbook-correct resizing using multirate digital signal processing. The multirate approach involves two steps.

If an image is being resized from *M* to *L* pixels, the first step generates a set of upsampled pixels by inserting *L* - 2 intermediate samples between each pixel. In Figure 1d, five source pixels

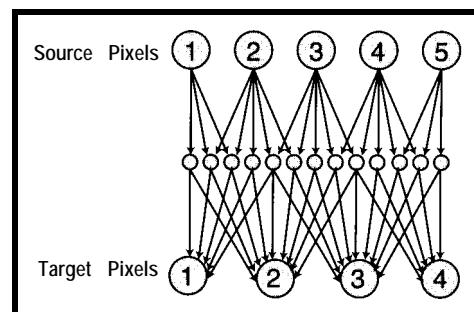


Figure 1d—The “textbook correct” algorithm ensures each source pixel makes a contribution to the output, regardless of the resize factor.

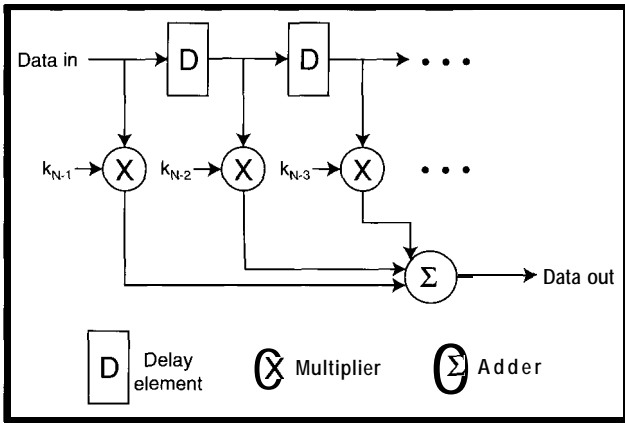


Figure 2-h a classic finite impulse response (FIR) filter implementation, delay elements  $D$  properly align input data with filter coefficients  $k_N-x$ , generating the proper data out sum.

must be reduced to four target pixels. Therefore, three intermediate samples are inserted between each source pixel. These upsampled pixels are generated using an interpolator.

The second step generates target pixels from the upsampled pixels using a decimator. The classic pipelined structure of a finite impulse response (FIR) filter decimator is shown in Figure 2. Basically, the decimator

multiplies upsampled pixels by a set number of coefficients called *filter taps* and sums the products to produce a single target pixel. The coefficients are calculated based on digital-signal-processing theory to produce target pixels with a minimum of aliasing. This system of upsampling prior to creating target pixels minimizes

aliasing for any resize factor.

However, implementing this algorithm presents two difficulties. The first is caused by the directly proportional relationship of the resize factor to the number of upsampled pixels.

For example, if a user wanted to resize 1000 pixels down to 999, then you'd start by generating the upsampled pixels:

$$\begin{aligned} \text{upsampled pixels} &= 1000 + [(1000 - 1) \times \\ &\quad (999 - 1)] \\ &= 998,002 \end{aligned}$$

Since these upsampled pixels must be generated in real time, either fast clock cycles or large amounts of VLSI real estate are needed to pipeline the upsampling process.

The second difficulty lies in the classic FIR filter structure shown in Figure 2. New filter coefficients are required for each resize factor to ensure upsampled pixels are decimated properly to produce unaliased target pixels. Calculating FIR coefficients for every resize factor is difficult and expensive to implement in silicon. These two problems make implementation of the multirate approach impractical on a silicon microchip.

Fortunately, new algorithms have made it possible to implement correct video scaling on a single VLSI chip.

## RESIZING TECHNOLOGY

What's needed for high-quality resizing is separable FIR filters which

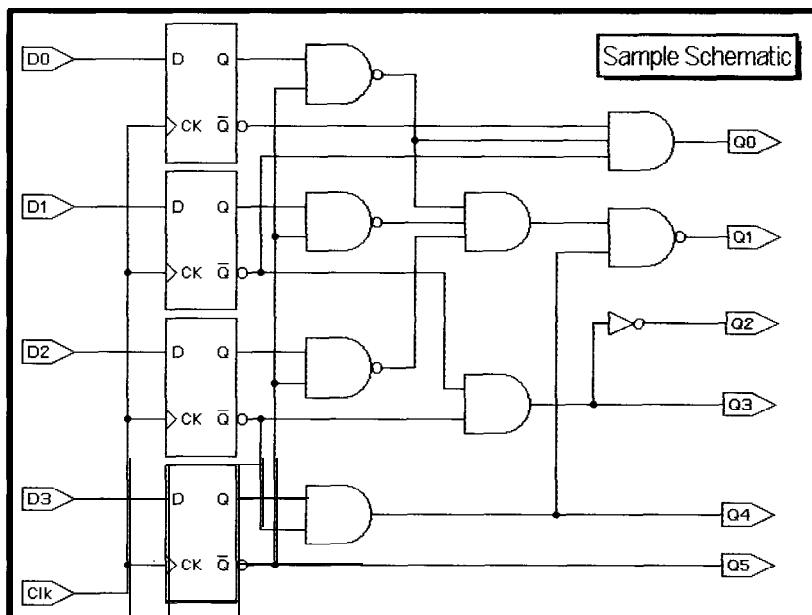
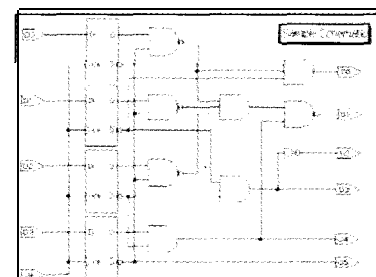
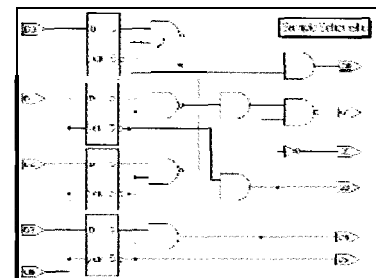
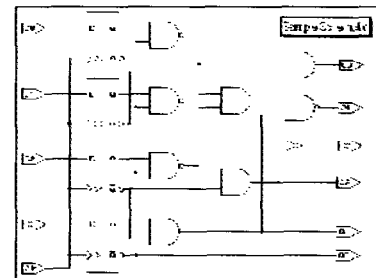


Figure 3-Source image (left) is unchanged. The right row shows three different methods of scaling the source image to 45% of original size. Pixel dropping (top) and bilinear interpolation (middle) lose precious image data. The bottom image, reduced using the Acuity Resizing algorithm, maintains image integrity.



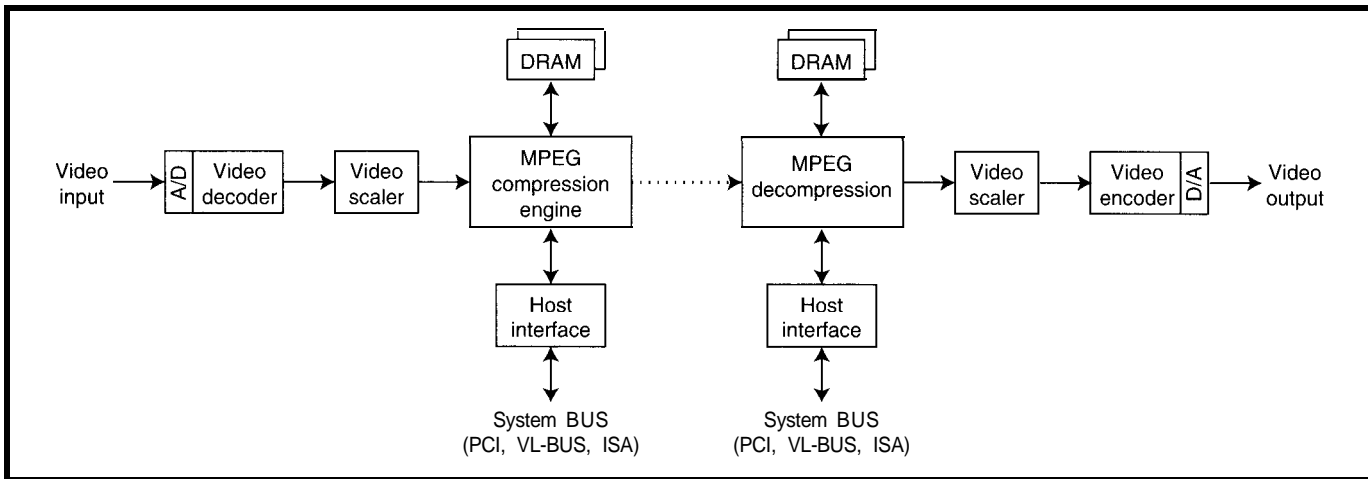


Figure 4—Video scaling before and after compression is an effective way of improving transmission throughput.

provide arbitrary and independent scaling in both vertical and horizontal dimensions. Image fidelity is retained by independently selecting the optimum vertical and horizontal filters so that image frequency content is removed and aliasing is minimized.

The Acuity Resizing family, a series of video/image resizing engines from Genesis Microchip, implements this technology. Depending on the selected resize factor, the engines can operate in reduction, bypass, or zoom modes with FIR filters ranging in complexity to 65 taps. The number of filter taps is selected automatically based on the resize factor.

As well, all required vertical and horizontal FIR filter memory resides on chip. The ICs can be cascaded to resize extremely large images or to implement high-speed systems.

The family currently includes four devices: the gm865x1, gm833x1, gm833x2, and gm833x3. All chips consist of vertical and horizontal filters, an output FIFO, a parallel host interface, and associated control. The output FIFO buffers the different input and output data rates while a parallel host interface enables the user to load source and target image parameters. Control consists of pixel input handshaking, field reset, and output FIFO empty and half-full flags.

Figure 3 compares the quality of the different resizing algorithms. As you can see, pixel dropping and bilinear algorithms (bilinear is linear interpolation in both vertical and horizontal directions) distort the image with

horizontal and vertical aliasing. The Genesis scaling algorithm maintains the image quality at all resize factors. Distortions caused by lower-cost algorithms make them unacceptable for even moderate-quality systems.

### RESIZING TO ENHANCE VIDEO COMPRESSION

With contemporary techniques, it is now possible to display multiple,

crystal-clear, overlapping video windows on your computer desktop. But, if you want to send these moving pictures to a friend or keep more than a few on your hard drive, image size is still a problem.

One major stumbling block to wider integration of digital video is a lack of transmission and storage bandwidth. Full-motion video (running at 30 fps) requires transmission

WORLD'S SMALLEST

486™  
DX

Embedded PC with  
Floating Point,  
Ethernet & Super  
VGA Only 4" x 4"

**The PC/II +i includes:**

- 486DX™ CPU at 25MHz or 33MHz clock frequency
- Full 8K Cache with Floating Point
- Ethernet Local Area Network
- Local Bus Super VGA Video/LCD
- Up to 2MBytes Flash™ with TFFS
- 4 or 16MBytes User DRAM
- PC/104 or ISA Bus compatible option (with adapter)
- 4" x 4" Format; 6 watts power consumption at +5 volt only

486DX and Flash are registered trademarks of Intel Corp as are PC, AT, IBM, megatel of Megatel Computer (1986) Corp

(416) 245-2953

megatel™

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



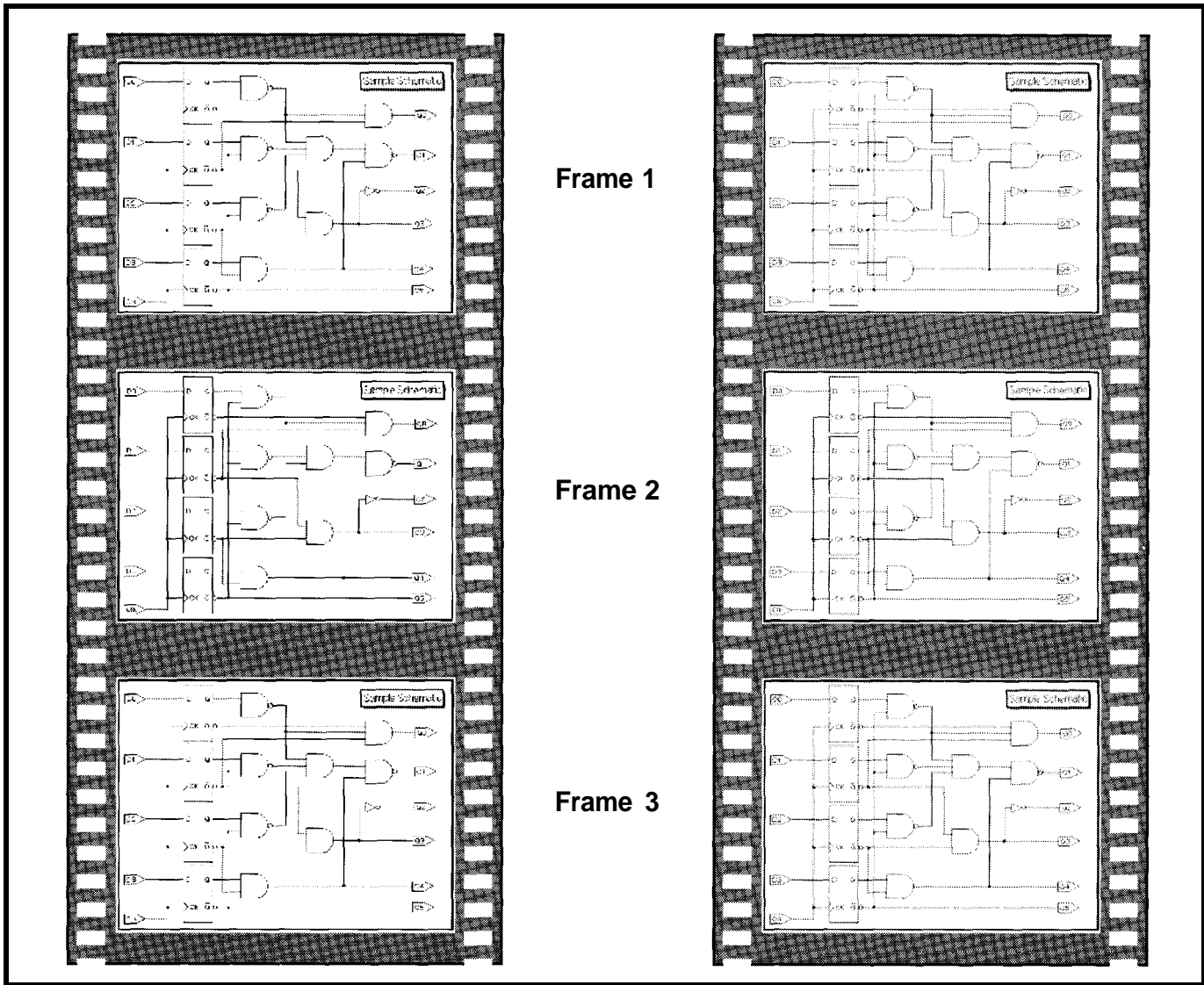


Figure 5—In a motion video sequence, reduced to 47% of its original size, bilinear interpolation (left) demonstrates a high degree of temporal image distortion while the technique used by the Acuity Resizing series maintains image integrity. The source image for the middle row is shifted vertically and horizontally one pixel each way. The source image for the bottom row was shifted vertically and horizontally two pixels each way.

bandwidths a full magnitude larger than those required for static images. The MPEG video compression standard has reduced this barrier. MPEG-1 and MPEG-2 use motion compensation, discrete cosine transformation, and entropy coding techniques to achieve extremely high compression rates. Many new Hollywood films are now being encoded with MPEG-1 and stored on CD-ROM for playback by computer and home-video playback machines.

Figure 4 is a block diagram of an MPEG compression system enhanced by scaling. Analog video is digitized and resized to reduce the amount of data prior to compression. Next, the data is crunched and saved to disk.

To display previously stored MPEG compressed video, the process is reversed. A compressed video file is read, decompressed, and resized to its original size (or any other desired size) by a resizer. After scaling, the video is converted back to analog format and displayed on a monitor.

An MPEG engine can perform optimal compression only on high-quality video received after video scaling. If a lower-quality method is used, the compressed file size increases because the MPEG engine works harder to encode the high-frequency artifacts introduced by aliasing. With inferior resizing techniques, a slight shift in the input source can completely change the

resulting resized image content. In a motion video sequence, this can result in image flickering or scintillation.

To simulate camera movement in Figure 5, the source image was shifted one pixel horizontally and vertically prior to resizing. As you can see, bilinear interpolation demonstrates a high degree of temporal image distortion while the techniques used by the Acuity Resizing chips maintains image integrity.

The ability to finely control the image size in the video stream before and after compression has many applications. For example, if you have a fixed satellite or cable transmission bandwidth, you can transmit only a certain number of compressed video

channels. Scaling video prior to compression enables you to dynamically fit more channels into the same fixed bandwidth. Experiments have yielded improvements in compression throughput of up to 40%, depending on image content.

A video conferencing application can benefit in a similar manner.

Normally, the entire data channel is devoted to transmitting the faces and voices of the two participants. Using this video resizing technique adds an extra communications channel so participants can share documents or other information while conversing.

### AN EVOLUTIONARY STEP

Digital video is the next evolutionary step in modern computer operating systems. The combination of video and computer graphics is creating entirely new classes of applications in computer-based training, entertainment, and communication.

This integration relies on two complementary technologies: video scaling and video compression. A new generation of scaling and compression algorithms gives users unprecedented power to efficiently manipulate, transmit, and store high-quality digital video.

Integrated video and graphics are here to stay—they deserve to look their best!

*James Goel, B.A.Sc., is a senior video DSP engineer at Genesis Microchip. A former software and automation engineer, he spends his days researching compression improvements through scaling technology. He spends his nights dreaming of DSP. He may be reached at (905) 470-2742.*

### CONTACT

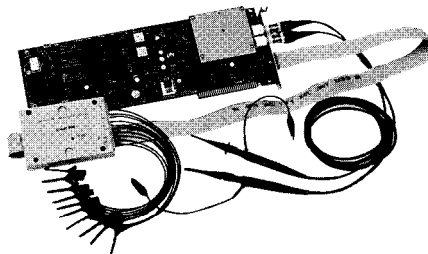
Genesis Microchip Corp.  
2111 Landings Dr.  
Mountain View, CA 94043  
(415) 428-4277  
Fax: (415) 428-4288

### IRS

407 Very Useful  
408 Moderately Useful  
409 Not Useful

## 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**



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

#### DSO Channels

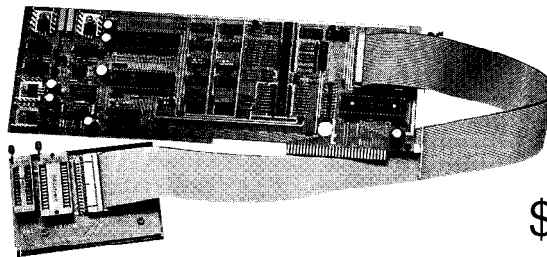
2 Ch. up to 100 MSa/s  
or  
1 Ch. at 200 MSa/s  
4K or 64K Samples/Ch  
Cross Trigger with LA  
125 MHz Bandwidth

#### Logic Analyzer Channels

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

## Universal Device Programmer

PAL  
GAL  
EPROM  
EEPROM  
FLASH  
MICRO  
PIC  
etc..

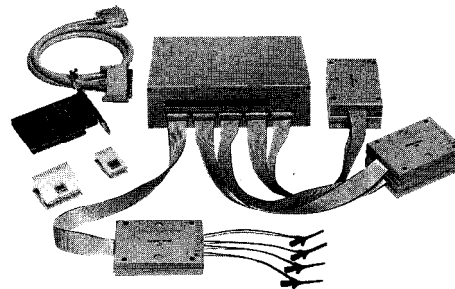


**\$475**

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 - LA121 00 (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



**Link Instruments**

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

# FEATURE ARTICLE

Chuck McManis

## A PIC-based Motor Speed Controller

To avoid the prohibitive cost of a packaged solution, Chuck takes the challenge to build his own DC motor speed control. His solution uses a PIC chip in a MOSFET H-bridge implementation.

**C**ontrolling the motor that moves the platform is a challenge faced by all mobile-robot builders. There are prepackaged solutions, which come in one of four forms: discrete components, monolithic motor-driver chips, electronic speed controls for R/C cars, and industrial controls. Unfortunately, each solution costs an order of magnitude more than the previous one. Many small companies are forced to build their own from discrete components or buy a motor control chip.

After building several controllers, I bought a commercial electronic-speed controller (ESC) for R/C cars. The controller had fully proportional forward and reverse control for a motor that could draw 10 A and it could be controlled with just one bit.

However, it cost \$89 on sale, and I needed two. The combination of high cost and high desirability had an inevitable effect—I had to figure out how to build it for less.

This article describes my project. After defining the problems in building an ESC, I describe the program I wrote for the PIC 16C54 which implements the servo controller mechanism. The conclusion looks at a couple of H-bridges I built (one cheap and one less so) that combine with the controller to give a fully functional ESC.

### THE CHALLENGES

The controller needed to be able to reverse and have proportional speed control over fractional-horsepower DC motors. The control input was a pulse whose width controls both the polarity and duty cycle of the output.

I chose to use the convention established for the control of R/C servos and commercial electronic speed controls which defines a pulse of 1500  $\mu$ s as the device's neutral or center position, a pulse of 2000  $\mu$ s as the full positive extension or actuation, and a pulse of 1000  $\mu$ s as the full negative extension or actuation. The pulse is repeated not more than once every 5 ms and not less than once every 20 ms for continuous actuation.

Figure 1 represents this problem in the form of black-box components. The arrows in this picture represent interfaces that can be defined for maximum flexibility. The first arrow, my challenge, is the "standard" servo interface. Although I've never seen an official standard, this is used by every servo or ESC I've seen or read about.

The servo controller in Figure 1 functionally consists of three parts: a pulse-width measurer (i.e., a pulse-capture unit), a PWM clock source that can be modulated, and an expiration timer to shut things off when an input hasn't been seen for a while.

The motor driver is simply a switch providing current to the motor based on what the servo controller asks for. This driver circuit is known as an H-bridge because it looks like an

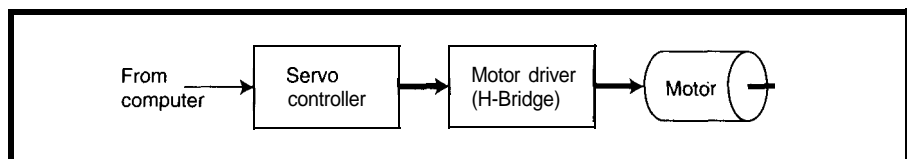


Figure 1—In the block diagram of typical ESC components, the arrows represent interfaces between the components.

H when diagrammed. The functional pieces are shown in Figure 2.

## DESIGNING THE SERVO CONTROLLER

As you can see, the servo controller can be implemented in several ways. You might build it out of individual components or out of a single FPGA. However, the design tools for FPGAs are priced in the stratosphere. PIC programmers, on the other hand, sell for \$200 and the development software is free.

The trick is to design a program that implements a servo controller. I first defined the servo controller as a state machine and then wrote an algorithm to implement a state machine. Next, I coded the algorithm in PIC assembly code to determine how fast I could get it to run. I then defined the timing constants.

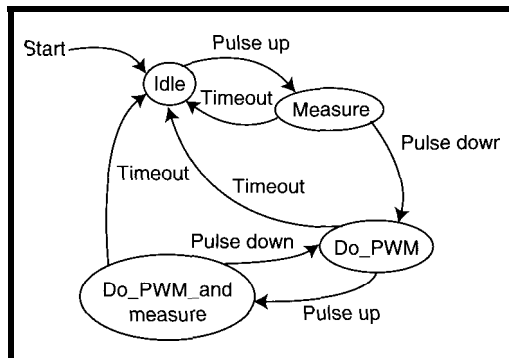


Figure 3—The first pass at the controller state machine is accurate, but doesn't necessarily reflect the best way to implement the application.

My first cut at designing a simple state machine that could implement the servo mechanism is shown in Figure 3. The state machine starts in an idle state. In this state, the processor monitors the status of the input pin. When the pin goes high, it switches to the Measure state.

In the Measure state, one of two things can happen:

- 1) the pin can go low, see the complete pulse, and move to the Do\_PWM state
- 2) the pin can stay high for longer than 2 ms and appear as a bogus input.

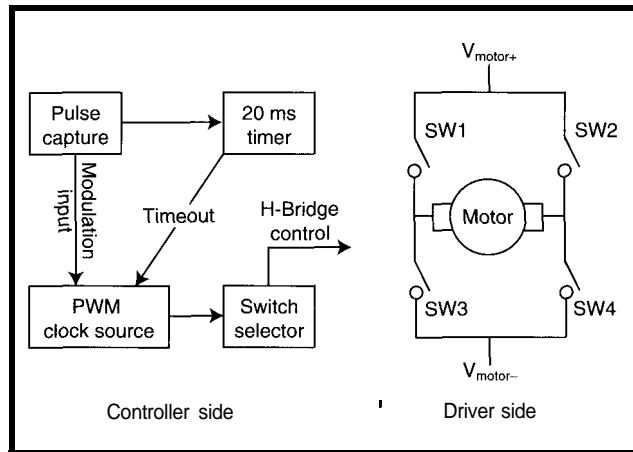


Figure 2—Adding detail to the block diagram in Figure 1 shows the functions contained in the ESC controller and driver boxes.

The state machine treats it as a timeout and returns to Idle.

Once it enters the Do\_PWM state, the state machine computes the duty cycle of the output waveform and sends it to the H-bridge.

This state exits under two conditions: either 20 ms elapses without a new pulse or the input pin goes high indicating an incoming pulse. When the input goes high, the state machine enters the fourth state—Do\_PWM\_and-Measure.

Here, it simultaneously generates the PWM output for the H-bridge and monitors the input pulse width. If a valid pulse width is received, it sets the pulse width being generated to the new value and returns to the Do\_PWM state. An invalid pulse times it out and returns it to Idle.

However, after coding this, I realized a better state machine could be used that had four states. As Figure 4 indicates, the states are now Idle, Idle\_and\_Measure, PWM, and PWM\_and\_Measure.

Although this looks more complicated than the first one, implementing it is easier. This model lets me factor out the measurement operation, leaving me with two main loops: Idle and PWM. Listing 1 offers

the first algorithm for taking measurements.

The **Take-Measurement** algorithm simply checks the state of the input pin. If it is high, it increments the width counter. If the input pin is low and the width counter is less than or equal to the minimum number of clicks for a valid pulse, it calculates the requested PWM duty cycle and direction and returns an indication of a valid result.

The trick to this algorithm is to call it at regular intervals. If you know the interval at which it is called, say every 10  $\mu$ s, then the conversion between the variable **Width** and the actual width of the pulse is simply **Width** times the calling interval.

The constants can also be determined using a 10- $\mu$ s interval. The midpoint of 1500  $\mu$ s corresponds to a width count of 150 (150  $\times$  10 = 1500). **MAX\_WIDTH** is then 200, and **GLITCH\_COUNT** is 9 (900  $\mu$ s).

Once coded, calculate the theoretical minimum interval between calls:

```

Loop:
  call Take_Measurement
got0 Loop
  
```

Knowing how many clock cycles **Take-Measurement** uses and the overhead of calling it yields the exact interval. Remember you can never achieve this maximum because the controller has to measure the input pin

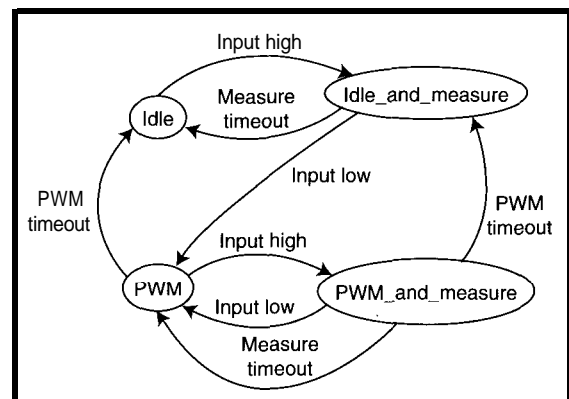


Figure 4—The modified state diagram for the controller's state machine is much more symmetric. It leads to the software design used

while simultaneously generating a PWM waveform, which limits its response time.

I chose to code the Idle and Idle-and-Measure states as a single loop that calls Take\_Measurement on a regular interval. With two loops, I could have used the PIC's low-power sleep mode, which cuts power consumption to 2  $\mu$ A. However, this requires a more expensive 16C71, the first 18-pin device that can be awakened. Since the PIC draws less than 5 mA when idling in a loop, the benefit doesn't justify the cost. The algorithm for the Idle loop is in Listing 2.

Take-Measurement0 processes the input pin and calls Delay(). This ensures that Delay() is called regularly. The call is also needed because later we call Take\_Measurement() from within the PWM loop, and the time it takes to process the PWM waveform is nonzero.

The main loop is the PWM loop, which generates the variable-width square wave as well as watching for other incoming pulses. As you can see in Listing 3, there is a lot more going on in the PWM loop around the Take-Measurement0 call, which limits how often it's called.

## WRITING THE CODE

The only task left for the servo controller is to convert the algorithms into code. The code is written using Microchip's MPASM assembler (see Microchip's data sheet for a quick overview of the PIC instruction set and syntax). I've included most of the interesting code here in the listings, but you'll want to download the complete code from the Circuit Cellar BBS to fill in the holes.

The most sophisticated part of the software measures the width of the incoming pulse. As mentioned, the design requires the measurement routine (see Listing 4) to be called on a regular basis by the state loops.

This code executes in only 14 clock cycles. Execution time of this routine determines the frequency of generating the PWM signal.

DO-MEASURE has two jobs to perform: it must accumulate time when the input pin is high and track

Listing 1—The Take\_Measurement pseudocode measures the incoming PWM duty cycle and direction.

```

Take-Measurement:
  if (InputPin == 1)
    Width = Width + 1
    if (Width > MAX-WIDTH) do
      Timeout = TRUE
    end
  else do
    if (Width <= GLITCH-COUNT)
      return NO-PULSE
    else do
      PWM_Constant = Width MIDPOINT
      if (PWM_Constant < 0)
        Direction = REVERSE
      else
        Direction = FOWARD
      PWM_Constant = abs((PWM_Constant / MAX-CONSTANT) * MAX-WIDTH)
      Width = 0
      return GOT-PULSE
    end
  end
end
return NO-PULSE

```

Listing 2—The Idle loop pseudocode calls Take\_Measurement on a regular interval.

```

Idle:
  while (Take_Measurement() != GOT-PULSE) do
    call Delay();
  end
NewState(PWM)

```

Listing 3—The motor PWM signal is generated completely in software.

```

PWM:
  while (time < 20 ms) do
    for count = 0 to PWM_CONSTANT do
      if (count < PWM_Width)
        output(Direction) // turn on the H-bridge
      else
        output(OFF) // turn off the H-bridge
      if (Take_Measurement() == GOT-PULSE)
        goto PWM // restart the loop
      end
    end
  end
NewState(Idle)

```

the input pin from a low-to-high-to-low state, which indicates a control pulse has been received. Once it recognizes a control pulse, the width is calculated and the necessary pins are set to turn on the H-bridge. The code must also recognize and screen out invalid pulses. My first challenge was checking for invalid pulses in the fewest possible clock cycles.

Invalid pulses come in two flavors. If they're too short, a glitch occurs; if

too long, there's a timeout. I've used countdown timers to detect both.

The counter GLITCH-COUNT detects "too short" pulses. The counter initializes to the minimum count value needed to recognize a valid pulse. Then, while the input pin is high, this counter decrements until it reaches zero, where it is no longer decremented. The Boolean condition (GLITCH\_COUNT == 0) is true when the pulse is longer than the initial value of



GLITCH\_COUNT, and false if the pulse is shorter (see Listing 4). If GLITCH\_COUNT is non-zero, NO-PULSE is executed.

Timeout, the second invalid pulse, occurs when the input pin remains pulled high. Given the nature of the circuit, this pulse occurs when the circuit input is left disconnected. It can also occur if the controller stops operating correctly. In either case, the controller should turn off the outputs since timeout is not valid. To detect a timeout, I use the width counter implemented in COUNT\_LO and COUNT\_HI.

If you think that by clearing the counter to zero and counting up while the pin is high gives the width of the input pulse, you're correct. As a 16-bit counter, it could measure pulses much wider than the ones we are interested in. However, to detect overflows, we'd be forced to do 16-bit subtraction of some maximum legal value and then see if a borrow was generated—a solution far too time consuming.

Instead, I preload the counter with the maximum legal value and decrement it. If the pulse is too wide, the counter underflows. Timeout simply checks to see if bit 7 (sign bit) of the counter is true (counter has gone negative). Thus, with a glitch or timeout, the code reloads the counters and looks for the next valid pulse.

Note that I don't just branch to NO\_PULSE when I detect a timeout since that would make that path through the code take one clock less than the code that detects a glitch. In all cases, the time it takes to process a measurement must remain the same.

When the input is in a true state, the width count accumulates (Listing 5) by decrementing GLITCH\_COUNT, COUNT\_HI, and COUNT\_LO as needed. The only tricky bit is the down counter. Since the PIC does not set the carry flag when a decrement underflows a register, the loop must check the low byte for zero. At zero, the next decrement underflows and affects the high byte. A nice property of this technique is that the 16-bit decrement operation only takes four clock cycles.

As you can see from the code, once the down counter has under-

Listing 4—The first part of the pulse measurement code screens out invalid pulses.

```
DO-MEASURE:
    BTFSC  SERVO-IN      ; Check for high
    GOTO   INP_HIGH     ; Yup, count it.
    MOVF  GLITCH,F      ; Check Glitch counter
    BTFSS  ZERO-BIT     ; != 0 means Glitch.
    GOTO   NO-PULSE     ; This was a glitch
    BTFSS  COUNT_HI,7   ; Check for overflow
    GOTO   GOT-PULSE    ; Process a good count
    MOVLW  GLITCH-COUNT ; Alternate NO-PULSE entry
    MOVWF  GLITCH       ; Store it in GLITCH
    MOVWF  COUNT-LO     ; Just a glitch,
    MOVLW  1            ; Load count with constant
    MOVWF  COUNT-HI
    RETLW  0            ; And return

NO-PULSE:
    MOVLW  GLITCH-COUNT
    MOVWF  GLITCH
    MOVWF  COUNT_LO
    MOVLW  1
    MOVWF  COUNT_HI
    NOP
    RETLW  0
```

flowed (bit 7 is set), the code stops accumulating counts and branches to INP\_TIMEOUT. This prevents a rollover of the down counter should the high pulse remain for a long time.

A valid input pulse is taken care of by GOT\_PULSE (Listing 6). This routine converts the counted width of the pulse into the parameters needed for the H-bridge. The challenge here is to have the pulse width (depending on its width) change what the controller is doing by setting the value of command variables PWM\_CMD and CMD\_REG.

Since there is a large interval between valid pulses, this code takes more clocks. The code shouldn't take too long, however, as the motor may be on when it is called and this "on time" is added to the current pulse.

This code also controls the minimum time between legal pulses that the controller can correctly respond to. Although the design specifies a minimum of 5 ms, the code implements it in under 1 ms. Finally, the code specifies the time constants starting with GLITCH\_COUNT.

Listing 5—The second part of the pulse measurement code accumulates a width count when the input pin is true.

```
INP_HIGH:
    MOVF  GLITCH,F      ; 3 clocks have executed when we enter
    BTFSS  ZERO-BIT     ; Check Glitch
    DECF  GLITCH        ; Is it already 0?
    BTFSC  COUNT_HI,7   ; No, then decrement it
    GOTO   INP_TIMEOUT  ; Not a timeout already
    MOVF  COUNT_LO,F    ; Its a timeout
    BTFSS  ZERO-BIT     ; Test COUNT-LO for zero
    DECF  COUNT_HI,F    ; If its zero, this is it
    DECF  COUNT_LO,F    ; Subtract one from COUNT_HI
    RETLW  0            ; Subtract one from the count

INP_TIMEOUT:
    NOP                ; 9 clocks have executed when we enter
    NOP                ; Filler to get 14 clocks
    NOP                ; Filler to get 14 clocks
    RETLW  0            ; Filler to get 14 clocks
```

Listing 6—The third part of the pulse measurement code computes the portions of the H-bridge to turn on and the PWM constant.

```
GOT- PULSE:
MOVW  D'128'           ; This is the midpoint.
SUBWF  COUNT_LO,F
BTSS   CARRY-BIT      ; Carry is set (no borrow)
GOTO   DEAL-WITH-MINUS ; Figure out what to do now.
POSITIVE-RESULT:
MOVW  REVERSE_CMD     ; It's probably reverse
MOVWF  CMD_REG        ; Store it in command register
NEW_PWM_VALUE:
MOVW  DEADBAND       ; Dead band is ±6 (50 µs)
SUBWF  COUNT_LO,F    ; Subtract DEADBAND from count
BTSS   CARRY-BIT     ; If no borrow, continue
GOTO   DO-BRAKE      ; In the dead zone do braking.
MOVW  0'100'         ; Check if its in the range
SUBWF  COUNT_LO,W    ; Subtract 99 (max range)
BTSS   CARRY-BIT     ; Check if its Less than MAX
GOTO   MAX_PWM       ; Was >= 100
MOVF  COUNT_LO,W     ; Get count value.
<< FALL THROUGH TO DONE-PULSE >>
```

If the midpoint of the count is 1500 us and a function is called every 25 clocks or 5 us, then at the midpoint, **DO-MEASURE** has been called 300 times (300 x 5 us = 1500 µs).

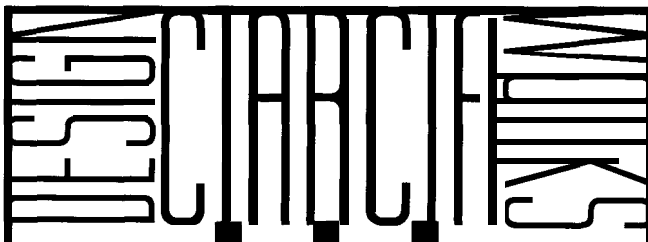
Also, since the midpoint corresponds to the "middle" value, it can be

represented by an 8-bit number (i.e., 128). Meeting this requirement ensures the most dynamic range in the system—it is able to count 128 counts before or after the midpoint. Thus, our measurement checks pulses from 1500 – 640 us through 1500 + 640 us to the

nearest 5 µs. Given these numbers, we calculate **GLITCH\_COUNT** to be the midpoint less the minimum pulse or 172 calls (300 calls – 128 calls).

After the first calculation (width – midpoint), the result is a number between -128 and +128. From this number, we can get the direction to set the H-bridge. If the number is negative (a wide pulse), the direction is forward. If the count is positive (a narrow pulse), the direction is backward. The only tricky bit here is that the count in **COUNT\_LO** and **COUNT\_HI** can be 100h on the exact minimum pulse. When this occurs, subtracting 128 from **COUNT\_LO** gives -128 (0 + carry) instead of the desired +128.

Listing 7, known as **DEALWITHMIS**, handles negative values. This code sets the direction of the H-bridge forward and takes the absolute value of the counter, making it a positive value. However, this code also checks for the special case of +128. Once the count value is normalized, execution resumes as the **POSITIVE\_RESULT** of Listing 6's **GOT-PULSE**.



Does your **Big-Company** marketing department come up with more ideas than the engineering department can cope with!

Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?

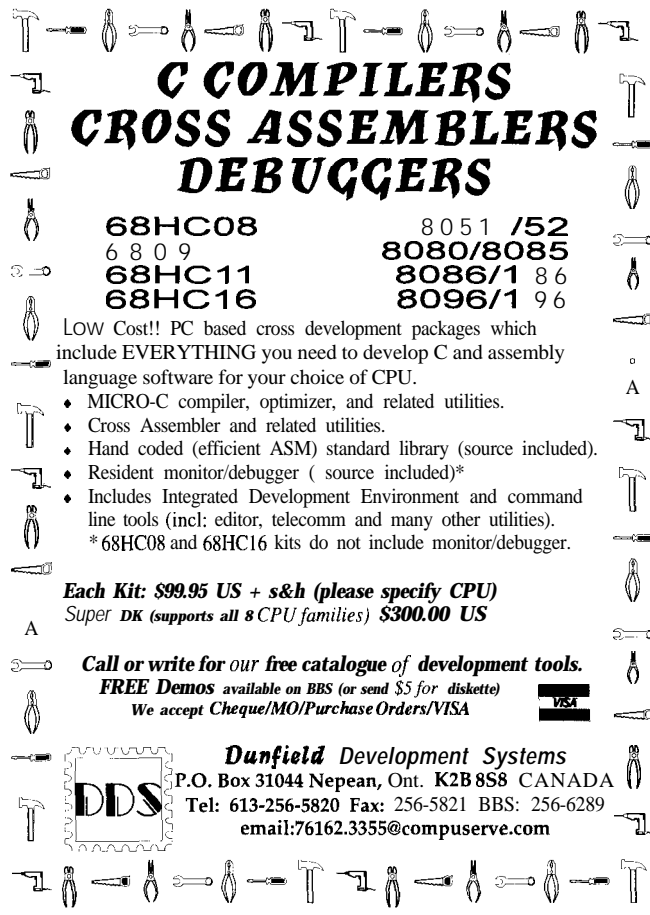
Steve Garcia and the Ciarcia Design Works staff may have the solution. We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems.

Whether you need an on-line solution for a unique problem, a product for a startup venture, or just experienced consulting, the Ciarcia Design Works is ready to work with you! Just fax me your problem and we'll be in touch.

REMEMBER A

**CIARCIA DESIGN WORKS**

FAX: (203) 871-8986



## C COMPILERS CROSS ASSEMBLERS DEBUGGERS

68HC08	8051 /52
6809	8080/8085
68HC11	8086/1 86
68HC16	8096/1 96

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 included).
- Resident monitor/debugger ( source included)\*
- Includes Integrated Development Environment and command line tools (incl: editor, telecomm and many other utilities).

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

**Each Kit: \$99.95 US + s&h (please specify CPU)**  
**Super DK (supports all 8 CPU families) \$300.00 US**

**Call or write for our free catalogue of development tools.**  
**FREE Demos** available on BBS (or send \$5 for diskette)  
 We accept Cheque/MO/Purchase Orders/VISA

**Dunfield Development Systems**  
 P.O. Box 31044 Nepean, Ont. K2B 8S8 CANADA  
 Tel: 613-256-5820 Fax: 256-5821 BBS: 256-6289  
 email:76162.3355@compuserve.com

**Listing 7**—When the input pulse is more than 1500  $\mu$ s wide, this code takes the absolute value of the delta. It also checks the special case of the absolute minimum pulse width.

```

DEAL- WITH- MINUS:
  BTFSCL COUNT_HI,0      Is bit 0 a one?
  GOTO  ADJUST-RESULT    Yes, so adjust
  MOVLW  FORWARD_CMD    It's wider than the midpoint
  MOVWF  CMD_REG         So this is the command.
  COMF   COUNT_LO,F      Convert to a positive value
  INCF   COUNT_LO,F      negate COUNT-LO
  GOTO   NEW_PWM_VALUE   Now go compute PWM constant.
ADJUST- RESULT:
  MOVLW  D'128'          256 128 = 128
  MOVWF  COUNT_LO        Fixup the low byte
  GOTO   POSITIVE-RESULT Now treat it normally.

```

**Listing 8**—When the pulse width is within the deadband, the H-bridge is placed in a braking mode.

```

DO- BRAKE:
  MOVLW  DEADBAND        Restore Count Value
  ADDWF  COUNT_LO,F      Add it back to Count
  MOVLW  DEADBAND-1     Check for boundary condition
  SUBWF  COUNT_LO,W      Subtract COUNT LO
  BTFSCL ZERO-BIT        Check to see if it's zero
  GOTO   ON-THE-EDGE     It's exactly DEADBAND - 1
  MOVLW  BRAKE_CMD       This is the brake Command
ON-THE- EDGE:
  MOVWF  CMD_REG         Store it
  MOVLW  D'100'          Very large PWM Constant
  MOVWF  PWM_CMD         So we get 100% braking action
  GOTO   DONE-PULSE      Return

```

**Listing 9**—The loop state must be initialized with a new PWM constant and H-bridge mode before reentering the PWM loop.

```

DONE- PULSE:
  MOVWF  PWM_CMD        ; This is the new PWM Constant
  INCF   PWM_CMD        ; Adjust 0-99 -> 1-100%
  MOVLW  1               ; Put 1 into INNER_CNT
  MOVWF  INNER_CNT      ; Exit inner loop on this iteration
  MOVWF  PWM_TMP
  MOVWF  COUNT_HI       ; Reset counter
  MOVLW  GLITCH-COUNT   ; for next time
  MOVWF  COUNT-LO
  MOVWF  GLITCH
  MOVLW  D'41'          ; Outer Loop count + 1
  MOVWF  OUTER_CNT      ;
  RETLW  1               ; Return true (for idle)

```

Starting at POSITIVE\_RESULT, the next step is to determine the dead band where the motor controller brakes the motors rather than causing them to move forward or backward. I previously decided to have a proportional zone 100 counts wide as this makes the value in PWM\_CMD a percentage of the duty cycle.

I also wanted a full-power zone in which the controller does not do any PWM at all but simply switches the

bridge fully on. I chose a dead band of 5, which gives a total dead band of 10, since it's taken out of both the positive and negative values. A dead band width of 10 or 50  $\mu$ s (10 x 5  $\mu$ s) is easily attainable. Even with a fairly inaccurate driving circuit, this gives you a  $\pm 25 \mu$ s margin of error.

In Listing 6, the width of the dead band is subtracted from COUNT\_LO. If it goes negative (an indication that the pulse width is in the dead band), the

code branches to DO\_B BRAKE, the routine included in Listing 8.

In testing this routine, I discovered the system alternately brakes and applies a 1% duty-cycle square wave when the input pulse is near the point that going forward or backward begins and braking engages. Although this low-speed oscillation isn't particularly harmful, it is annoying.

To correct it, I added a test in DO-BRAKE for reading a pulse width on one or the other edge of the dead band. When this tests true, I change command to off or zero.

By turning the bridge off, the motor simply coasts. Because of this modification, the output goes between 1% PWM and coast when the input alternately picks up the minimum power level (1% PWM) and brake. When the input is on the other edge, approaching the edge of the dead band, the output varies between brake and coast. Dealing with the boundary condition in this way provides a more stable system overall.

If COUNT\_LO is greater than the dead-band value, I continue ahead in the POSITIVE\_RESULT code. Since I've subtracted the dead-band value, the value in COUNT\_LO, which was 0-128, is now 0-123. However, any value greater than 100 is "full on" as far as the code is concerned.

The final step limits the output to 100 by comparing the value in COUNT\_LO to 100. If it is greater than 100, the carry bit is set and the code branches to MAX\_PWM:

```

MAX_PWM:
  MOVLW  D'100'
  GOTO   DONE-PULSE

```

MAX\_PWM simply sets the value to 100 (its maximum legal value) and reenters execution at DONE\_PU\_LSE, where all paths through the code terminate. DONE-PULSE finalizes all of the setup and is shown in Listing 9.

DONE\_PU\_LSE stores the computed PWM constant into PWM\_CMD and manipulates the control variables of the PWM\_LOOP code (see Listing 10). This latter step is required because a valid pulse can be received in the Idle and PWM loops.

**Listing 10**—The outer PWM loop sets the loop count to 40, which causes the if to run for 20 ms. The inner loop sends commands to the H-bridge.

```

PWM:
MOVLW D'40' ; Constant for 20 ms
MOVWF OUTER_CNT ; Outer Count
MOVF PWM_CMD,W ; Set up the inner loop counters
MOVWF PWM_TMP
MOVF CMD_REG,W ; Get H-Bridge Command
MOVWF CMD_TMP ; Put it in our temporary holder
MOVWF 'BRIDGE-OUT ; Turn on the motors
MOVLW D'99'
MOVWF INNER_CNT ;
GOTO PWMLOOP ; Line up command

PWM_LOOP:
CALL DO-MEASURE ; Do a "measurement" +
DEC F PWM_TMP,F ; Decrement the "On" time |
BTFSZ ZERO-BIT ; If not zero continue |
SWAPF CMD_TMP,F ; Turn off the low side | Inner
MOVF CMD_REG,W ; Get the Command Register | Loop
MOVWF BRIDGE-OUT ; Send it To the H-Bridge |
MOVF PWM_CMD,W ; Sort of a NOP |
DECFSZ INNER_CNT,F ; Decrement inner loop count +
GOTO PWMLOOP ; Continue Looping

MOVWF PWM_TMP ; Restore command (loaded in inner loop)
CALL DO-MEASURE ; This is on a 5 µs boundary
MOVLW D'99' ; Reset inner loop count
MOVWF INNER_CNT ; Like so
MOVF CMD_REG,W ; Put command into W
MOVWF CMD_TMP ; Reinitialized CMD_TMP
MOVWF BRIDGE-OUT ; Send it to the motors
DEC F OUTER_CNT ;
BTFSZ ZERO-BIT ;
GOTO PWM_LOOP ;

; Unrolled Idle loop

CLRF BRIDGE-OUT ; Turn off the motors
CALL DO-MEASURE ;
MOVWF TEMP ; Store result in TMP
MOVF TEMP,F ; Waste time, set Z bit
BTFSZ ZERO-BIT ; If zero no new value
GOTO PWM_LOOP ; If got a new value, then start
GOTO $+1 ; else Waste time
NOP ; ...
GOTO IDLE ; Go back to measuring

```

If the code is in the Idle loop, this step is unnecessary since the subsequent jump into the PWM code sets these variables. However, in the PWM loop, the code resets the variables so the PWM loop code believes that this is the first time through the loop.

The rest of the code is relatively straightforward and consists of three major chunks: PWM outer loop, PWM inner loop, and idle loop. The next chunk of code I'll describe is Listing 10's PWM outer loop.

This idea behind the code is simple: turn on the H-bridge, run the loop 100 times, count down the PWM

constant, and when it gets to zero, turn off the H-bridge. If the PWM constant is 1, the H-bridge is on for 1% of the time spent in the inner loop. If the constant is 2, the power is on 2% of the time. This continues until you reach 100, at which point the power to the H-bridge is on 100% of the time.

Running the loop 100 times determines the period of the output waveform. Since the least amount of time I can execute the loop is 5 µs, the period of the resulting waveform is 500 us (100 x 5 µs), which is a frequency of 2000 Hz. To modify the code for only 50 speed steps, use 250 us or 4000 Hz.

The PWM inner loop is distinct from the outer loop. The outer PWM loop times out the controller after 20 ms elapses with no new input arriving. As demonstrated, the inner loop executes in 500 µs, so 40 iterations of the inner loop takes 20 ms. When the outer count is decremented to 0, 20 ms has passed. The controller should switch the output off. However, if a valid pulse is received, the counter resets to 40 in DONE-PULSE, which causes it to run another 20 ms.

My pulse measurer design requires that DO-MEASURE be called as often as possible and always at the same interval. In an interrupt-driven system, a timer would go off every few milliseconds and call DO-MEASURE. But on this PIC, I don't have that luxury. Because of this limitation, the critical issue in the PWM loop is to call the DO-MEASURE routine on a regular schedule. This function goes to extraordinary lengths to maintain symmetry in both the inner and outer loops using a technique called limited loop unrolling.

The inner loop executes 100 times. However, by executing it 99 times and then recoding the same contents of the loop after the loop finishes, I can use the clock cycles that the inner loop normally uses to compute whether or not the outer loop is done. Notice in the listing that the instructions at the bottom of the outer loop are identical to the ones in the inner loop, except that they now compute something for the outer loop instead.

The extra clock used by the GOTO PWM\_LOOP instruction in the inner loop is used in the outer loop to reset the value in PWM\_TMP. Next, the inner loop calls DO-MEASURE, so the outer loop does as well. The inner loop then decrements PWM\_TMP to see if the output should be switched off.

The outer loop uses these cycles to reset the inner loop count and to send the command to the H-bridge. The inner loop sets the H-bridge outputs (as does the outer loop), then decrements its counter. If the counter is not yet zero, it circles back through its loop. In the meantime, the unrolled version in the outer loop decrements

the outer loop's counter and restarts the inner loop if that isn't zero. In this way, **DO-MEASURE** is called every 5  $\mu$ s.

The only other interesting bit in the inner loop is how it turns off the motor. Remember that after the PWM constant reaches zero we need to turn off the motor. Since the bridge takes four bits of output and Port A on the PIC is a 4-bit output port, an alternate command is stored in the upper four bits of the command register.

When **PWM\_TMP** crosses zero, I swap the halves of the command byte. The next time it is sent to the bridge, the low-side driver has been turned off.

The one extra instruction available from not branching back into the PWM loop turns off the outputs to the bridge. Next, I call **DO-MEASURE** right on schedule and check to see if a new pulse has arrived. If it has, I reenter the PWM code. If it hasn't, the code branches to idle after ensuring that the number of clocks that have passed is sufficient so that when I *do* call **DO-MEASURE**, it is on another 5- $\mu$ s interval boundary.

Figure 5 illustrates the response of the system to different-width input pulses given the program and the constants chosen.

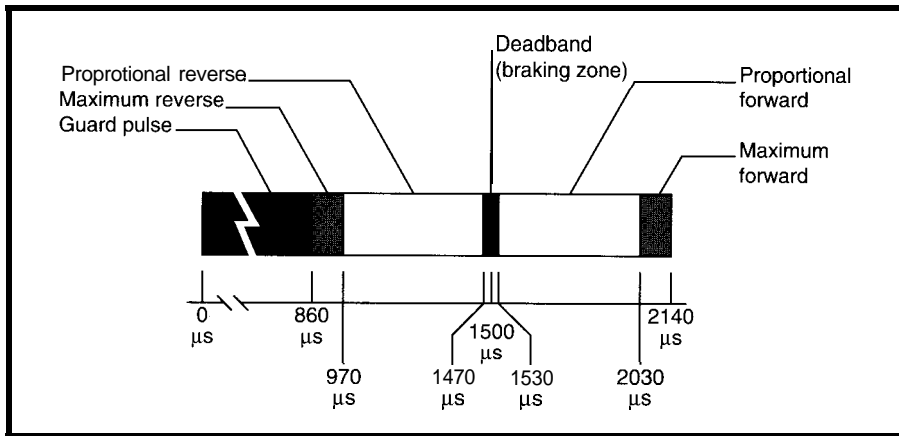


Figure 5-A graphic illustration clearly shows the response of the controller to different pulse widths

I leave the high-side driver energized because in FET designs it may be driven by a charge pump. Turning it off and on limits the efficiency of that circuit.

There is, however, one final bit of magic in this code which comes into play when the PWM loop finally exits. This final flourish takes care of the valid pulse which starts coming in when the PWM loop is running, but finishes after it exits. As you may have guessed, between the end of the PWM loop and reentering the idle loop, the code must continue to call **DO-MEASURE** every 5  $\mu$ s or risk getting bogus measurements.

Fortunately, there is a lot of extra time in the idle loop since all that is happening is that a potential incoming pulse is being measured. As the bottom of Listing 10 demonstrates, all I need to do is ensure that **DO-MEASURE** is called on schedule and that we reenter Idle as planned.

## DESIGNING HARDWARE FOR ESC

By doing the servo code in software, the hardware for the ESC is greatly simplified. There are two modules to the basic circuit: the servo controller and the motor driver or H-bridge. The servo controller drives one half of a quad optoisolator (see Figure 6). The other half is connected to an H-bridge and is implemented with bipolar (BJTs) or MOSFET transistors.

Notice that the servo module is controlled from the input into J1, the servo connector. This connector has three pins (V<sub>+</sub>, signal, and Gnd) and should nominally have 5 V on the power pin. The PIC is capable of using 4.5-5.5 V. The standard R/C receiver battery pack of four NiCD cells produces 4.8 V, which is acceptable.

Because of the PIC's flexibility, there is no regulator in the circuit. The square-wave pulse input to the servo connector falls within acceptable TTL levels (i.e., >3.0 V is a logic high). The

output pins of port A, RAO-RA3, connect to the LED anodes in the optoisolator. The LEDs then connect to 470-R current-limiting resistors.

Figure 7 shows the first H-bridge implementation. In the schematic, U1a-U1d represent the other half of the optoisolator from the servo module circuit. By using an optoisolator, the electrical noise generated on the motor's power-supply lines is kept out of the circuit's digital side. This greatly increases the reliability of the overall system, and, if you power the ESC from the same supply as your onboard computer, it protects the computer from interference as well.

The circuit uses the transistors as switches, thereby driving them to their saturation point. For this, the base current should be at least  $I_c/h_{FE}$ , amps. This prototype uses TIP120 (NPN) and TIP125 (PNP) transistors, which are power Darlington transistors with a minimum  $h_{FE}$  of 1000. For an 8-A current through the transistor, the base current ( $I_b$ ) needs to be about 8 mA. Use this figure and your motor battery voltage to choose the resistor value you need.

For example, if you have a 9.6-V motor power source (eight NiCd cells) and wish to get 8 mA through the resistor, you'd calculate:

$$\frac{9.6\text{ V} - 1\text{ V}}{0.008\text{ A}} = 1075\ \Omega$$

In my prototype I used a 1-k $\Omega$  resistor pack.

Four diodes, D1-D4, are reverse biased across the collector-emitter leads of the transistor to snub the reverse-current spikes occurring when an inductive load is switched off. The diodes can be any reasonable rectifier you have handy as long as they have fast recovery (<6.25  $\mu$ s—our minimum pulse width) and can handle a surge of 4-6 times the motor current. I've used 1N4002s with good success.

This circuit generally runs 3-5-A motors without needing a heatsink. However, if you draw the maximum current periodically (8 A), put heat-sinks on the transistors. Note that because this is a common-collector circuit, the motor voltage is present on the tabs of the transistor bodies. So,

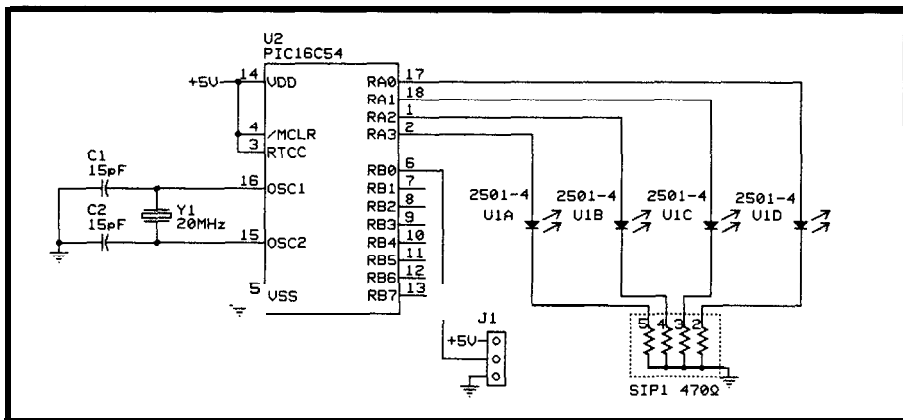


Figure 6—The ESC core uses a PIC processor. The interface between it and the electrically noisy H-bridge is a quad optoisolator. When built on a protoboard, its cost is about \$9 including sockets.

while Q1 and Q3 can go on the same heatsink, you must not put them on the same heatsink as Q2 or Q4.

Finally, the capacitor across the motor power leads filters noise generated by the motor's brushes and should be about 0.1  $\mu$ F.

The second implementation of an H-bridge is in Figure 8. In this circuit, N-channel, enhancement-mode power MOSFETs are used as switches. Given their geometry, MOSFETs handle higher currents than bipolar transistors. However, this feature is mitigated by the somewhat more complex drive requirements. Fortunately, using N-channels as high-side switches got a whole lot easier when Maxim introduced the MAX620 Quad High-Side Driver.

The MAX620's outputs are connected to the gates of the two high-side switches. The chip itself contains an integral charge pump that produces a voltage on the output pins that is  $V_{CC}+10$  V, sufficient power to turn on nearly any MOSFET.

By using logic-level MOSFETs (full enhancement mode with a  $V_{gs}$  of

just 5 V), this circuit uses motor supplies of 5-30 V. I used Philips BUKV455 MOSFETs, but the International Rectifier IRLZ44 is easier to come by and has a continuous-current rating of 35 A!

Selecting the resistors for this circuit is driven by the gates of the MOSFETs. The MOSFET's gate looks like a capacitor, so R3 and R4 have to be just small enough to drain off the gate charge when the optoisolator turns off. I found that 4.7-k $\Omega$  resistors worked quite well for a wide set of input voltages. At 5 V (worst case), the current through R3 when U1c is off is about 1 mA ( $V_{gs}/R3$ ), which is well able to turn off the MOSFET in the minimum 6.5  $\mu$ s.

If you don't use logic-level MOSFETs, you have two choices: increase the motor voltage so the  $V_{gs}$  appearing on the low-side switches is 10 V or more (e.g., a 12-V motor supply) or use the two unused drivers in the MAX620 to drive the low-side switches. Most MOSFETs have a  $V_{gs}$  max of 20 V, so if you use this second option, the motor voltage must be less

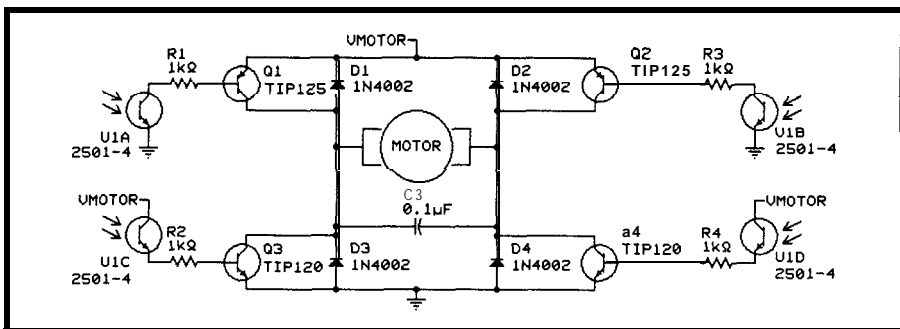


Figure 7—An inexpensive H-bridge can be made using bipolar transistors. The component cost is between \$5 and \$10. Its current capacity is around 5 A.

## Low Cost ICE for the '552

### Pods available for other 805 1 Family Processors:

The DryICE Plus is a modular emulator designed to get maximum flexibility and functionality for your hard earned dollar. The common base unit supports numerous 805 1 family processor pods that are low in price. Features include: Execute to breakpoint, Line-by-Line Assembler, Disassembler, SFR access, Fill, Set and Dump Internal or External RAM and Code, Dump Registers, and more. The DryICE Plus base unit is priced at a meager \$299, and most pods run only an additional \$149. Pods are available to support the 8031/2, 875 1/2, 80C154, 80C451, 80C535, 80C537, 80C550, 80C552/62, 80C652, 80C851, 80C320 and more. Interface through your serial port and a comm program. Call for a brochure or use INTERNET We're at [info@hte.com](mailto:info@hte.com) or [ftp](ftp://ftp.hte.com) at [ftp.hte.com](ftp://ftp.hte.com)

## Can you afford to debug 8031 code without it?

If you're debugging 8031/32 code our \$149 DryICE is what you need. Not an evaluation board - much more powerful. Most of the features of the DryICE Plus, but even lower in cost.

## You can afford it!

So, if you're still doing the U V (Cha-cha (1-2, burn, erase, 3-4, iburn, erase), or debugging through the limited window ROM emulators give, **call us now** for relief! Our customers say our emulators are the best Performance/Price value!

Look into our Single Board Computer solutions, too!

**HTE** HiTech Equipment Corp.  
9400 Activity Road  
San Diego, CA 92126  
(Fax: (619) 530-1458)

Since 1983

(619) 566-1 892



Internet e-mail: [info@hte.com](mailto:info@hte.com)  
Internet ftp: [ftp.hte.com](ftp://ftp.hte.com)

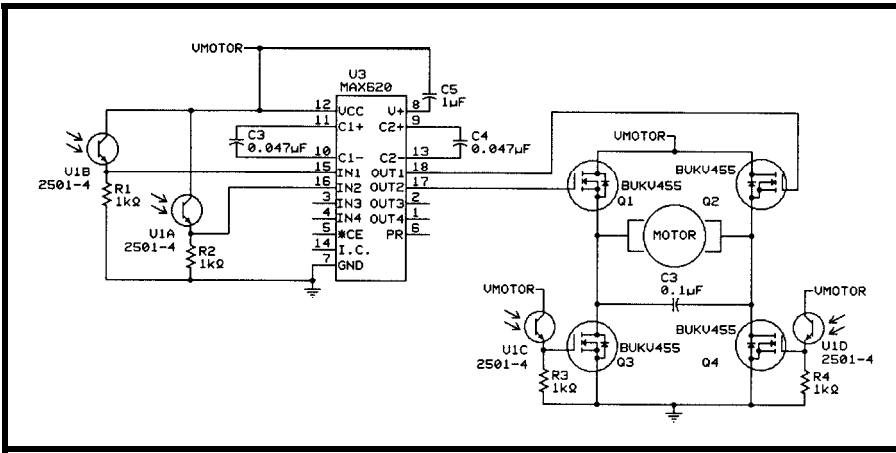


Figure 8-A more expensive H-bridge using N-channel MOSFETs is nearly three times as expensive at \$15-20, but it can handle seven times the current. Using MOSFETs or IGBT transistors, bridges that handle over 100 A are possible.

than 10 V (output = 10 V + 10 V = 20 V). Ideally, it should be in the neighborhood of 6 V.

The capacitors on the MAX620 are taken from the data sheet and should be 0.047  $\mu$ F for C1 and C2 and 1  $\mu$ F for the capacitor between V+ and the V<sub>CC</sub> pin. While these three can be eliminated by using the MAX621 chip, that chip is about twice as expensive.

## CONCLUSIONS

With this design, I can build a bipolar unit for less than \$20 and drive a pair of motors for less than \$50. The servo controller is generic enough that I can construct an arbitrarily large H-bridge for the output and tailor the interface for each unique motor I use.

While it currently uses only a single bit for input, this system could

easily be modified by simple changing the code to take a signed 8-bit value and translate it directly into a PWM output.

**Chuck McManis is an engineer (BSEE) who has been writing software for the last 20 years. He's currently responsible for networking and security for the Hot Java group at Sun Microsystems in Mountain View, CA. He may be reached at cmcmanis@sun.com.**

## SOFTWARE

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

## IRS

- 410 Very Useful
- 411 Moderately Useful
- 412 Not Useful

## 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" 2764/128 EPROM programmer, three 8-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.

### PROCESSOR

- 80C528-bit CMOS processor w/BASIC-52
- Three 16-bit counter/timers
- Six interrupts
- Much more!

### MEMORY

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

### INPUT/OUTPUT

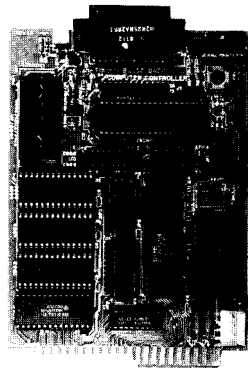
- Console RS232 - autobaud detect
- Lineprinter RS-232
- Three 8-bit parallel pods
- EXPANDABLE!
- Compatible with 12 BCC expansion boards

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

<b>BCC52</b>	Controller board with BASIC-52 and 8K RAM	\$189.00	Single Qty
<b>BCC52C</b>	Low-power CMOS version of the BCC52	\$199.00	
<b>BCC52I</b>	-40°C to +85°C industrial temperature version	\$294.00	
<b>BCC 52CX</b>	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!



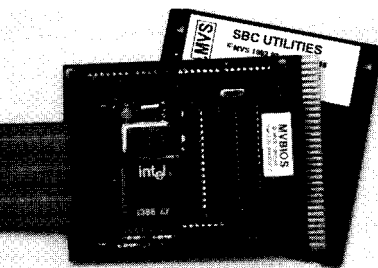
## 386 SBC \$83

OEM (1K) PRICE

INCLUDES:

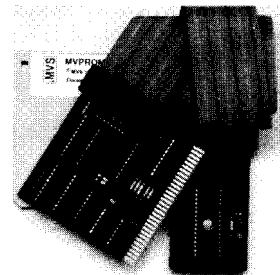
- 5.25" (8250 USART) PAR (32 BITS MAX)
- 32K RAM, EXP 64M
- STANDARD PC BUS
- LCD, KBD PORT
- BATT. BACK. RTC
- IR00-15 (8259 X2)
- 0237 DMA 8253 TMR
- BUILT-IN LED DISP.
- UP TO 8 MEG ROM
- CMOS NVRAM

USE TURBO C,  
BASIC MASM  
RUNS DOS AND  
WINDOWS  
EVAL KIT \$295



## \$95 SINGLE PIECE PRICE UNIVERSAL PROGRAMMER

- DOES 8 MEG EPROMS
- CMOS, EE, FLASH, NVRAM
- EASIER TO USE THAN MOST
- POWERFUL SCRIPT ABILITY
- MICROCONT. ADAPTERS
- PLCC, MINI-DIP ADAPTERS
- SUPER FAST ALGORITHMS



OTHER PRODUCTS:

8088 SINGLE BOARD COMPUTER	.....	OEM \$27	• 95
PC FLASH/ROM DISKS (128K-16M)	.....	21	75
16 BIT 16 CHAN ADC-DAC CARD	.....	55	195
WATCHDOG (REBOOTS PC ON HANGUP)	.....	27	95

\*EVAL KITS INCLUDE MANUAL  
BRACKET AND SOFTWARE.  
5 YR. LIMITED WARRANTY  
FREE SHIPPING  
HRS: MON-FRI 10AM-6PM EST

**MVS** MVS BOX 850  
MERRIMACK, NH  
(508) 792 9507



# Home Automation & Building Control

**48**

**Innovations in Home Automation  
& Building Control**

**51**

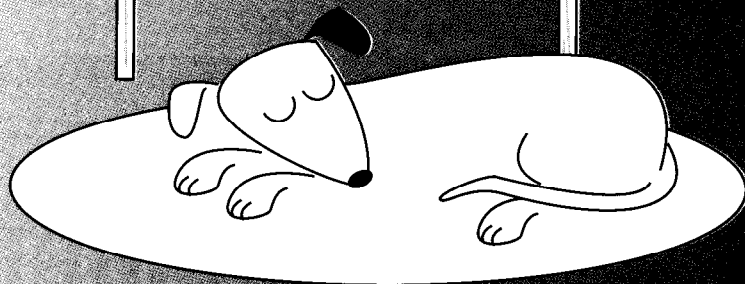
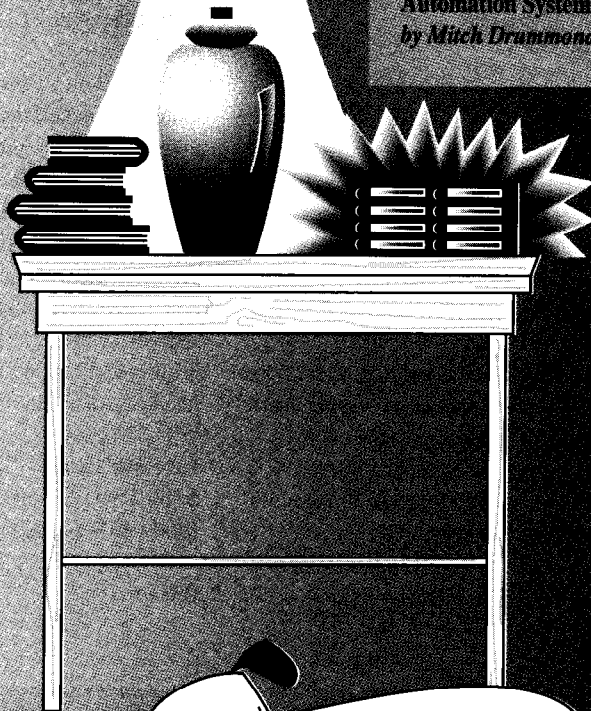
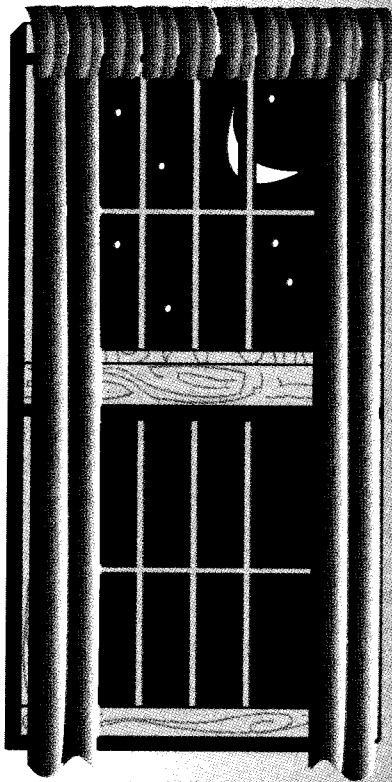
**A Home Control System Based on Fuzzy Logic**  
*by Walter Banks, Ashok Patel & Sherif Abdel-Kader*

**59**

**CAL: Part of the Solution**  
*by Grayson Evans*

**69**

**The Display-3**  
**Add a Human Interface to Your Home  
Automation System**  
*by Mueh Drummond*



# Inno Vations

## IN HOME AUTOMATION & BUILDING CONTROL

edited by  
Harv Weiner

residential electronics.

The CENode Points family provides two classes of components: Connection Points and Application Points. Connection Points offer a complete CEBus-compliant network interface for power line (PL) or radio frequency (RF) communications. Connection Points ensure that OEM products are interoperable with any other manufacturer's CEBus-compliant products. They also provide a standard microprocessor interface to the CEBus network and consistency between different kinds of communications media such as PL and RF.

Application Points offer complete CEBus-compliant application implementations for simple sense and control functions. Users only need to provide their own application circuitry and packaging to produce a complete CEBus-compliant product.

CENode Connection Points for PL applications are available for \$20 in OEM volumes. CENode Connections Points for RF applications will be available in the third quarter of 1995.

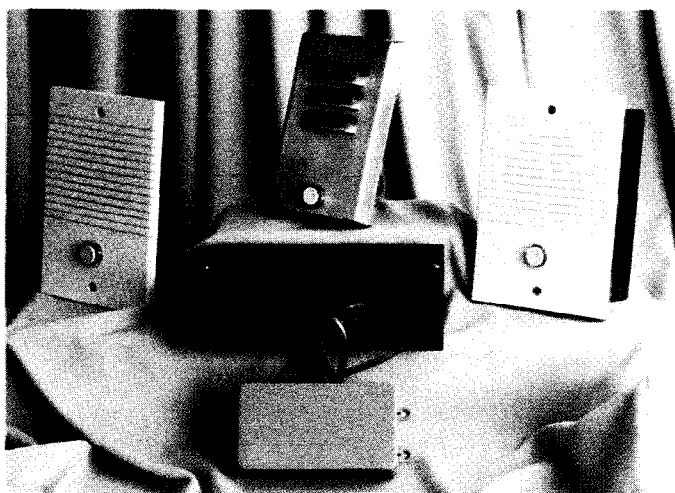
### SIMPLIFIED WIRELESS CONTROL NETWORKS

The promise of intelligent control networks for home and commercial automation took a great leap forward when Intellon unveiled CENode Points, a new family of network components that lets manufacturers easily embed wireless, distributed-control capabilities within their products.

Based on CEBus, the industry-standard communications protocol for home networks, the CENode Points family provides a set of CEBus-compliant building blocks that can be easily integrated into home appliances, lighting, security systems, utility meters, heating and cooling systems, phone systems, and other

**Intellon Corp.**  
5100 West Silver Springs Blvd. • Ocala, FL 34482  
(904) 237-7416, Ext. 204 • Fax: (904) 237-7616

#511



### TELEPHONE SECURITY INTERCOM

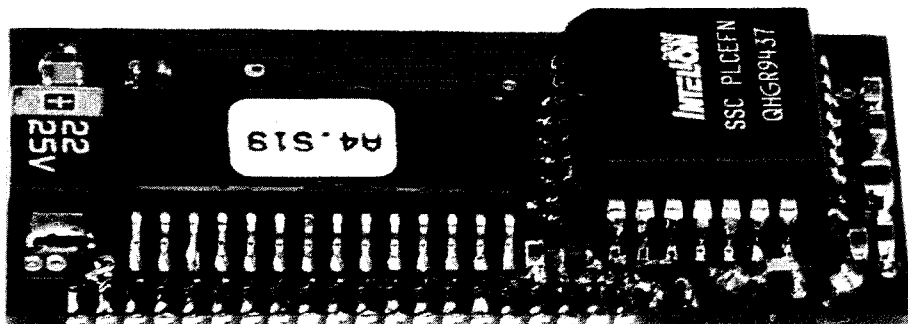
The Tu-Dor Doorman is an intercom which enables a doorbell to be answered from any telephone. The system is compatible with commercial or single-line phone systems and does not require a dedicated trunk port to operate.

The Doorman requires the installation of a bell or buzzer. When someone activates the bell by pushing the door button, the occupant picks up the phone and is automatically connected to the door station. The system is compatible with cordless phones and is particularly useful when employees and security people are working alone and need flexibility in responding to doors.

Door stations are available in plastic or stainless steel. The system can be upgraded to include remote door-strike activation. Prices start at less than \$200.

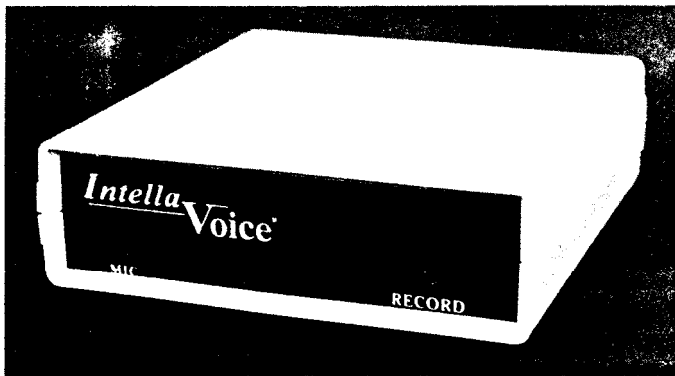
**Solutions Worldwide**  
P.O. Box 8110-812  
Blaine, WA 98231  
(604) 582-4806  
Fax: (604) 582-4386

#510



## X-10 SPEAKS

**IntellaVoice**, a stand-alone X-10 message controller from Intella-Home, provides voice announcement for X-10-based home automation systems. Owners customize their own announcements to accompany key X-10 events. The voice-to-EEPROM storage retains messages even without power. Its built-in microphone and automatic-gain circuits produce high-quality natural voice reproduction. Up to eight, nearly 10-s announcements can be recorded. The onboard processor monitors the power line using the X-10 TW523 interface for key event commands, then plays the correct announcement for that event.



Intella-Home, a security and home automation installer for over four years, has been using its IntellaVoice units exclusively in private installations up to now. Announcements can be heard through the built-in speaker or sent to whole-house audio systems through the supplied auxiliary output jack. Owners of automated homes can enjoy a personal wake-up message while the coffee is brewing or a bedside warning when a yard motion detector is triggered. IntellaVoice can even announce hidden automation activity like "lowering air conditioning" or "water heater's off." The unit comes preprogrammed to accept voice messages without extensive user setup and may be recorded thousands of times without loss of quality.

IntellaVoice sells for \$379.

**Intella-Home, Inc.**  
P.O. Box 780392 • Sebastian, FL 32958  
(407) 589-0970

#512

# InnoVations

## IN HOME AUTOMATION & BUILDING CONTROL

### 3-V TEMPERATURE SENSOR

Analog Devices introduces a 3-V temperature sensor IC with a voltage output guaranteed to exhibit accuracy better than 2°C and nonlinearity better than 0.5% over its full 0-100°C temperature range. The **AD22103** offers high gain and direct connection to standard analog-to-digital converters and requires no additional signal-conditioning circuitry or linearity compensation.

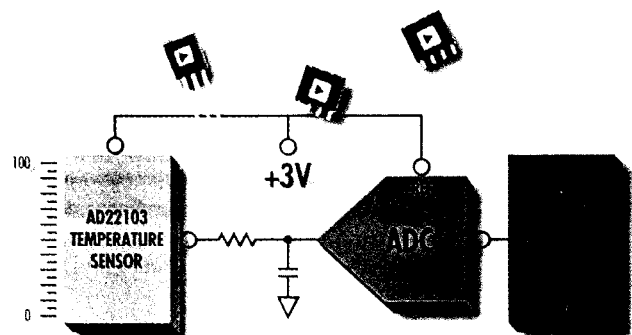
The AD22103 is ratio-metric, providing sustained precision operation as battery voltage levels decrease. The IC is housed in small TO-92 or SO-8 packages and is well-suited for temperature-monitoring solutions in computers, cellular and cordless telephones, as well as portable and low-power electronic equipment.

The AD22103 is equipped with on-chip linearization and signal conditioning. These features eliminate external signal-conditioning circuitry and reduce sensor-system development costs. Since the sensor dissipates very little power, it contributes no appreciable system heat, eliminating the self-heating errors associated with other electronic temperature sensors.

The AD22103 sells for \$0.98 in 1000-piece quantities.

**Analog Devices**  
One Technology Way  
Norwood, MA 02062-9106  
(617) 937-1428  
Fax: (617) 821-4273

#513



# T

his project started as an exercise in software portability and fuzzy logic. After working on microcontroller projects for years, we wanted to look

at some of the issues in porting software between execution platforms.

We chose a fuzzy-logic-based home environment control system because it's an application that is well understood and offers many opportunities for experimentation. As well, we can put a fuzzy-logic tool that we've been using through the paces. The tool extends C to linguistic variables as well as conventional 8- and 16-bit integers.

## FUZZY LOGIC

Fuzzy logic adds the concept of linguistic variables to the variable types most people are familiar with.

Linguistic variables describe data in application-specific terms. We can say, for example, that the day is hot. Further, we can qualify the linguistic term by scaling it between fuzzy 0 and fuzzy 1 (usually represented by 0 and 255). A hot day might have a crisp value of 100°F and a linguistic value of fuzzy 1. At 60°F, it is barely hot (perhaps fuzzy 0.2). At 40°F or so, it is not hot at all.

Our ability to naturally describe our problems in linguistic terms makes fuzzy logic easy to use. A fuzzy rule might be:

**IF room IS hot THEN  
ac is on-hi gh**

This rule refers to two linguistic variables **hot** and **on-hi gh**. It is interesting that linguistic variables are context sensitive.

# A Home Control System Based on Fuzzy Logic

It has been a few years since interest in fuzzy-logic technology for embedded applications started. And, it is surprising that computing has taken so long to accept fuzzy-logic technology. For the most part, fuzzy logic presents a cleaner, clearer interface to the real world—a way of expressing ourselves in terms of technology's application, and not in terms of the fundamental science on which the technology is based.

One problem with finding suitable applications comes from attempts to implement applications in fuzzy logic based on the underlying science of an application. We were competing with implementations of the science describing a real-world application, rather than two implementations of an actual application.

This small difference is significant. Most scientific application models are linear representations of the application. Implementations with fuzzy logic at best equal alternative implementations. On the other hand, many contain some nonlinear aspects, which fuzzy logic easily handles.

For example, consider a simple thermostat control system for a living room. To control the temperature, we start with two decisions: "when the room is cold, turn on

WALTER BANKS, **ASHOK PATEL**  
& **SHERIF ABDEL-KADER**

After reminding us of the basics of fuzzy logic, Walter and his associates concentrate on the needs of environmental control. They specifically look at some of the implementation quirks that come with writing fuzzy logic code for a home-control system.

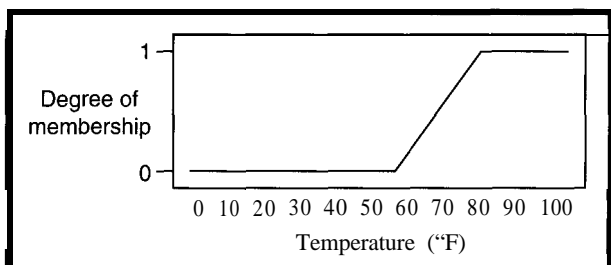
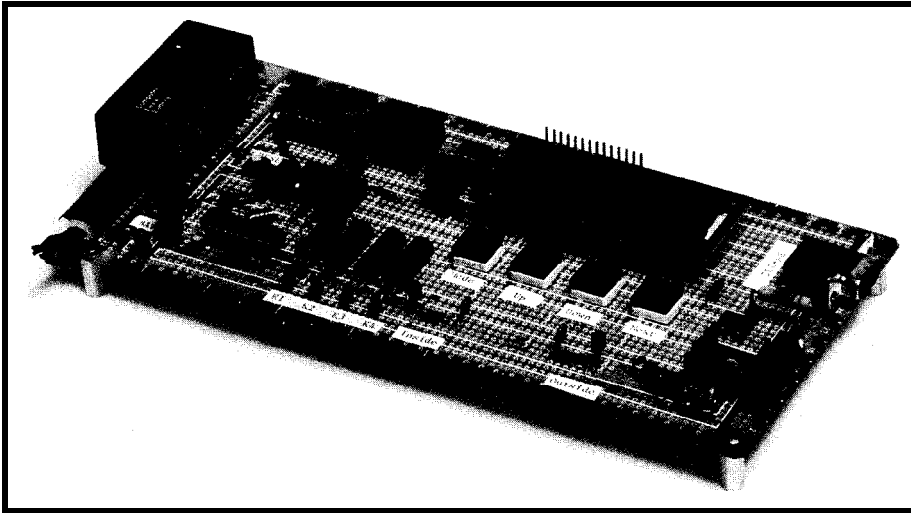


Figure 1: This graph compares room temperature with degree of membership to determine the linguistic variable **HOT**.





**Photo 1:** There is nothing critical about either component layout or wiring on this fuzzy-logic thermostat prototype board. Clock speeds are low and the analog signals (0-5 V) are significantly above the noise levels on the board.

the heat” or “when the room is too hot, turn off the heat.”

We express these ideas formally to the computer with the following rules:

```
IF room IS Cold THEN heat IS on
IF room IS Hot THEN heat IS off
```

Now, suppose we move to Arizona, where there are cold and hot temperatures. To cope with the heat, we add rules to our system:

```
IF the room IS Cold THEN
  airConditioner IS off;
IF the room IS Hot THEN
  airConditioner IS on;
```

The control system is implemented so that it gathers a collection of independent thoughts about a problem presuming that each idea is part of the total solution.

This all sounds good, but will it work? We’ve introduced the concept of linguistic variables that have some meaning in the real world. Let’s look at the linguistic variable, “room IS Hot.”

room really indicates room temperature, so let’s build a scale of temperature on the x-axis. On the y-axis, we normalize our conclusions between 0 and 1. (Boolean logic has only two values 0 and 1, while fuzzy logic has all the analog values of 0-1.)

We begin with 100°F, which we all agree is hot. We set that value to 1. We also agree that 90°F is still hot, so its value is 1 as well. At 70°F, we begin to have differing opinions. Some people say 70°F is not really hot, while others still insist it is. To account

for differing opinions, we set the value at 70” to 0.5. Finally, when the temperature drops to 60°F, we all agree it’s no longer hot. We set the value there to 0. Figure 1 graphically depicts this change in membership.

Now that we’ve defined the linguistic term **HOT** for our computer, it’s simple to write a subroutine. On a scale between 0 and 1, the code needs to return how relevant **HOT**

is given a temperature value. For more granularity, increase the range of possible return values from, for example, 0 to 255.

The code only requires a few instructions to implement on most computers, even the simple ones used in household appliances. **HOT**, in this example, describes a condition in the room (i.e., it is a member of room).

The code fragment in Listing 1 passed through Fuzz-C, a software preprocessor. The linguistic variable **HOT** is defined as a trapezoid (i.e., it has four arguments). The arguments are formed from the crisp intersection for each corner of the trapezoid.

The Fuzz-C preprocessor generates C code from this definition (see Listing 2). As you can see, it does two things. First, it declares **room** as type **char** (its crisp value), and second, it generates a function which takes the room temperature as an argument and returns a value between fuzzy 0 and fuzzy 1, indicating the relevance of linguistic variable **HOT**.

Even though Listing 2 shows just one linguistic variable, in this application, the crisp variable **room**

**Listing 1:** The linguistic variable **HOT** is defined as having a trapezoidal-shaped membership function. The definition corresponds to the graph.

```
LINGUISTIC room TYPE char MIN 0 MAX 150

MEMBER hot { 60, 90, 150, 150 }
```

**Listing 2:** This code translates the linguistic variable from a definition to an executable function. The function can be easily implemented on most small embedded microcontrollers.

```
char room;

char room_hot (char __CRISP) {
  if (__CRISP < 60)
    return(0);
  else {
    if (__CRISP <= 90)
      return(((__CRISP - 60) * 8) + 7);
    else {
      return(255);
    }
  }
}
```

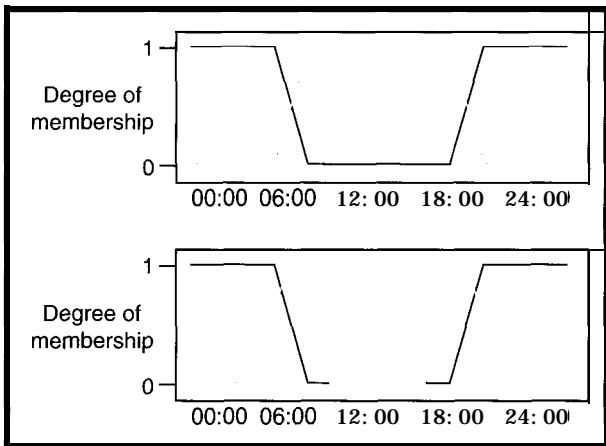


Figure 2: *The first graph compares 24-hour time and linguistic variables day (green) and night (red). The lower graph compares 24-hour time and linguistic variables morning, evening, and night setback (red).*

n i g h t's degree of memberships is always fuzzy-one.

The linguistic variables **morn i n g** and **even i n g** are declared with trapezoids in the same manner, except with the appropriate time periods.

n i g h t s b, the last linguistic variable, also uses a fuzzy expression. It has been declared as n i g h t not including even i n g. Similarly, **days b** is also a fuzzy expression stating the day setback temperature time is day not including **morn i n g** and **even i n g**.

Fuzz-C enables you to use variables instead of constants to declare linguistic variables. Using fuzzy expressions to generate some of the linguistic variables results in a few fixed variables which define several linguistic variables.

Generating code to produce degrees of membership is not difficult—either by hand or with some development aid. Figure 2 and Listing 3 offer a full definition of the linguistic variables associated with time on the thermostat project.

#### HARDWARE

The fuzzy logic thermostat prototype (see Figure 3) was implemented around a

may have many linguistic members: **HOT**, **COLD**, and **NORMAL**.

Somewhat more interesting are the linguistic variable's declarations related to time of day. With the goal of referring to time in terms we use, divide the day into 240 parts. In a 24-h day, 6-minute resolution nicely fits into an X-bit variable while still providing the resolution necessary to deal with control problems.

In our application, there are six linguistic variables in a 24-h period:

**day**, **n i g h t**, **morn i n g**, **even i n g**, **daysb** (day setback), and **n i g h t s b** (night setback). It is easy to add more linguistic variables or to change the way they are defined or used.

**day** is defined as a trapezoid starting at 5:30 A.M. By 6:30 A.M., it is fully day and continues to be so until 5:30 P.M. when it starts to taper off. By 6:30 P.M., it is no longer day.

**n i g h t** is defined as NOT **day**. **n i g h t** therefore is a complement of **day** and the sum of **day** and



# HCS II Home Control System

**CIRCUIT CELLAR, INC.**  
4 Park Street, Suite 12 • Vernon, CT 06066  
Tel: (203) 875-2751 • Fax: (203) 872-2204

- Energy Management
- Security and Alarm
- Coordinated Home Theater
- Coordinated Lighting
- Monitoring and Data Collection

Get all these capabilities and more with the Circuit Cellar HCS II. Call, write, or FAX us for a brochure. Available assembled or as a kit.

EXTENDED SYST

HCS II BASIC SYST

PL-Link

TW523

PS12-1

Buffer/Terminator

L/O Board

Module

Microchip PIC 16C74. The 16C74 was used for the prototype because it met all the requirements of our project: several analog inputs, low-power capability, and more ROM space than we expected to need even for performance-data gathering.

The design was expected to use a 32-kHz crystal for the processor clock. While building the prototype, we decided it would be useful to add a serial port so a PC could monitor the execution of the fuzzy-logic controller. The serial port meant we needed a faster processor clock so we could keep up with the serial data port. During development, we used a 4-MHz crystal.

Household thermostats have standard wiring in most homes. The heating, air conditioning, and fan systems are controlled by contact closures to a 24-VAC supply. The humidifier relay is a contact closure brought out to two terminals. Although it appears that some form of triac-based electronic switch could be easily used, one of us had an experience where a new high-efficiency system would not operate with a triac switch. Instead, we used a relay that is compatible with a broad spectrum of heating systems.

The 24-VAC supply lines to the thermostat power the fuzzy-logic thermostat controller. The 24-VAC supply passes through a bridge rectifier, and a series resistor limits the power dissipation of the 7805 5-V regulator. We provided a simple power-fail detection circuit to allow the software to turn off the heating, air conditioning, and LCD display.

During a power failure, the software needs to maintain only the date and time. In the prototype, we divided one of the ports into two 4-bit portions. Four of the bits control the output relays for heating, air conditioning, fan, and humidifier. The remaining four bits sense key presses.

A 4-key keypad makes contact closure to ground with 10-k pull-up resistors. The keys are labeled Mode, Up, Down, and Next. Software filters the key bounce.

The temperature sensors are Analog Devices AD22100s. These three terminal sensors are easy to use—just connect between  $V_{CC}$  and ground. We used the low-pass filter (1 k followed by 0.1  $\mu$ F) recommend by Analog Devices.

The NVRAM stores all of the user-definable parameters such as room temperature setpoints and day and night setbacks. The NVRAM is a serial memory part with data and address information passed serially

**Listing 3:** *The linguistic variables associated with the crisp variable hours define common references to the time of day. The shape of the membership functions allow smooth transitions between adjacent or overlapping time periods.*

```
LINGUISTIC hours TYPE char MIN 0 MAX 240
// hours; .1 hour 0 240 for a day
{
  MEMBER day      { 55 , 65 , 175 , 185 }
  MEMBER night    { FUZZY { hours IS NOT day } }
  MEMBER morning  { 50 , 60 , 80 , 90 }
  MEMBER evening  { 160 , 170 , 190 , 200 }
  MEMBER nightsb { FUZZY { hours IS night AND hours IS NOT
    evening } }
  MEMBER daysb    ( FUZZY { hours IS day AND hours IS NOT
    evening AND hours IS NOT morning } )
}
```

**Listing 4:** *This routine converts the ADC readings from the Analog Devices AD22100 sensor into actual temperature readings.*

```
#define F_const 400
#define F_offset -78*256

#define C_const 222
#define C_offset -61*256

char convert-temp (char temp){
  return (((long)temp * F_const) + F_offset) >> 8);
}
```

so it can be implemented in an 8-pin part. This device is commonly used in consumer and industrial devices. The NVRAM code is another example of reusable driver code that originally came from a modem application using the Zilog Z8.

## DISPLAY DESIGN

The two-line, 16-character display shows current room temperature, humidity, and outdoor temperature. The display is a standard LCD driven by a 4- or 8-line interface. Since both interfaces are available, only interface speed and available processor I/O lines need be considered.

The display's software interface has been well documented in various publications (INK 8). The LCD display drivers were initially written in C for a barcode reader about 5 years ago. We simply recompiled the C source for the PIC16C74 (we've used this same source code with several microcontrollers).

## TEMPERATURE CONVERSION

As we mentioned before, we chose the Analog Devices AD22100 for indoor and outdoor

temperature sensing. Three-terminal temperature sensors have made temperature measurements by microcontroller-based systems very easy. The sensor is designed for automotive applications that normally experience wide temperature swings.

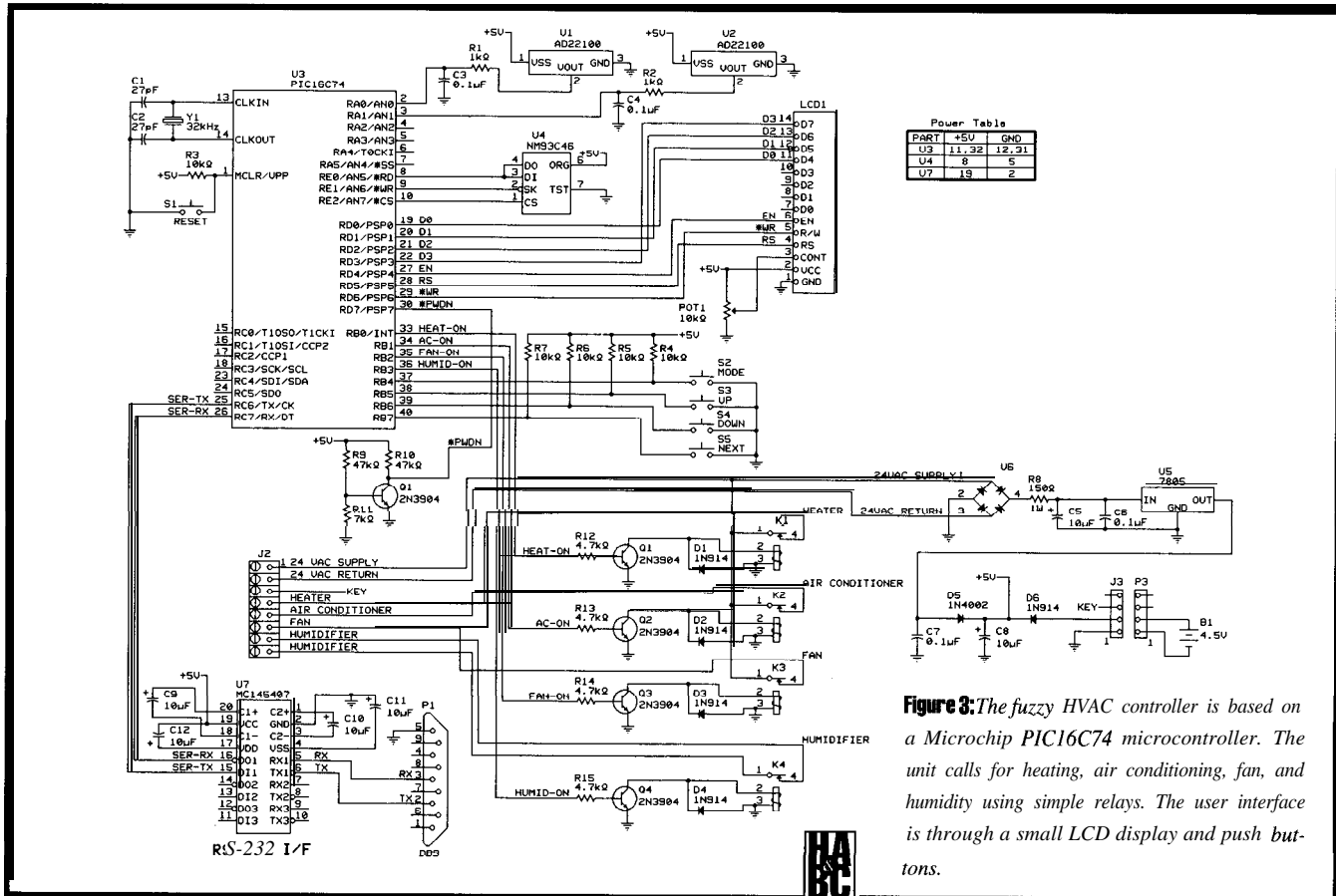
Unlike the thermistors typically used for temperature measurement, this sensor delivers a very linear 22.5 mV per  $^{\circ}$ C (other equally suitable solid-state sensors of this type are offered by other silicon vendors).

This temperature sensor requires  $V_{CC}$  and ground and produces an analog signal in the range of 0–5 V. The working range is actually 0.25–4.75 V for a temperature range of –50–+150 $^{\circ}$ C. Although this range is a little coarse for precision, it's acceptable for a home thermostat.

The 16C74 we employed for the prototype has eight 8-bit A/D converters each with a range of 0–5 V. If precision is needed, it is possible to implement the ADC inputs with expanded scales only to cover a limited range, but with 8-bit accuracy







**Figure 3:** The fuzzy HVAC controller is based on a Microchip PIC16C74 microcontroller. The unit calls for heating, air conditioning, fan, and humidity using simple relays. The user interface is through a small LCD display and push buttons.

Call 24hrs for FREE 64 page Color Catalog

**HOME AUTOMATION SYSTEMS, INC.**

Largest Selection of HOME AUTOMATION products in the World!

Browse us on the WEB! <http://www.techrtail.com/smarthome>

Over 500 unique products!

Hundreds of hard-to-find automation, X10, and wireless control products. Computer control of your home, security systems, surveillance cameras, infra red audio/video control, HVAC, pet care automation, wiring supplies, and much, much more!

**HOME AUTOMATION SYSTEMS, INC.**  
151 Kalmus Drive, Ste L4, Costa Mesa, CA 92626  
Questions 714-708-0610 FAX 714-708-0614

Call **800-SMART-HOME**  
800-762-7846

**FREE**

## Microcontroller Networks (µLANs)

With Cimetrics' 9-Bit µLAN you can link together up to 250 of the most popular 8- and 16-bit microcontrollers (8051, 80C196, 80C186EB/EC, 68HC11, 68HC16, PIC16C74).

**The 9-Bit µLAN is:**

- **Fast**—A high speed (62.5k baud) multidrop master/slave RS-485 network
- **Flexible**—compatible with your microcontrollers
- **Reliable**—robust 16-bit CRC and sequence number error checking
- **Efficient**—low microcontroller resource requirements (uses your chip's built-in serial port)
- **Friendly**—Simple to use C and assembly language software libraries, with demonstration programs
- **Complete**—includes network software, network monitor and RS-485 hardware
- **Practical**—applications include data acquisition and distributed control

**9-Bit RS-485 Multidrop Network**

- Microcontroller or PC Master
- Intel 8051 HVAC/R Lighting Control
- Intel 80C186EB/EC Point-of-Sale
- Motorola 68HC11 Process Control Transportation
- Microchip PIC16C74 Fire Detection Security

**Cimetrics TECHNOLOGY**  
Cimetrics Technology • 55 Temple Place • Boston, MA 02111-1300. Ph 617.350.7550 • Fx 617.350.7552

We translated the A/D converter value to actual temperature by multiplying by a constant and adding a zero-count offset. Since we know the environment's working range, we don't need range checks on the data. In Listing 4, the code converts the ADC result to Fahrenheit. Constants for Celsius are also given.

## PUTTING IT ALL TOGETHER

We now have all the pieces we need—displays, keypad, temperature sensors, reed-relay outputs, and definitions for time and comfort. Although we define our comfort in absolute temperature terms, it is easier to use a conventional thermostat setpoint to set the desired temperature and let the linguistic functions of comfort (COLD, NORMAL, and HOT) be based on the relative temperature error. Figure 4 shows the temperature comfort of the room. A value of 0 means the room's temperature is at the setpoint.

As we established earlier, fuzzy rules are a series of individual statements about the problem. Listing 5's partial source of the fuzzy-control block shows some of the rules used in this project.

The first few rules are obvious. We then move to the energy-saving rules. Here, given information we have about inside and outside temperatures, we make choices based on natural warming or cooling.

With multiple rules, we have to decide how we want the computer to evaluate an outcome. Typically, it would evaluate each rule independently and weigh each rule by the strength of its logical argument.

Another method involves *consequence functions*. These functions are fuzzy logic's way of resolving conflicts in the decision-making process and provide what is referred to as *defuzzification*. Fuzzy literature details many ways of doing this, each author extolling the virtues of a preferred method.

We used one of the standard methods called *center of gravity* for our consequence functions. In general; this method sums the results and the weight (degree of membership) of each input, and computes a single answer to accurately reflect all of the results. In effect, center of gravity provides smooth transitions between competing terms.

**Listing 5:** *This jizzzy control function regulates the heating and air conditioning from fuzzy rules. Fuzzy rules are independent intuitive control functions. The consequence functions resolve the actual control settings for heat and air conditioning.*

```

FUZZY room_control {
  IF room IS cold THEN
    ac IS OFF
    heat IS ON

  IF room IS normal THEN
    ac IS OFF
    heat IS OFF

  IF room IS hot THEN
    ac IS ON
    heat IS OFF

  IF room IS cold AND OutsideTemp IS hot THEN
    heat IS OFF

  IF room IS hot AND OutsideTemp IS cold THEN
    ac IS OFF

```

This project has simple on and off actions even though the control calculations are done for continuous control. Consequence functions still are able to resolve different control schemes with varied inputs (see Listing 6).

All of this is tied together with a *main* function shown in Listing 7. This function is conventional controller code which reads the indoor and outdoor temperature, calls the fuzzy-control function, then calls the LCD information display routine. The day and night setback routines are an unusual use of a linguistic variable that gives a smooth temperature transition throughout the day. The define for *NightSetback* shows how linguistic functions may be called individually.

## PROTOTYPE

We built the prototype system on a small development board (see Photo I). The

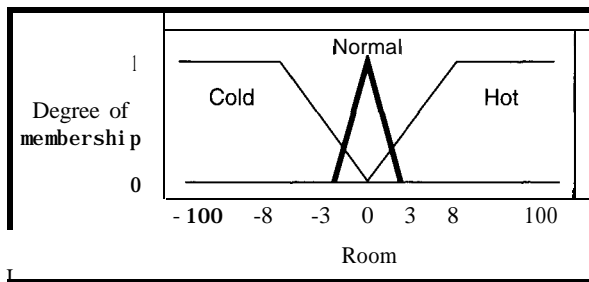
photo shows the noncritical nature of the construction of the project. All of the analog signals are noncritical in the 0-5-V range. Layout of the digital signals is noncritical due to the low clock speeds used in this project.

We made a number of small changes to the design to accommodate the developmental nature of the project. First, we added a serial port to the processor so we could monitor the internal state of the thermostat. To achieve acceptable serial data rates, we increased the processor's clock speed from 32 kHz to 4.00 MHz. While this has little effect on the prototype, it increases battery drain when conventional power fails.

In hindsight, the serial port could be useful. As well as fulfilling a monitoring function, it could set the thermostat's operating parameters.

## PROJECT PORTABILITY

Within the last few years, most manufacturers have introduced versions of their microcontrollers—Microchip PIC 16C74 (what we used), Zilog Z8, National COP8, and Motorola MC68HC05 and MC68HC08—which meet this project's requirements. We didn't discount the 8051 families, but were more familiar with software tools for the rest of the chips mentioned.



**Figure 4:** You can combine linguistic terms to provide for room variables as a function of temperature errors: normal (bold), cold, and hot.



We recompiled the source code for the project on each of the above microcontroller platforms. In each case, the necessary changes to the source code were limited to replacing the processor descriptor header files and making minor changes to port and bit assignments.

The most significant change is the A/D conversion method used on each of the processors. Most of the embedded B-bit processors have at least one family member with onboard A/D conversion capability. Realistically, you could produce a demonstration board with several processor sockets-ne for each supported processor.

## FUZZ-C

We used the Fuzz-C preprocessor to add linguistic variables to our C program. This inexpensive preprocessor works with almost any C compiler. Fuzz-C is actually implemented as a simple compiler: it reads in a mixture of C and fuzzy-logic declarations and functions and reproduces the C unchanged. The fuzzy logic is translated into its C equivalent.

Like most fuzzy-logic projects, this one contains a little fuzzy-logic-related code (linguistic variables, consequence functions, and fuzzy-logic control functions). It also has a lot of traditional code required to scan keyboards, drive displays, control relays, and send and receive serial characters.

Fuzzy logic is starting to be seen as a tool that can add significantly to a developer's ability to describe a problem. Fuzzy-logic techniques enable developers to deal with applications that are not well defined in the crisp sense or have many different operating parameters, some of which may be nonlinear.

The files posted on Circuit Cellar BBS contain both the source code for the project and the intermediate C code and its compiled form. You can easily retarget the intermediate C code for other platforms.

## FUTURE WORK

It's reasonable to tie the thermostat to a central home controller bus such as that used by the HCS II. You could then centrally control and monitor an HVAC system and room environments.

All engineering projects fall victim to three fundamental shortages: lack of information, time, and resources. Of the three, the last is the easiest to solve and the first, impossible.

**Listing 6:** Here's the consequence function for heat. The defuzzification method selected is center of gravity, which means each call to the consequence function from the fuzzy rules is weighted. After all of the rules have been executed, the consequence function returns a fuzzy result. The action statement turns the heat on if the result is greater or equal to 128.

```
CONSEQUENCE heat TYPE char MIN 0 MAX 255 OEFUZZ cg
ACTION { Heat_On = heat >= 128;}

MEMBER OFF { 0 }
MEMBER ON { 255 }
```

**Listing 7:** The complete mainline for the fuzzy home environment controller ends up being very simple. Most of the software in this system is ordinary C for scanning keyboards, updating time, driving displays. A little of the code is dedicated to linguistic variables and fuzzy functions. NightSetback shows another way that the linguistic variable nightsb may be used to get smooth transitions between different time periods. The fuzzy control function room\_control() is called as a C subroutine.

```
#define NightSetback(((nightsb(hours) * NightOffset) / 256)
void main (void)
{
    Init_all0;
    while(1) {
        if (keypressed)
            Service_mode;
        else {
            InsideTemp = convert_temp ReadAD(Inside));
            OutsideTemp = convert_temp ReadAD(Outside));
            room = InsideTemp - Room_SP - NightSetback() - DaySetBack();
            room_control0;
            Display();
        }
    }
}
```

Probing the problem illuminates the science involved, but to achieve results, we must still accomplish the engineering. Fuzzy-logic-based systems address both science and engineering with a single brilliant idea: "Why not describe the final solution to the problem intuitively?" Engineering and science then become side effects of the solution rather than the other way around.

*Walter Banks is the President of Byte Craft, a company specializing in code creation tools for embedded microcontrollers. He may be reached at walter@bytecraft.com.*

*Ashok Patel is the president of Softart Microsystems where he does hardware and software designs using microcontrollers for the communications industry.*

*Sherif Abdel-Kader is a software support specialist Of Byte Craft. He is a graduate from the*

*University of Cairo has extensive experience in both embedded systems development and customer support.*

## SOURCE

MPC compiler for PIC 16C74,  
Fuzz-C  
Byte Craft Limited  
421 King St. N.  
Waterloo, Ontario  
Canada N2J 4E4  
(5 19) 888-6911  
Fax: (5 19) 746-675 1  
info@bytecraft.com

## I R S

413 Very Useful  
414 Moderately Useful  
415 Not Useful



# T

he January 1980 cover of *BYTE* is Robert Tinney's classic painting of a futuristic home computer in an elegantly carved wooden cabinet. A

hand-held RF remote-control device lies next to it on the table. The computer's screen reads, "MADAM: DINNER IS SERVED." The issue's theme is domesticated computers and the lead article is Steve Ciarcia's "Computerize a Home."

The article describes an interface between a home computer and a new gadget on the market, a BSR (X-10) power-line carrier controller and light switch. For many people, including myself, who struggled in the early '80s to make domesticated computers into a business, this article represents the beginning of what we now call "home automation."

On page 91 of the same issue is a boring, black and white ad by a small company called Microsoft that promotes their only product, a simple BASIC compiler (yawn).

A lot has happened since 1980. Microsoft and home computers are now big business. Consumers understand the benefit of owning a computer and have a wealth of hardware and software that performs truly beneficial, educational, and entertaining tasks.

# CAL: Part of the Solution

Home automation technology has certainly improved...some. There is more press about home automation—more gadgets, more modules, more stuff—but, unfortunately, home automation remains in its infancy. It is still more of a concept than a product, still the domain of the technically competent, and still struggling to be a business.

How come?

The answer is simple. Home automation has not made the transition that home computers did in the late '80s and early '90s. It lacks a standardized bus, standardized OS, and usable applications. Home automation must make that same transition or it will remain simply a curiosity.

The transition requires three changes:

GRAY SON EVANS

Need a better **feel** for what's required in programming a CEBus application? If so, check out Grayson's article on CAL, the Common Application language. As a key member of the CEBus committee, **Grayson** brings us an inside corner on the highest level of CEBus specification.

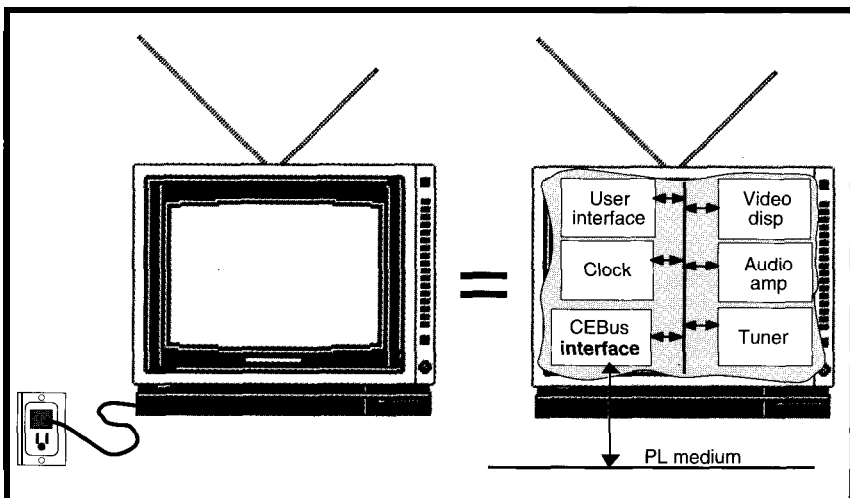


Figure 1: Any CEBus-compliant product consists of a series of contexts at a network node address.



- a standardized physical interface between devices in the home so information can be easily and reliably exchanged
- a standardized way for devices to interoperate and talk to each other using a common language
- a wealth of truly beneficial applications (i.e., things for devices to do which are available to the consumer at reasonable cost)

Aware of the problem, the CEBus Committee, sponsored by the Electronic Industries Association (EIA) and made up of consumer product manufacturers (including home computer manufacturers), worked to solve the first two problems with the development of EIA-600. The CEBus standard details the physical interface, message protocol, and language necessary to allow the third item (consumer applications) to be developed.

This article takes a closer look at the most important aspect of CEBus—its interoperable language called CAL.

## CAL

The CEBus Common Application Language (CAL) defines what products say to each other and was specifically developed for residential product control. By establishing a common product model and common set of commands, residential products can communicate with other products without knowing how each specific product operates, who built it, or what's in it.

Unlike C or Pascal, CAL is a command language consisting of two major parts:

- the definition of a data structure that models product operation
- the description and syntax of the messages that operate on the data structure.

CAL adheres to many object-oriented principles typically found in languages such as C++, but it is not an object-oriented language.

<b>General</b>		<b>HVAC</b>	
00	Universal	40	Environmental Zone
02	User Interface	41	Environmental Sensor
04	Data Channel	42	Environmental Status
05	Time	43	Environmental Zone Control
<b>Audio/Video</b>		<b>Utility</b>	
10	Audio Amplifier	50	Utility Metering
11	Medium Transport	51	Utility Monitoring
12	Tuner	56	Energy Control
13	Video Display	57	Energy Management
14	Audio Equalizer		
<b>Lighting</b>		<b>Security</b>	
20	Light Sensor	60	Security Zone/Sensor
21	Lighting	61	Security System
22	Lighting Zone	62	Security Control
24	Lighting Zone Control	63	Security Alarm
<b>Convenience</b>		<b>Appliance</b>	
81	Window control	70	Washer
82	Door/Gate	72	Water heating
84	Pool/Spa	73	Dryer
		74	Refrigerator/Freezer

**Table 1:** Even a partial list of CEBus contexts illustrates that every major electronic subsystem in the house can be addressed using CEBus.

CAL messages originate from and are received and parsed by the CAL interpreter, which is coded into the Application Layer protocol software of all CEBus-compliant products. The CAL interpreter is the heart of any CEBus product, its design requirements set early in its development. The CAL interpreter must:

- be common to all residential devices
- perform both control and data acquisition functions

- perform network administration and management functions
- be reusable and extensible
- be customizable
- be simple enough to code in small microcontrollers

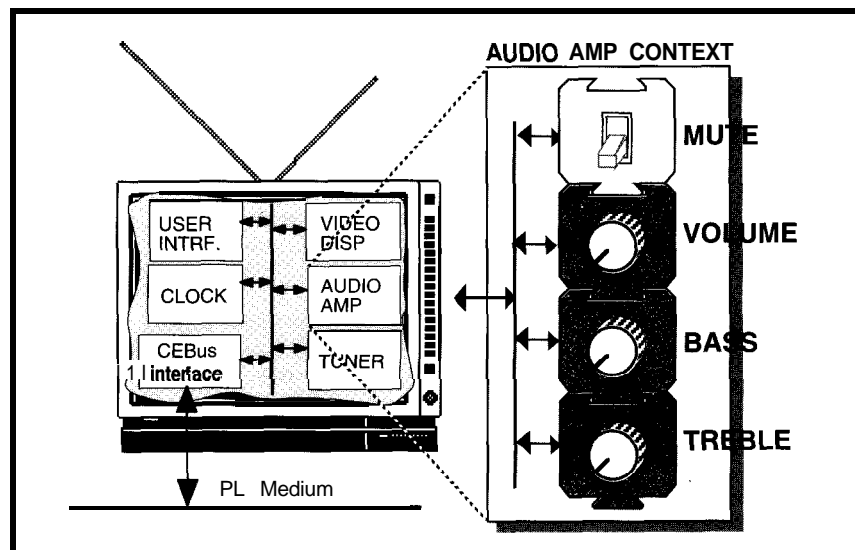
## CAL DATA STRUCTURE

The design of CAL assumes that all electrical appliances and products in the home have a hierarchical structure of common parts and that the operation of the common parts is similar from product to product.

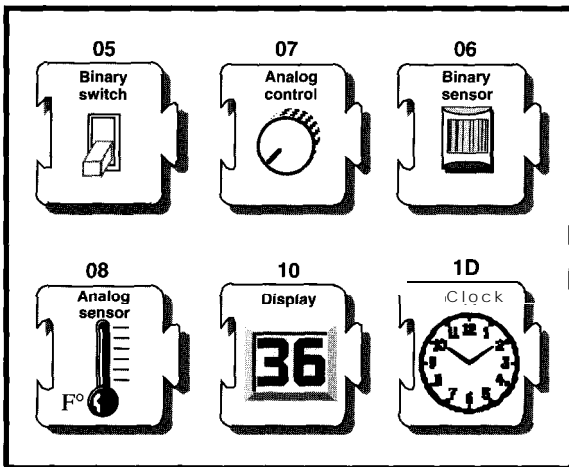
CAL treats each product as a collection of one or more of these common parts called *contexts*. A context defines a functional subunit of a product whose operation can be defined and remains constant regardless of where it's used.

For instance, CAL does not know how a TV operates because a TV is too general a product category. Depending on the TV, it could have a clock, audio amplifier, tuner, and built-in surround sound. However, CAL does know how to operate contexts, and all CEBus devices, including TVs, consist of one or more contexts.

A CEBus TV looks like a collection of contexts at a network node address (see Figure 1). Depending on the model's features of the model, a CEBus TV might contain contexts for a video display, an audio



**Figure 2:** Each context consists of one or more objects. Four of the twenty Audio Amplifier Objects include Mute, Volume, Bass, and Treble.



**Figure 3:** CEBus object classes typically correspond to real-world devices. Objects are plug-and-play in each context.

amplifier, a tuner, a clock, a user interface, and so on.

CAL defines more than 50 different contexts for everything from lighting to security, heating and air conditioning, washing and drying. Table 1 gives a partial context list.

Each context, regardless of what product it's in, operates the same way.

The audio amplifier in the TV, stereo receiver, speaker phone, and intercom all work alike.

If a CEBus product knows how to set the volume in the audio context of one product, it knows how to set the volume in all products.

As shown in Figure 2, each context consists of one or more objects. Each object is a software simulation (or model) of a control function of a context. Only 4 of the 20 objects specified in the actual CEBus audio

context are shown. The volume, bass, and treble analog-control objects, and the mute binary-switch object represent control functions typically found in audio amplifiers.

Objects model tasks performed by users to control a product. To turn a light off or on, use a switch, defined by CEBus as a *binary switch object*. To adjust the stereo receiver's volume or to raise



ID	Object	Type
01	Node Control	
02	Context Control	
03	Data Channel Receiver	I
04	Data Channel Transmitter	I
05	Binary Switch	I/O
06	Binary Sensor	O
07	Analog Control	I/O
08	Analog Sensor	O
09	Multiposition Switch	I/O
0A	Multiposition Sensor	O
0B	Matrix Switch	I/O
0F	Meter	I
10	Display	I
11	Medium Transport	I/O
13	Dialer	I
15	List Memory	I
16	Data Memory	I
17	Motor	I
19	Tuner	I
1A	Tone Generator	I
1C	Counter	O
1D	Clock	O

Table 2: A partial list of the 25 possible CEBus objects shows which ones are input only (I), output only (O), and input and output (I/O).

or lower the thermostat's temperature, use a control (knob), defined as an *analog control object*. All analog controls are similar. They

*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.

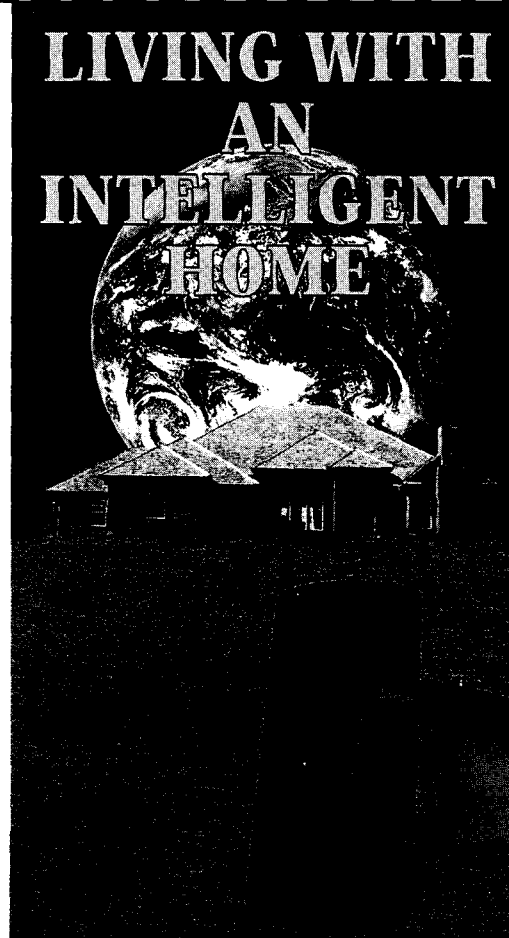
Written by David Gaddis,  
author of *Understanding & Installing Home Systems*

This VHS video cassette retails for \$24.95 plus \$5 s&h.

It is being offered to Circuit Cellar INK subscribers for only \$17.95 plus \$4 s&h (U.S.)

**ORDER TODAY!**

Don't let this exciting 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) 8752752

Fax (203) 872-2204

can be physically set to any value between their minimum and maximum values.

Other controls found on consumer products include multiposition switches (e.g., the switch selecting the audio source on the stereo receiver), keypads, and so on. Several sensor control devices provide information about a product function. Analog sensors (e.g., temperature or light-level sensors) provide the sensor the equivalent of an analog control: they can assume any value between their minimum and maximum values.

All objects are defined by a class number (see Figure 3). An object class defines the generic operation of the object.

When an object is used in a specific context, it assumes a specific instantiation, such as volume or temperature control, of a context's function. Table 2 is a partial list of the 25 predefined objects usable in CAL.

Objects consist of a set of instance variables (IVs). Like variables in any software program, IVs have a length or size and a data type. All network operations on contexts are performed by reading from and writing to object IVs. The IVs defining each object are listed in the object tables of CAL specification. There is a table for each of the 25 objects.

Table 3 shows the analog control object. The object class and name are given in the definition's top line. The description section gives the general use of the object and is followed by a list of all IVs defined for an object class. An IV label is an ASCII string of one or more characters. The label is how the IV is referenced in a message. Any IV label and name in bold type is required to be supported in the object. Other IVs are optional.

The second column shows which IVs are read-only (R) or

(07) Analog control				
Models operation of a continuously variable control such as knob, slider, or analog setting device. Used to set a variable value for a function over a range of values at a defined resolution.				
IV	R/W	Type	Name	IV description
U	R	n	units-of-measure	Units of measure used in context selected
S	R	n	step-size	Smallest increment that control can be changed
r	R	n	step-rate	Rate of change of control. Zero implies instantaneous change
N	R	n	min_value	Minimum value control can be set
M	R	n	max_value	Maximum value control can be set
D	R	n	default_value	Default value control can be set
C	R/W	n	current-value	The current value of control
P	R	n	previous_value	Used in reporting to track the last value of C
R	R/W	d	reporting_condition	Test condition used in reporting
H	R/W	d	report-header	Report message body
A	R/W	d	report-address	Node address of report message

Table 3: The CEBus object table for the analog control object (07) offers a quick look up.

read/write (R/W) as referenced from the network and not internally to the product.

The third column shows the data type of the IV where *b* is Boolean, *n* is numeric (integer or real), *c* is a character string, and *d* is binary data.

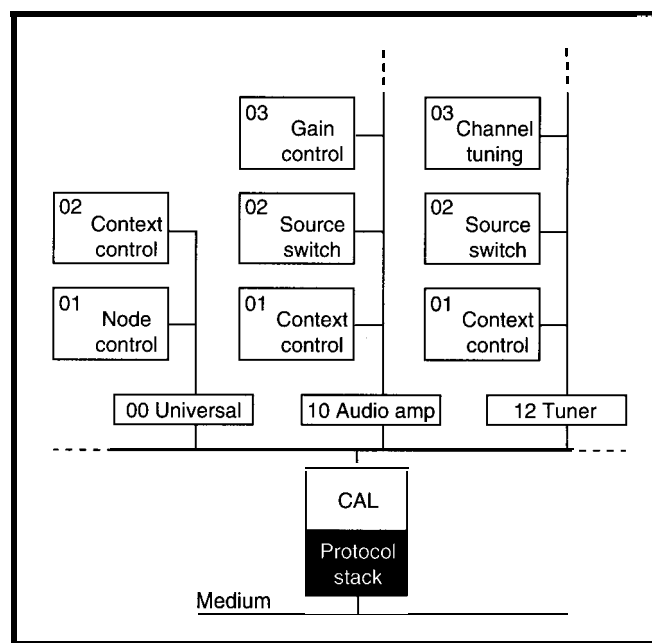


Figure 4: The tree structure of the context's model in a CEBus product hierarchically offers specific controls.

Generally, read-only IVs provide information for other nodes on the network about the application of the object. For example, the `units_of-measure`, `step-size`, `step-rate`, `min_value`, `max_value`, and `default_value` IVs in Table 3 define the characteristics of an object's particular use. None of these IVs are required, but they may be of benefit to other nodes.

### CONTEXT/OBJECT DATA STRUCTURE

As Figure 4 illustrates, the context and object data structure in a product can be thought of as a tree structure. Incoming messages are addressed to a specific object in a specific context. Each context is addressed by its class. The objects in each context are addressed by their sequential position in the context starting at 01. The CAL interpreter locates the object and performs the command in the incoming message on object IV.

Every CEBus node must contain the universal context (00). The universal context does not model any functional system of a product. Rather, it stores the global CEBus housekeeping information about the node in the node control object.

The node control object (01) is the location for global device information such as the system and node address, the product serial number, and other information that applies to the entire product. IVs in the node control object simplify product identification on the network, perform address





configuration, and determine product capability. Table 4 shows the contents of the node control object. While only the bold IV must be used, other IVs enhance interoperability, simplify field address configuration, and should be used.

### OBJECT IMPLEMENTATION

Objects are coded as a set of IV data variables. Application software is associated with the variables. Unlike objects in C++ and similar object-oriented languages, CAL object variables are exposed to the network through the CAL interpreter. The application code that executes the object's function is hidden.

IVs may be thought of as the interface between the product application and the outside network. IVs are read or written by any node on the network (they can be protected using authentication if necessary).

The application code does the same. It checks for changed values and acts accordingly, or it updates the values.

### OBJECT NETWORK TYPES

CEBus objects are divided into three network categories: output,

Node control object (01) Node control				
Required storage object of Universal Context				
IV	R/W	Type	Name	Context function
w	R/W	b	power	Device power, 0 = off, 1 = on
l	R/W	b	on-offline	Online/offline state of device
s	R	c	serial#	Manuf. serial
n	R	c	manuf_name	Manuf. product name
m	R	c	manuf_model	Manuf. product model
c	R	c	product_class	Product class number
p	R/W	c	product_name/location	Location of product in house
h	R/W	d	<b>system-address</b>	16-bit system address
a	R/W	d	<b>node-address</b>	16-bit node address
g	R/W	d	group-address(s)	Zero or more 16-bit group addr.
b	R	n	capability_class	0, 1, 2, 3...
reset	R/W	b	reset	Resets device to factory defaults
o	R	d	<b>context_list</b>	List of used contexts in product
f	R/W	n	<b>configured</b>	1 = address configured
i	R	b	<b>setup</b>	Used during
u	R/W	n	<b>user-feedback</b>	User interface IV during
t	R/W	b	config_master	Indicate node is
d	R	d	source_mac_addr	Node addr. of last received pkt.
e	R	d	source_system_addr	System addr. of last received pkt.
k	R/W	d	authentication keys	One or more keys
R	R/W	d	reporting_condition	
H	R/W	b	reporting-header	
A	R	n	reporting-address	
P	R/W	d	previous-value	Of power IV (w)

Table 4: In the universal context, use the object table for the node control object (01).

input, and input/output. The categories define whether the object primarily sends, receives, or both sends and receives messages. Objects, represented by one of three symbols, correspond to the network type.

### CAL MESSAGES

CAL messages are generated by objects via the CAL interpreter or the object's application code. Objects communicate with other objects by setting or reading their IVs.

There are two general types of messages: command and response. The

command message is sent to a device to perform an action such as setting or reading the value of an IV. If a command message is sent that generates a return value (such as reading an IV), a response message is returned.

The CAL command message follows a specific syntax. In Figure 5, the message sets a CEBus light to 50% brightness. The message consists of a <context ID>, <object number>, and <method>, optionally followed by an IV and one or more IV arguments. The <> characters enclose an element identifier and may be made up of one or more simpler elements. The [] characters enclose optional parts. The D stands for a delimiter token (F5 hex).

As you can see, the argument in Figure 5 is the number 50 (ASCII 5, 0). Numeric values are represented in messages as ASCII since representation of a number in a product depends on how it's implemented.

<context ID>	<object #>	<method>	[<IV> [D <arguments>]]
Lighting	Light level	setValue	current_value 5 0
21	02	45	43 F5 35 30

Figure 5: Here, the general message syntax shows a message to set a light to a 50% brightness level.

The <context ID> <object #> pair forms the destination object address for the message. It identifies a particular object in a particular context in the product.

The method identifies an action to be performed by the CAL interpreter on one or more IVs in an object. Methods are usually used with one or more arguments, separated by the delimiter token. Each method operates on a specific data type. Table 5 lists a few commonly used methods.

### RESPONSE MESSAGES

A response message is generated by a node whenever the node receives a message that uses the `explicit_invoke` application-layer service, regardless of the command message. The general syntax for the

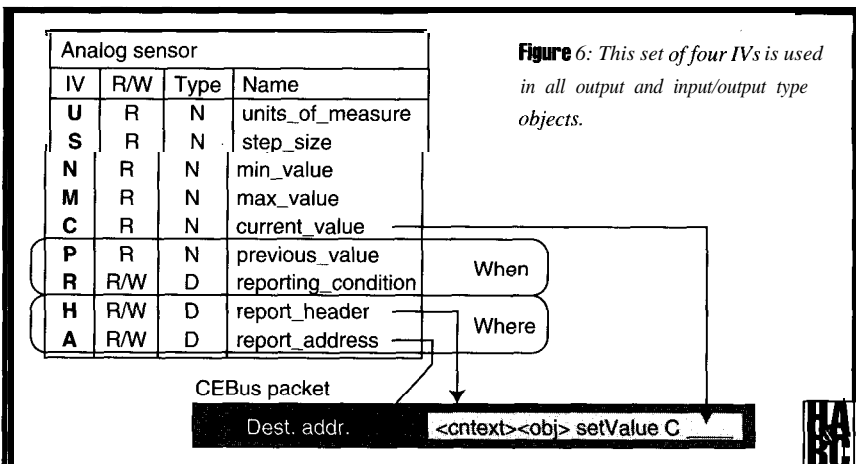


Figure 6: This set of four IVs is used in all output and input/output type objects.

response ASDU is:

<status token> [<returned data>]

The <status token> is a one-byte indicator of the type of response, and <returned data> is any data returned as a result of the command message.

The method `getValue` IV returns the contents of the IV in a response message. The <status token> can be the <result completed token> followed by <returned data>, <error token> followed by an error number, or <failure token> indicating a Boolean expression in a message evaluated false and therefore not executed.

## MESSAGE GENERATION

Objects send messages to other objects in one of two ways. The object's application code generates a message directly or it uses a reporting condition to cause the CAL interpreter to generate a message automatically. The first case is called an application message and the second method is called a reporting message.

Network output and input/output objects contain a set of optional Reporting IVs: **reporting-condition**, **report-header**, **report-address**, and **previous-value**. These IVs are used as a group by the CAL interpreter to determine when and where to send a message to another node. Their function is illustrated in Figure 6.

- **report-condition**—this function contains a Boolean expression describing a condition in the object to test whether a report message should be sent. The function uses standard CEBus Boolean expression syntax (e.g., "C > 85" (or E4 38 35).
- **report-address**—this data variable contains the node and system address of the destination device.

- **report-header**—this data variable contains the CAL message to send to the destination device (less the **current-value** argument). The message contains the context, object, method, and IV. The **current-value** is appended to the IV argument.
- **previous-value**—this value holds the **current-value** when the last message was generated. It is used by the CAL interpreter to compare the present value of **current-value**. It is automatically updated by the CAL interpreter whenever a report is sent.

Whenever **report-condition** evaluates true, the message in the **report-header**

(with the object's **current-**

Hex	Method	Arguments	Operates on: Data types			
			B	N	C	D
41	<b>setOFF</b> Sets a boolean IV to 0 (FALSE)	IV	Y	N	N	N
42	<b>setON</b> Sets a boolean IV to 1 (TRUE)	IV	Y	N	N	N
43	<b>getValue</b> Returns the contents of the IV in a response message	IV	Y	Y	Y	N
44	<b>getArray</b> Returns count number of bytes of the data IV starting at the offset byte. The default offset is 0 (beginning of the data), the default count is all of the data.	IV [z <offset>] [z <count>]	N	N	N	Y
45	<b>setValue</b> Sets the IV to <value>. If no <value> is given, the IV is set to the object default value.	IV [z <value>]	Y	Y	Y	N
46	<b>setArray</b> Stores <data> bytes into a data IV starting at an <offset> number of bytes. The default <offset> is 0 (beginning of the IV).	IV z [<offset>] z <data>	N	N	N	Y
56	<b>if</b> Used for the conditional execution of <message list> based on the result of <boolean-expr>. <message list> can contain any message that is valid for the Object. <boolean-expr> may contain any IV that is used in the Object.	<boolean-expr> BEGIN <message list> [ELSE clause] END				

Table 5: Here are several of the most common methods showing arguments and applicable data types.

**report-value** appended) is sent to the address in **report-address** to update the value of a target IV. The **report-message** contains the target context or object address. The advantage of using reporting is that it is field programmable since the reporting IVs can be written after product installation.

## OBJECT BINDING

*Object binding* establishes an address correspondence between a network output object and one or more network input objects, usually stored in the object reporting IVs. Most objects in each CEBus context are intended to work (or bind) directly with specific objects in other contexts.

The binding is expressed in the context interconnect diagram. Figure 7 shows a typical example of a context

interconnect diagram for two HVAC contexts. The environmental sensor context (41) resides in a device that measures temperature and/or humidity. The objects bind to a set of corresponding network input objects in the environmental status context (42). The environmental status context is used in equipment that needs to know the inside or outside temperature or humidity (e.g., a thermostat, TV, or PC).

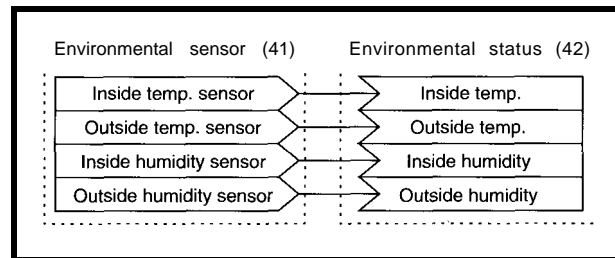
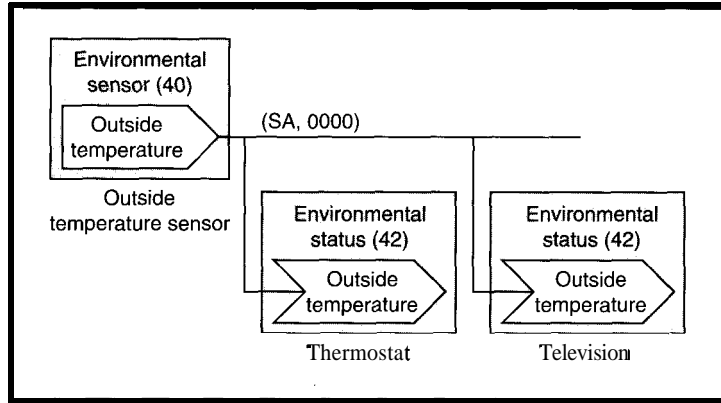


Figure 7: A simple context interconnect diagram shows the binding of output objects to input objects. Contexts can contain any combination of input, output, or input/output objects.

The objects in the environmental **sensor context** send messages reporting a change in temperature or humidity to the environmental status context. This design makes interconnecting or binding products in the field easy since only the system and node address of the device containing the environmental status context needs to be known.

A more general (and desirable) binding does not require the destination node address. Figure 8 illustrates an outside temperature sensor device containing the environmental sensor context and uses the outside temperature object (03). This product is intended to bind to all environmental status or outside temperature objects in the home. When the outside temperature object sends a message reporting an outside temperature, it uses the broadcast node address



**Figure 8:** In general binding via the broadcast node address, the outside temperature object message, which reports the current outside temperature, is picked up by all node containing context 42, outside temperature object.

(0000) so it will be received by all nodes in the house.

Figure 9 shows the implementation of the outside temperature sensor product and the thermostat. The outside sensor contains the data structure for object 03 of the environmental sensor context (41). The application code updates the `current_value IV` from the A/D converter. The **report-condition**



temperature value. The message is executed by any node in the house containing the environmental status or outside temperature object, such as the living room television and the heat-pump thermostat. The application code in the thermostat updates the outside temperature display from the `current_value`.

Note that in both products only the objects and the IVs needed for the applica-

tion. The CAL interpreter checks the `current_value` against the `previous_value IV`. When they differ by 1°, the interpreter generates a new message to the `report-address` of SA, 0000 (the house system address or the broadcast node address).

The message to context 43, object 03 updates its `current_value` to the most recent temperature value. The message is executed by any node in the house containing the environmental status or outside temperature object, such as the living room television and the heat-pump thermostat. The application code in the thermostat updates the outside temperature display from the `current_value`.

# Home Control is as simple as 1, 2, 3...

Energy Management



Security & Alarm

A

Coordinated Home Theater

A

Coordinated Lighting

A

Monitoring & Data Collection

Get all these capabilities and more with the Circuit Cellar HCS2-DX. Call, write, or fax us for a brochure. Available assembled or as a kit.

1) The Circuit Cellar HCS2-DX board is the brains of an expandable, network-based control system which incorporates direct and remote analog and digital I/O, real-time event triggering, and X-10 and infrared remote control. Control programs and event sequences are written on a PC in a unique user-friendly control language called XPRESS and stored on the HCS2-DX in nonvolatile memory.

2) The Relay BUF-Term provides hardwire interface protection to the HCS2-DX. Its 16 inputs accommodate both contact-closure and voltage inputs within the range of ±30 V. Its eight relay outputs easily handle 3 A AC or DC, to directly control solenoids, motors, lamps, alarm horns, or actuators.

3) The PL-Link provides wireless X-10 power-line control to the HCS II system. The PL-Link is an intelligent interface that sends, receives, and automatically refreshes X-10 commands. It works with all available X-10 power-line modules.

Of course, there's a lot more! The HCS II system has phone and modem interfaces, infrared remote control, voice synthesizers, and much more. Whatever your application, there's a configuration to accommodate it.

## GETSTARTED TODAY WITH AN HCS2 "123-PAK"

The 123-PAK consists of an HCS2-DX board, Relay BUF-Term board, PL-Link board, TW523 power-line interface, PS12-1 power supply, serial cable, and V3.0 HCS XPRESS software.

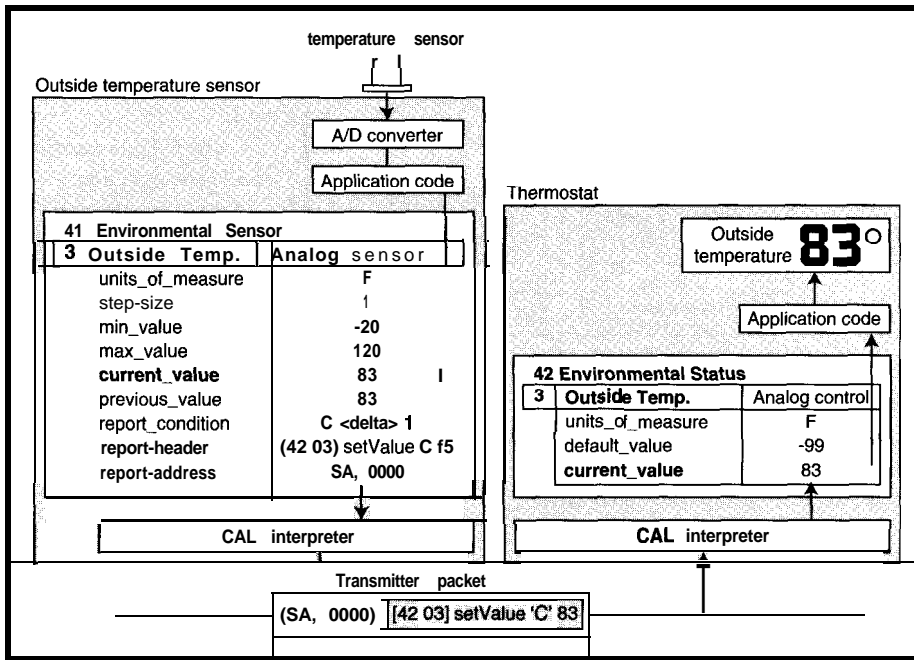
Assembled \$65 1      Kit \$461

# CIRCUIT CELLAR, INC.

4 Park Street, Vernon, CT 06066  
(203) 875-2751 • Fax (203) 872-2204

Prices shown are plus shipping.

#207



**Figure 9:** Here's an example of the outside temperature sensor device using only object 03 of the environmental sensor context. Messages are received by the inside thermostat containing the environmental status context, object 03.

tion are used. Binding via the broadcast node address makes field installation of products a trivial task.

## MAKING (OR BUYING) A CAL INTERPRETER

The good news about CAL (and the rest of CEBus for that matter) is that it can be implemented in just about any microcontroller. This is particularly beneficial if a product is already using a microcontroller and has a little spare horsepower and memory. You can either buy interpreter code or write it yourself.

The interpreter does two tasks: parse and act on incoming messages, and generate any messages required by the reporting condition IVs of objects. It must handle the following minimum tasks:

- parse incoming messages
- perform numeric conversion on message numbers
- handle resource allocation requests
- handle address configuration
- interpret reporting conditions and generate report messages
- interface to application code
- detect error conditions and send error messages

Although there are many ways to implement software that performs these tasks, the most common method is a simple

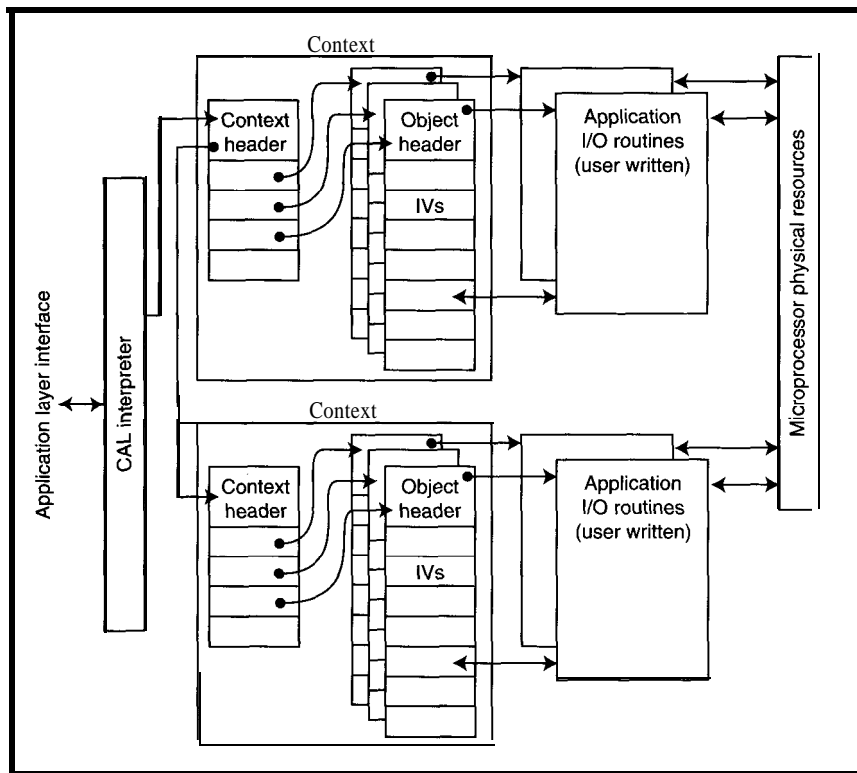
interpreter. An interpreter handles the widest class of received messages and generalizes method execution and object access. An interpreter must have some data structure

that represents the implemented contexts, objects, and IVs so that IVs can be read, written, and tested.

The data structure must also be accessible from the device application software. Figure 10 offers an example context or object data structure implemented as a series of linked lists. The context list contains one list element per context. The context element contains a list of implemented objects and a pointer to the object IV data structure. The object structure contains a pointer to the application code (if any) associated with the object.

Message parsing is straightforward. The interpreter looks at the incoming message, determines whether it can be performed based on what is supported in the device, and executes the method (Figure 11). If anything about the message causes parsing and execution to fail, an error reply message must be generated.

The CAL interpreter may call the application routines for execution, or the application routines may be contained in a main routine and call



**Figure 10:** This example of a context or object data structure is used by a CAL interpreter. Here, the application routines are called by the interpreter. A pointer to the routine is kept in the object header of the object structure.

the interpreter. In either case, execution timing requirements must be carefully watched. The CAL interpreter must keep up with incoming messages. Application routines need to keep up with changing IV values and application requirements.

Very simple CAL interpreters, capable of only minimal requirements, have been written for small microcontrollers in less than 5 KB of code. A full CAL interpreter, capable of executing the majority of parsing and reporting requirements, requires 15-20 KB.

### TWO DOWN, YOUR ONE TO GO

The advantage of CAL is that it knows through context data structures how residential consumer products operate. This makes building home automation systems, software, and

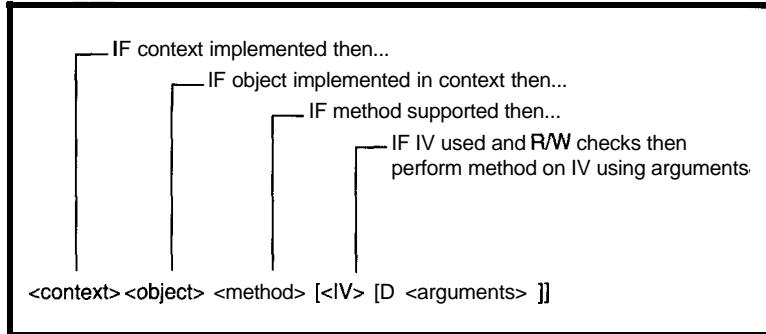


Figure 11: The CAL interpreter follows specific parsing steps for incoming message.

compliant products possible. In doing this, the CEBus protocol and physical layer specification solves the first required transition step by providing "a standardized, physical interface between devices in the home so information can be easily and reliably exchanged."

CAL also solves the second required transitional step—"a standardized way for devices to interoperate and talk to each other using a common language."

It's up to you to complete the third requirement—"a wealth of truly beneficial applications (i.e.,

things for devices to do which are available to the consumer at reasonable cost)."

*Grayson Evans has been actively involved in home automation and communication network development for the past 14 years. He currently runs The Training Department, which specializes in CEBus and*

*related home automation training, and is a technical consultant to the EIA for the development of the CEBus and other consumer electronics standards. Grayson is also founder and president of Archinetics Inc., a manufacturer of complete home automation systems. He may be reached at 71001.624@compuserve.com.*

### I R S

- 416 Very Useful
- 417 Moderately Useful
- 418 Not Useful

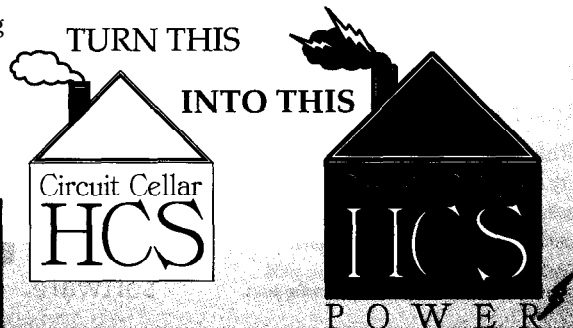


## Home owners: Hear the little word you've been longing to hear... ANNOUNCING THE HCS II VERSION 3

Now you can upgrade your HCS II to Version 3.0 and add these exciting features:

- ▶ New HCS/PC Interface
  - Query and override any system input, output, or parameter from the PC
  - Revamped HOST Program
    - Use a mouse to resize, move, open, and close all HOST windows
    - Send messages from XPRESS to a HOST window for debugging
    - Set or clear system inputs, outputs, or X-10 modules and try out speech strings from HOST
  - Modem Support
    - Call your HCS from a remote location and check its status, load a new XPRESS program, or retrieve logged data
  - Caller ID
    - Access Caller ID data from XPRESS to announce or log who's calling
- ▶ Support for more digital I/O expansion boards
  - Read and write eight netbits at a time with Netbyte
- ▶ Send messages to network modules directly from XPRESS
- ▶ Detect loss of AC power from within XPRESS
- ▶ Plus lots more for just \$60

Circuit Cellar, Inc., 4 Park St., Vernon, CT 06066  
(203) 875-2751 • Fax: (203) 872-2204





needed a human interface to my home-brew automation system. This interface had to be simple, good looking, and fully programmable.

Since my home control system is based on a PC, I could have used the monitor and keyboard, but I didn't like the PC's bulkiness and wanted the option of multiple interfaces.

My other choices included dumb terminals (too big and ugly), off-the-shelf terminals (expensive), or a home-made display terminal (perfect in all aspects, except my time).

Build my own? A great idea! I decided to call it Display-8.

## REQUIREMENTS

Once I decided to design my own display terminal, a list of features soon appeared:

- LCD display for text display
- 8 buttons for user input
- 8 LEDs for system status
- attractive enclosure-wall or table mount
- 875 1 based (easy to wire-wrap)
- powered from PC.

Since the most flexible display terminal would use an RS-232 interface to the PC, the buttons and display would then appear as a dumb terminal interface. The buttons generate the ASCII numerals 1-8 and text from the PC is displayed on the LCD display. Escape codes access LCD features and LEDs.

## DISPLAY-S HARDWARE

Figure 1 shows the schematic for the Display-8 terminal. Display-8 is based on an Intel 875 1 (U1) because the internal RAM and EPROM simplifies wire-wrapping. With the 875 1, I can use all four I/O ports as I wish: port 0 reads the key switches, port 1 interfaces to the LCD display, port 2 controls the LEDs, and port 3 performs various control functions. I also have a large amount of code already written for this processor, which should save me some time.

# The Display-8

## Add a Human Interface to Your Home Automation System

The LEDs are controlled by software writes to port 2. The switches are debounced by multiple software reads from port 0.

Display-8 includes a Hitachi LM052L 2-line x 16-character display. Typically, you would use decode logic to have the LM052L appear as external data space. However, since I have ample pins on the 875 1 and do not enjoy wire-wrapping, I connected the LM052L directly to the processor. Software controls the pins directly.

The LM052L data bus is connected to port 1 of the 875 1. The LM052L register-select pin is tied to P3.4 (T1), the read/write pin is tied to P3.6 (WR), and the enable pin is tied to P3.7 (RD).

A MAX232 transceiver from Maxim provides the RS-232 interface. This chip internally generates the  $\pm 12$  V needed for RS-232 and requires only four capacitors. Display-8 uses the 875 1's built-in serial port on pins P3.0 (RXD) and P3.1 (TXD).

Connector J1 is an RJ-11 connector through which power is supplied and serial communications take place.

For Display-8, I used C&K switches that include a DPDT switch and LED in one housing. The switch fits into a standard 14-pin socket footprint. Everything is easily wire-wrapped onto a single protoboard. Photo 1 shows the component placement.

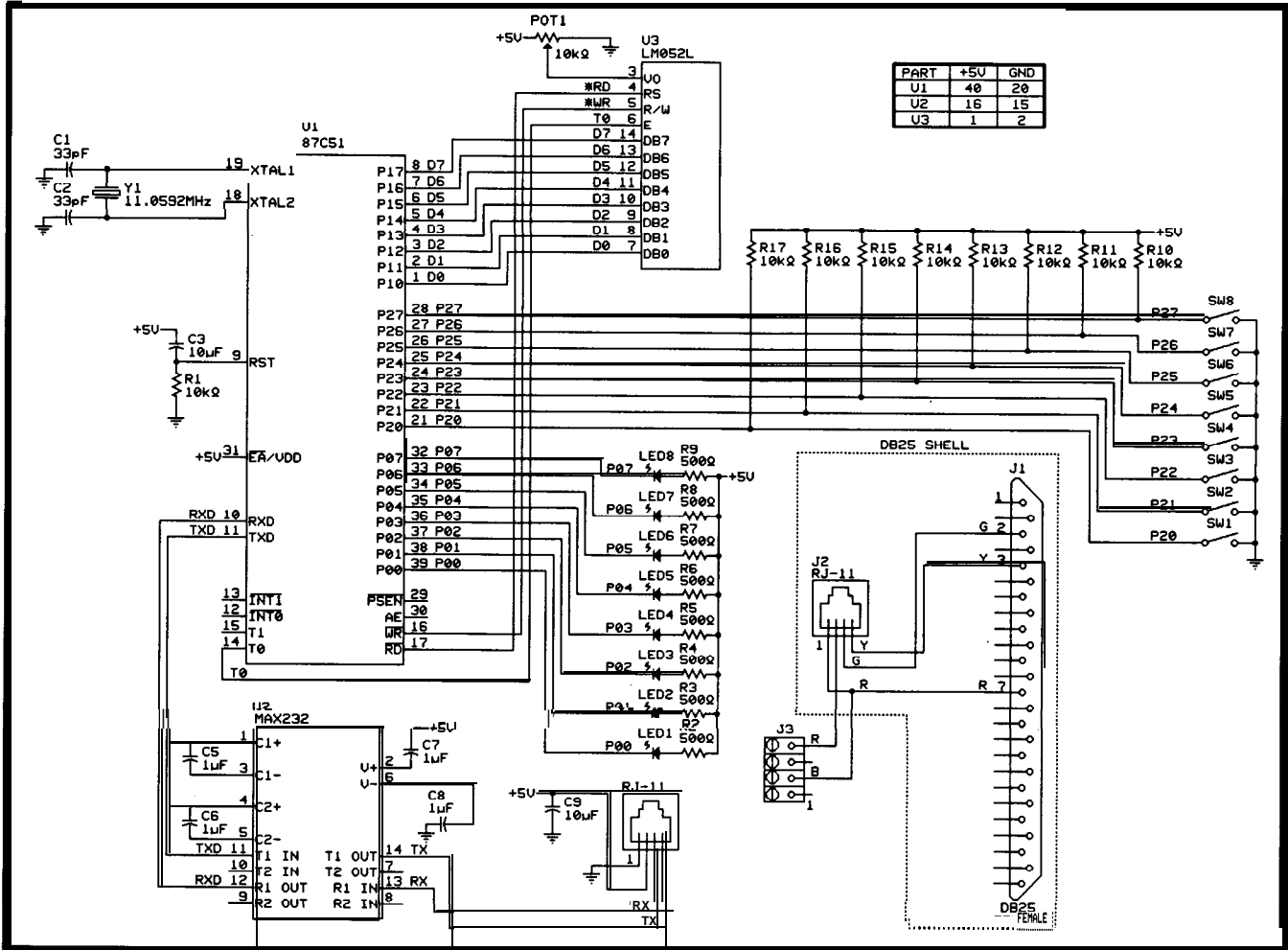
## DISPLAY-S ENCLOSURE

The most important issue of this display was that it needed to be attractive. I used a premade aluminum enclosure from Radio Shack, so I was able to drill and cut holes for the display and

## MITCH DRUMMOND

While many home control systems operate passively in the background, users **often** like some hands-on interaction with the system. Display-8 offers that interaction in a small, unobtrusive package.





**Figure 1:** Display-8 is based on the 87C51 microcontroller for simple assembly. The switch, LED, and LCD hardware decoding is performed by the microcontroller software.

key switches. (If you implement this idea, remember to make your package safe by filing the cut edges.)

For a tabletop unit, the enclosure comes with rubber feet and the cable exits in the rear. My unit is wall mounted and includes a cutout so it can be mounted over a standard electrical box. I painted the finished display white to match the decor of the house.

### PC CONNECTION

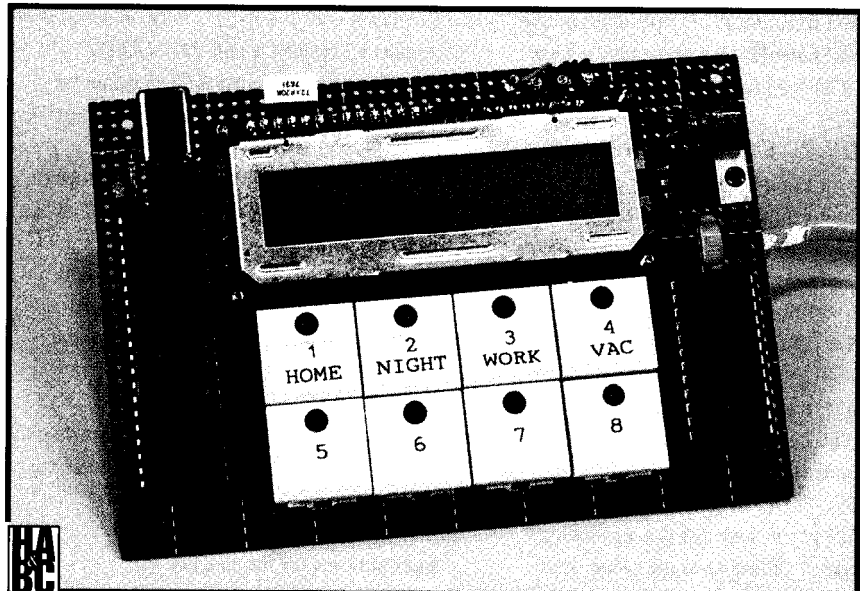
Since this display is powered from the PC, I used an adapter cable at the PC to combine the serial interface and the power supply (see Figure 1). I used a DB-25-to-RJ-11 converter and a disk drive power supply extension cable from Radio Shack.

**Photo 1:** IC sockets protect the components during assembly. The resistors are located under the LCD display. Once placed in its enclosure, Display-8 is ready for table or wall mounting.

### DISPLAY-8 SOFTWARE

The more complicated issue was the 875 1 software. I wanted to be sure the program fit into the 4 KB of program space,

but more importantly, the 128 bytes of data space. For simplicity, I chose to use the Archimedes 805 1 C cross-development tools. This choice





**Listing 1:** Interrupts are only used for input with the 8031 serial I/O routines.

```

interrupt void serial_isr(void)

if (RI) {
    rx_buff[rx_buff_head] = SBUF; /* store char in buffer */
    RI = 0;
    ri_flag = 1;
    rx_buff_head++; /* increment head pointer */
    rx_buff_head &= RX_BUFF_MASK; /* Wrap around the end */
    if (rx_buff_head == rx_buff_tail) { /* If at tail, */
        rx_buff_head; /* drop character instead */
        rx_buff_head &= RX_BUFF_MASK;
    } /* end if RI */

if (TI) {
    TI = 0; /* tell foreground routine */
    ti_flag = 1;
}

return; /* end serial_isr */
}

char get_char(char * c)

if (rx_buff_head != rx_buff_tail) { /* check for empty buf */
    *c = (rx_buff[rx_buff_tail]);
    rx_buff_tail++; /* increment tail pointer */
    rx_buff_tail &= RX_BUFF_MASK; /* wrap-around if needed */
    return TRUE;
}

else {
    ri_flag = 0; /* empty buf, clear flag */
    return FALSE; /* and fail. */
} /* end get_char */

void send_char(unsigned char c)

while (ti_flag == 0);
ti_flag = 0;
SBUF = c;
return; /* end send-byte */
}

```

worked out very nicely in that I did not have to do any assembly language programming. All interrupt service routines (ISRs) and output routines were written in C.

The software includes ISRs for both the serial port and the 5-ms timer tick. The timer ISR simply sets a global flag that is checked by the main routine. The main routine calls `scan_input` when the timer flag is set. The `scan_input` routine first reads the current state of the input pins and stores the current states in an array of characters.

The routine then integrates the input values for a count of `MAX_CNT` (three reads or 30 ms). If the integrated input state is 0, then a button is

pressed and a character is sent to the host. A hex 31 is added to the switch number (`cn_t`) to indicate an ASCII 1-8.

The serial ISR is not as simple. Listing 1 shows the serial port routines. The serial ISR first checks the RI flag. If a character has been received, `SBUF` is read and stored in a circular buffer and the head (input) pointer is incremented.

The OR function checks for wrap around. If the head pointer is equal to the tail pointer, the buffer has overflowed. In this case, the head pointer is decremented, and the last character is lost. The ISR sets `ri_flag` to indicate that a new character is in the input buffer. If the serial ISR detects a transmit interrupt, the output buffer is available and `ti_flag` is set for the foreground routine.

The main routine checks the variable `ri_flag` and calls the routine `get_char` to get the character from the serial buffer. This routine checks that the head pointer does not equal the tail pointer. The character is read from the address of the tail pointer, and the tail pointer is incremented. If the tail pointer equals the head pointer, the buffer is empty and the `ri_flag` variable is cleared.

The software in Display-8 supports several control characters including formfeed (0Ch), new line (0Ah), carriage return (0Dh), and backspace/delete (08h). Also, escape codes control the LEDs and the special functions of the LCD driver.

Listing 2 shows the routines involved in character output. The `display()` routine is called for each character that is received. Based on the last character received (possibly an escape code), the correct action is taken. If an escape code is received, a flag is set and the routine returns while waiting on the next character.

The normal characters are sent to the `lcd_putchar()` function, which controls the LCD. This function looks for control characters and sends characters to the LCD using `lcd_char()` and control commands using `lcd_control`.

The secret to the `lcd_putchar()` routine is to keep a copy of the display in memory (variable `disp_copy[1]`). The LCD controller has memory for a 2 x 40 display, but I am using only a 2 x 16 display. The software only stores an image of the 2 x 16 memory.

Also, the position of the cursor (row and column) is tracked by the global variables `row` and `col`. If a new line is found, the cursor advances to the next line. If it is already on the second line, the second line is copied to the first line, and the cursor is positioned at the start of the second line.

After each character is sent to the display, the position is incremented and checked for a wrap-around condition. When a wrap-around condition occurs, the cursor is repositioned, and the second line is copied to the first if necessary.

A formfeed character executes a display home command that clears the display and positions the cursor in the first position. A `backspace` command repositions the cursor back one space, prints a blank, and repositions the cursor again. A carriage return simply repositions the cursor at the beginning of the current row.



## OPERATION

When plugged into a PC, Display-8 can be used with any communications program. Characters sent to Display-8 are echoed (so you can see what you're typing). Consult the LCD data manual for a complete list of the characters available. Carriage return, new line, formfeed, and backspace characters are all supported by Display-%

When buttons on the Display-S are pressed, the ASCII characters 1-8 are sent to the PC communications program.

To control the LEDs, send an **LED\_CHAR** command (hex OE, dec 14) and then a bitmap of which LEDs to turn on. LED 1 corresponds to the least-significant bit of the byte.

Other special functions of the LCD can be executed by issuing a **CONT\_CHAR** command (hex OF, dec 15) and then the LCD control byte. This byte is written to the control register of the LCD display.

Graphics characters can also be defined in this way. Data read-back capabilities of the LCD display are not available to the serial port user.

## FINALE

Display-8 provides an excellent user interface to my home control system. In fact, I like the design so much that I am planning on building more units to use around for other control functions.

*Mitch Drummond is a design engineer for the cable television industry. Mitch has been developing embedded systems for military and commercial applications for over ten years. He's currently developing a home automation system in his spare time. Mitch may be reached at [mitchd@uvsg.com](mailto:mitchd@uvsg.com).*

## SOFTWARE

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

## I R S

- 419 Very Useful
- 420 Moderately Useful
- 421 Not Useful

Listing 2: *The function `lcd_putchar` controls the LCD's I/O pins for writing data to the display.*

```
void lcd_putchar(unsigned char c_out)

    int i;

    if (c_out == '\a') {          /* No bell on display, ignore. */

    else if (c_out == '\b'){     /* backspace, delete char. */
        if (col > 0){
            lcd_control((char) (DS_ADDRESS+((row*40)+col-1)));
            lcd_char(' ');
            lcd_control((char) (DS_ADDRESS+((row*40)+col-1)));
            disp_copy[row*COL_CNT+col-1] = ' ';
            col --;
        }

    else if (c_out == '\f'){     /* formfeed, so clear screen */
        lcd_control(DS_HOME);
        row = 0;
        col = -1;
        for (i=0; i<ROW_CNT*COL_CNT; i++)
            disp_copy[i] = ' ';
    }
    else if (c_out == '\n'){     /* new line. If on 2nd row, */
        if (++row == ROW_CNT){   /* copy to first. */
            lcd_control(DS_HOME); /* Clear the display */
            for (i=0; i<COL_CNT; i++){ /* copy 2nd row to 1st row and */
                disp_copy[i] = disp_copy[i+COL_CNT]; /* clear 2nd row memory */
                lcd_char(disp_copy[i]);
                disp_copy[i+COL_CNT] = ' ';
            }
        }
        row = 1;

        col = -1;                /* put cursor at start of 2nd row. */
        lcd_control((char) (DS_ADDRESS+40));

    else if (c_out == '\r'){     /* Insert carriage return, put */
        col = -1;                /* cursor at first column */
        lcd_control((char) (DS_ADDRESS+(row*40)));

    else {
        lcd_char(c_out);
        disp_copy[row*COL_CNT+col] = c_out;

    if (++col == COL_CNT){      /* inc ptr; process next char */
        if (++row == 2){        /* copy 2nd row to 1st row and */
            lcd_control(DS_HOME); /* clear 2nd row memory */
            for (i=0; i<COL_CNT; i++){
                disp_copy[i] = disp_copy[i+COL_CNT];
                lcd_char(disp_copy[i]);
                disp_copy[i+COL_CNT] = ' ';
            }
        }
        row = 1;
    }
    col = 0;                    /* reposistion cursor in correct place */
    lcd_control((char) (DS_ADDRESS+((row*40)+col)));
    }
    return;
}

void lcd_char (unsigned char c_out)
{
    wait_bf();                  /* Write character to display */
    DATA_P = c_out;           /* wait for the LCD ready flag */
    RW = WRITE;                /* place the data on the bus */
    RS = DATA;                /* set to write */
                                /* select the data register */
}

```

(continued)

**Listing 2: continued**

```

E = 1;          /* pulse the enable line */
E = 0;
return;        /* return, assume it worked */

void lcd_control(unsigned char c_out)
{
    wait_bf(); /* Write ctrl byte to display */
    DATA_P = c_out; /* wait for the LCD ready flag */
    RW = WRITE; /* place the data on the bus */
    RS = CONTROL; /* set to write */
    E = 1; /* select the control register */
    E = 0; /* pulse the enable line */
    return; /* return, assume it worked */
}

void wait_bf(void)

char bf;

DATA_P = 0xff;
RS = CONTROL;
RW = READ;
do {
    E = 1;
    bf = DATA_P.7;
    E = 0;

while (bf);
return;

void init_disp(void)

char i, j;

for (i=0; i<125; i++){ /* wait for power to settle and*/
    delay(); /* init display. Pause 15 ms */

RS = CONTROL;
RW = WRITE;
DATA_P = 0x38; /* 8 bit, 2 lines. 5 x 7 font */
for (j=0; j<4; j++){
    E = 1;
    E = 0;
    for (i=0; i<35; i++)
        delay(); /* 4.1 ms */

lcd_control(0x08); /* display/cursor/blink off */
lcd_control(0x01); /* clear display */
lcd_control(0x06); /* increment dd ram, no shift */
lcd_control(0x0C); /* disp on, cursor/blink off */

display( 'H'); /* Just a test for the PROM */
display( 'e');
display( 'l');
display( 'l');
display( 'o');
display( ' ');
display( 'W');
display( 'o');
display( 'r');
display( 'l');
display( 'd');
display( '!');
display( '\r'); /* return home */
return;

```



#214

## Motion Control

From the software to the motor shaft! Everything you need to automate a machine tool or an OEM motion job.

Artisan-CNC software is a true 4-axis machine controller w/user interface, not just a 'driver'. It takes tool-path files in industry std G-codes, HPGL, or Excellon PCB drill codes. It's been refined & updated since 1989.

Fancy features include: Continuous contouring, feedrate override, cutter radius compensation, big 1" screen read, toolpath graphics: PLC macro language for automation & I/O xtrol, & much, much more.

Very fast assy language runs on a 386/486 VGA to 40K steps/sec/axis! Indexer-I/O card for ISA bus included in prii. 130 pg tech-ref manual & 500K Help.

Make a PCB drill, CNC a Mill in your shop, or use as OEM for machine tools, cut-to-length, labeling, etc..

Software from \$349 - \$849. Systems from \$2K-\$5K. Call us for the whole scoop. Use 800# to get catalog or to order \$15 60pg Tech Info-pak + demo disk. Call 612-641-1797 for a sales engineer.



**ah-ha!**  
Design Group, Inc

We answer questions. Take the first CNC step & call us at 612-641-1797  
Box 14519 Mpls, MN 55414, 800-473-7650, 10-6pm CST

#212

## Tech-Arts Home Automation

• **Text to Speech Board** serial I/O \$ 89

• **Temperature boards:** w/16 sensors \$239  
w/8 sensors plus 8 analog inputs \$179  
both boards: -40° to 146°F, serial I/O

• **Digital I/O ISA cards:** 48 I/O ports \$125  
96 I/O ports \$ 169  
192 I/O ports \$249

• **4-Port ISA Serial Board** w/1 6550s \$129  
com 1-8 & irq's 2,3,4,5,10,11,12,15

• **J-Servo controller board** serial I/O \$ 89

• **Windows NT TelcomFAX Personal** \$129

-Automatic Drapery Controller \$ 139

**Call for our complete catalog!**

support 315-455-1003

308 E. Molloy Rd Fax 315-455-5838

Syracuse, NY 13211 BBS 315-455-8728

Promotions and prices are subject to change without notice. Call for guarantee and warranty information.

**800-455-9853**  
Visa • MC • Amex • COD

#213

## Home Automation



## Worthington Distribution

**1-800-282-8864**

NO MINIMUMS

NO HANDLING FEES

TRUE DEALER PRICING

36 Gumbetown Road. Paupack, PA 18451

#214

74 Firmware Furnace

82 From the Bench

86 Silicon Update

92 Embedded Techniques

100 ConnecTime

## FIRMWARE FURNACE

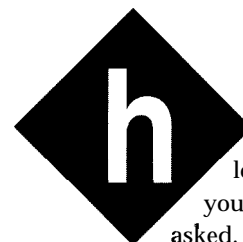
Ed Nisley

# Journey to the Protected Land: The Mystery of Scan Code Set 3



Just a little chat—with your keyboard

that is. Building on the principles covered in last month's column, Ed shows us how to coax the PC into talking to the keyboard.



Have you ever looked at one of your old projects and asked, "Who wrote this code and what was I thinking?" I know I have! Sometimes you can replace a page of tortured logic with a single, obvious, crystal-clear function.. .that is, until you look at it again in a few years.

Last month, you saw how the PC's keyboard evolved from a fairly simple subsystem into a complex mess. Each change made sense at the time, but the end result is essentially incomprehensible. Imagine designing a system that produces eight bytes for a single key.

This month, I'll examine the keyboard hardware and firmware built into every PC. Protected mode gives us the opportunity to switch the keyboard's fundamental operation into something sensible. As you'll see, talking to the keyboard exercises some interesting machinery.

Crystal clear? Check it again next year!

### CONTROLLING THE CONTROLLER

The keyboard controller on the system board is an 8042 Universal Peripheral Interface, also known as a microcontroller. It's generally easy to spot since a 40-pin DIP looks terribly out of place on a system board where

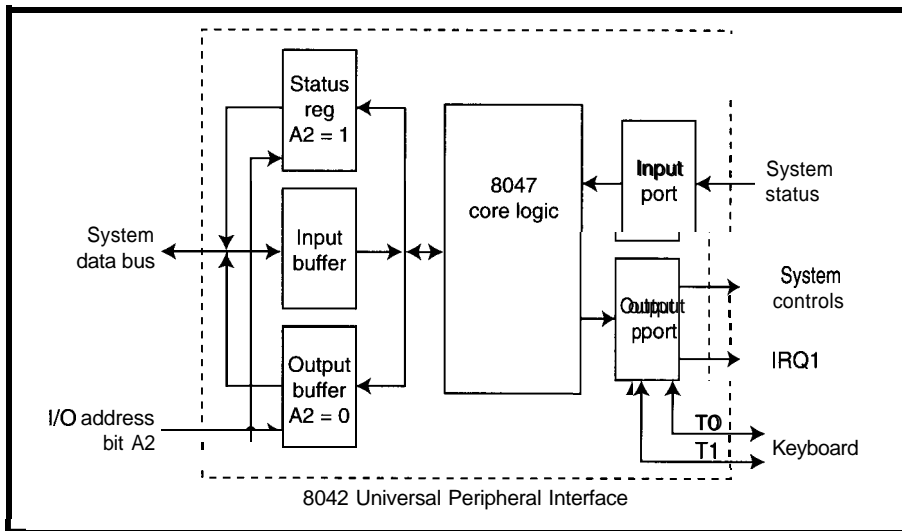


Figure 1—The system board keyboard controller is an 8042 microcontroller interfaced to the main CPU through a pair of byte-wide I/O ports at addresses 60 and 64. Port 64 sends commands to the controller and returns status bits; the values read and written are not identical. Port 60 is a bidirectional data port. The 8042 handles the keyboard's serial data format and serves as an interface to the system board's status and control bits.

three or four surface-mount LSI parts hold several million logic transistors. Current systems sport genetically engineered 8042 descendants stuffed with specialized speed-up hardware.

A great deal of weirdness surrounding the keyboard controller goes away as you read the Intel 8042 data sheet. Figure 1 sketches the key features. As you can see, all the logic is inside the 8042! The main system's only access is through two I/O ports at addresses 60 and 64 (hex) and the output driving IRQ 1.

Reading port 60 selects the 8042 Output Buffer through which the controller sends keyboard scan codes and other information to the '386SX. Reading port 64 selects the 8042 Status Register, which contains the five firmware-controlled bits and three

hardware flags shown in Listing 1. Internal hardwired logic sets Status Register bit 0, the OBF flag, when the 8042 firmware writes a byte into the Output Buffer and clears it when the '386SX reads the byte.

The 8042 firmware controls an internal gate that routes OBF to the pin driving IRQ 1. Each time the firmware writes a byte into the Output Buffer, IRQ 1 goes high and the '386SX CPU executes the IRQ 1 interrupt handler. You can enable and disable this interrupt in three places: the 8042, the 8259, and the CPU's Interrupt Flag.

Writing a byte to either port 60 or port 64 loads the 8042 Input Buffer and sets Status Register bit 1, the IBF flag. Hardwired logic also copies I/O address bit 2 into Status Register bit 3 on each write, giving the firmware an

easy way to distinguish the two sources. IBF goes low when the 8042 firmware reads the Input Buffer.

The keyboard controller firmware accepts data through port 60 and commands through port 64. Although the '386SX CPU writes to two separate ports, they neck down to a single chunk of 8042 hardware. You must verify that IBF is low before writing to either port, lest you overwrite a previous value before the 8042 has read it.

With that hardware background in mind, the keyboard controller should make more sense. The 8042 firmware recognizes about 120 commands written to port 64, most of which are entirely irrelevant for normal operation. Listing 2 shows the few commands used by the FFTS keyboard interface.

The terminology doesn't help much. The '386SX writes these commands to port 64. One of the commands, 60 hex, is *Write Command Byte*. The *Command Byte*, shown in Listing 3, is subsequently written to port 60. The only way to get familiar with this stuff is to use it.

Bit 6 of the Command Byte, *Trans 1* ate, determines whether the controller translates raw keyboard scan codes into system scan codes. This bit is normally on because the keyboard defaults to Scan Code Set 2, the baroque multibyte scheme I described last month. When *Trans 1* ate is zero, the controller passes the keyboard's scan codes directly to port 60 without modification. As I'll discuss later, turning this bit off is essential for the FFTS interface.

When bit 4, *DisableKbd*, is on, the 8042 firmware forces the keyboard clock line low to prevent the keyboard from sending anything to the controller. Normally, your set-up routine turns this bit on and your operating code uses the Enable and Disable Keyboard commands to flip the bit, thus eliminating the need to write a new Command Byte.

Setting bit 0, *EnableInt*, on enables the IRQ 1 interrupt whenever the OBF bit goes on. The hardware doesn't care why the Output Buffer became full; OBF goes on whenever the 8042 firmware writes a byte

Listing 1—You may read the keyboard controller status byte from port 64 at any time. The 8042 sets *Output Full* whenever a byte is available at port 60. This flag also triggers *IRQ 1* if interrupts are enabled. The hardware sets *Input Full* immediately after each write to port 60 or 64, indicating that the firmware has not yet processed the byte. The controller sets *ParityError*, *RecTimeout*, and *Trans Timeout* after performing several retries on its own.

```

RECORD          STATFLAGS {
StatFlag_ParityError:1; kbd serial parity error
StatFlag_RecTimeout:1 ; timeout during kbd message
StatFlag_TransTimeout:1 ; timeout after ctl message
StatFlag_NoKeylock:1 ; 0 = keylock switch ON
StatFlag_CmdReceived:1 ; 1 = last write to Port 64
StatFlag_SysFlag:1 ; keyboard OK or SysFlag = 1
StatFlag_InputFull:1 ; 1 = no write to 60/64
StatFlag_OutputFull:1 ; 1 = read data from 60

```

whether it's a scan code or a response to a command. You must ensure that your keyboard interrupt handler is never surprised by a byte that isn't a keyboard scan code.

The three routines shown in Listing 4 handle low-level system keyboard controller I/O. I have omitted statements that generate the tracing and debugging messages you'll see later.

## KEYBOARD CONVERSATIONS

Notwithstanding the preceding discussion, the system keyboard controller's main purpose in life is converting between keyboard serial data and PC parallel data. Generally, our code talks directly through the keyboard controller to the keyboard itself, so we must know what the microcontroller under the keycaps expects to hear.

Unless the system keyboard controller is busy processing a command written to port 64, it simply passes data written to port 60 directly to the keyboard in serial form. The IBF flag indicates that the data previously written to either port 60 or 64 hasn't been processed yet. Listing 4b takes care of this with the same code that writes data to the Input Buffer after a system-keyboard-controller command.

Similarly, the controller converts any data arriving from the keyboard to parallel form and places it in the Output Buffer register, triggering IRQ 1 if interrupts are enabled. Reading port 60 extracts the data, resets OBF, and clears IRQ 1. The controller disables the keyboard while OBF is set, eliminating the possibility of an overrun.

Listing 4c reads Output Buffer bytes without regard to where they came from and thus returns bytes from the keyboard as well as the system keyboard controller. This code is useful only after commands producing a response byte because the normal keyboard scan codes should go to the IRQ 1 interrupt handler.

Serial data flows between the system keyboard controller and the keyboard at a peak rate of about 10 kbps (see Photo 1, FF, *INK* 59). The average data rate is much lower, limited mainly by delays between the

**Listing 2—***Most of the keyboard controller commands are not useful during normal operation. The FFTS routines use these few commands. Listing 3 shows the bits in the Command Byte.*

```
CCMD_RDCMD      = 020h    ; read controller command
CCMD_WRCMD      = 060h    ; write controller command
CCMD_TEST       = 0AAh    ; test controller
CCMD_DISABLE    = 0ADh    ; disable keyboard
CCMD_ENABLE     = 0AEh    ; enable keyboard
CCMD_SYSRESET   = 0FEh    ; reset the entire system!
```

bytes—even the fastest typists have trouble generating a few hundred keystrokes per second. I'd say the link is fast enough.

The communications protocol's details aren't of much use to us right now. Suffice it to say that when the system keyboard controller lowers the clock line, the keyboard cannot transmit information and stores keystrokes in its internal buffer. You must disable the keyboard using the system keyboard controller Disable Keyboard command (AD hex) before issuing any commands using the Input or Output Buffers. If the keyboard remains enabled, it may send a byte that arrives just after your command and confuse the proceedings.

The keyboard sends an acknowledgment for every byte it receives from the system. If the byte had good parity and timing the keyboard sends FA hex (pronounced "Ack"). Most errors results in FE hex [say "Error"]. Mercifully, the system keyboard controller handles error conditions by resending the byte several times before setting the Status Register error bits.

Listing 5 shows how to send a byte to the keyboard and process the acknowledgment. I suspect the only proper response to an error that makes it past the system keyboard controller's retries is "Your keyboard just died." In this routine, I simply ignore persistent errors and continue without complaint.

The keyboard recognizes about a dozen commands, many of which aren't relevant to our purposes. Listing 6 shows the few we'll need for the FFTS routines.

Now onto the fun part!

## CONVERTING THE CODES

The complexity of deciphering all the scan codes produced in all the shift states for all the keys seemed too daunting when I first looked into this topic. Frankly, writing a replacement BIOS keyboard handler wasn't something I wanted to tackle!

When faced with an impossible situation, sometimes you can restate the problem so the solution is obvious. In this case, a light went on when I read that PS/2 and some other key-

**Listing 3—***This Command Byte defines the controller's overall operating mode. If the system includes a F'S/P-style system-board mouse port, several of the Command Byte bits have different meanings. The BIOS sets Translate, SysFlag, and EnableInt after each system reset. The FFTS keyboard routine turns Translate off to gain direct access to the keyboard's scan codes.*

```
RECORD CTLFLAGS {
  CtlFlag_Res7:1=0      ; reserved, must be 0
  CtlFlag_Translate:1=1 ; translate AT to PC
  CtlFlag_PCMode:1=0   ; use PC serial intf
  CtlFlag_DisableKbd:1=0 ; force kbd clock low
  CtlFlag_DisableInhibit:1=0 ; override keylock
  CtlFlag_SysFlag:1=1  ; 1 after BIOS setup
  CtlFlag_Res1:1=0     ; reserved, must be 0
  CtlFlag_EnableInt:1=0 ; enable IRQ 1 interrupt
```

```
CMD_NORMAL = (MASK CtlFlag_SysFlag) + MASK CtlFlag_EnableInt
```

**Listing 4-a)** The *KeySendCmd* routine waits until the 8042 Input Buffer Full flag is clear, then writes the command to port 64. b) *KeySendData* waits until both the Input and Output Buffer flags are clear, then writes the data byte to port 60. Any scan codes arriving before *KeyFlushOutput* are lost. c) The *KeyReadData* routine polls the Status Register until the Output Buffer flag goes high, then reads the byte. The keyboard interrupt handler snags the byte first unless you disable IRQ1.

```

a)
PROC   KeySendCmd
ARG    CmdByte: DWORD
USES   EAX

CALL   KeyWaitInBuff      ; wait for prev cmd to finish

MOV    EAX,[CmdByte]      ; fetch the command
OUT    KEY_CMD,AL         ; send it out

RET
ENDP   KeySendCmd

b)
PROC   KeySendData
ARG    DataByte: DWORD
USES   EAX, ECX

CALL   KeyWaitInBuff      ; wait for cmd to clear
CALL   KeyFlushOutput     ; discard any pending bytes

MOV    EAX,[DataByte]    ; fetch the data
OUT    KEY_DATA,AL       ; send it out

RET
ENDP   KeySendData

c)
PROC   KeyReadData
USES   ECX

MOV    ECX,MAX_DATAWAIT  ; maximum delay

@Stall:
IN     AL,KEY_STATUS      ; key ready yet?
TEST   AL,MASK_StatFlag_OutputFull
JNZ    @Fetch

CallSys CGT_TMR_DELAYMS,1; nope, wait a bit
LOOP   @@Stall           ; and try again

@Fetch:
IN     AL,KEY_DATA        ; fetch the data
Punt
MOVZX  EAX,AL             ; clean up the byte
MOV    [LastCode],EAX    ; and save for examination

RET
ENDP   KeyReadData

```

boards supported three scan-code sets, one of which produced a single byte per keystroke. I knew a bit about why Scan Code Set 3 existed, a tale told here last month. The only remaining question was how many clone keyboards supported that feature.

Rick Freeman and the folks at Computer Options here in Raleigh graciously loaned me one of every

keyboard in the store. Adding those to my ragtag collection, I checked out a dozen Enhanced keyboards and found that, with a single exception, they all supported Scan Code Set 3. The oddball, a Northgate C/T keyboard, dates back to 1988 when Enhanced keyboards were very, very new.

While that's not conclusive proof that all PC keyboards respond cor-

## FREE Data Acquisition Catalog



# 1995

PC and VME data

acquisition catalog

from the inventors of  
plug-in data acquisition.

Featuring new low-cost  
A/D boards optimized  
for Windows,

DSP Data Acquisition,  
and the latest

Windows software.

Plus, informative  
technical tips and  
application notes.

Call for your free copy

**1-800-648-6589**

## ADAC

American Data Acquisition Corporation  
70 Tower Office Park, Woburn, MA 01801  
phone 617-935-3200 fax 617-938-6553

rectly to the code in FFTS, it gave me enough confidence to try this trick. I'm interested in hearing how this works out on your system. But, if your keyboard doesn't support Scan Code Set 3, you get to decode its output. I want no part of Scan Code Set 2, thank you very much.

The code in Listing 7 handles the transition from the BIOS default keyboard settings to the new conditions. It runs the self-test routines in the system keyboard controller, and the keyboard then sends several commands to the keyboard. If any of the first few commands fail because of a missing keyboard or missing feature, the keyboard is disabled and unusable in FFTS.

I always set my keyboards for the shortest typematic delay (250 ms) and the fastest repeat rate (30 characters per second). Recognizing that your reflexes may vary, I've declared a group of constants that go all the way to stun: two characters per second after a one second delay.

Scan Code Sets 1 and 2 automatically set all keys to typematic-make-break mode, leaving the BIOS to sort out the superfluous make codes. Scan Code Set 3 works differently, placing only the typewriter and cursor keys in typematic mode. Most of the remainder are make-only keys that send a single code when they're pressed, do not repeat no matter how long they're down, and do not send a break code when they're released.

Most of the shift keys operate in make-break fashion, sending only a single make code and a single break code. Oddly enough, the right-Alt and right-Ctrl keys are make-only, which is sensible when you see the main-frame and minicomputer keyboard keycaps: one is an Enter key and the other does something similar. Remember, PCs aren't the only computers in the world!

The good news is that you can reprogram the key modes to suit your needs. A single command sets all the keys to the familiar typematic-make-break mode used in the other Scan Code Sets. This mode is appropriate for typewriter keys, cursor keys, and a few others, eliminating the need to

**Listing 5—**The keyboard acknowledges each byte if receives with either FA hex (good) or FE hex (error). This routine resends the byte a few times and then simply ignores the error. You must disable the keyboard before calling this routine to ensure that Key F7 us h Oup t doesn't inadvertently discard a keystroke.

```

PROC   KeySendDataAck
ARG    DataByte: DWORD
USES   ECX

      MOV     ECX, MAX_RETRIES    retry counter

@Resend:
      CALL   KeyWaitInBuff       wait for cmd to clear
      CALL   KeyFlushOutput      discard any pending bytes

      MOV     EAX, [DataByte]    ; fetch the data
      OUT    KEY_DATA, AL       ; send it out

      CALL   KeyReadData         ; fetch byte
      MOVZX  EAX, AL            ; clean it up
      CMP    AL, KRSP_ACK       ; ack?
      JE     @Done              ; yes, done
      CMP    AL, KRSP_RESEND    ; resend?
      JNE    @@Done             ; no, ignore it
      LOOP  @Resend             ; yes, try again

@@Done:
      RET
      ENDP   KeySendDataAck

```

reprogram each and every key individually.

Three additional commands set individual keys to make-only, make-break-only, or typematic-only mode. The key's make scan code follows the command byte, which means you must program each key individually. The keyboard accepts these commands when any Scan Code Set is active, but they apply only to Scan Code Set 3.

The make-break mode is a natural for shift and lock keys as the make code indicates that the key is down and the break code says it's up. There

is no need to process and discard additional make codes.

The most useful function keys in make-only mode are Esc, Ins, Home, and End. These keys produce a single scan code that triggers a single action. Again, not having to deal with repeated keys simplifies programming.

Some applications can probably take advantage of typematic-only function keys that send repeated make codes with no break code at the end. All the FFTS definitions reside in a table, making it easy to contort the keyboard to suit your needs.

**Listing 6—**The keyboard responds to a variety of commands sent through the system keyboard controller. Some systems include an onboard mouse port driven by the system keyboard controller; the mouse controller responds to a slightly different command set. This list includes the most useful keyboard commands.

```

KCMD_WRL EDS    = 0EDh ; write to keyboard LEDs
KCMD_CODEMODE = 0F0h ; get/set scan code mode
KCMD_RDID     = 0F2h ; read keyboard ID
KCMD_RATE     = 0F3h ; set typematic delay & rate
KCMD_ENABLE   = 0F4h ; enable keyboard
KCMD_ALL_TMB  = 0FAh ; all keys typematic/make/break
KCMD_ONE_T    = 0FBh ; set single key to typematic
KCMD_ONE_MB   = 0FCh ; set single key to make-break
KCMD_ONE_M    = 0FDh ; set single key to make-only
KCMD_RESET    = 0FFh ; reset to power-on defaults

```



Listing 1-The protected-mode *FFTS* keyboard interface uses Scan Code Set 3 because it's much easier to process than the PC default, Scan Code Set 2. This routine tests the system keyboard controller and the keyboard, sets the keyboard's new operating modes, and prepares the interrupt handler. If the keyboard isn't present or doesn't support Scan Code Set 3, the code displays a message and doesn't enable the keyboard.

```

MOV     [KeyEnable],1                assume OK.

CALL    KeySendCmd,CCMD_TEST         test controller
CALL    KeyReadData                 fetch result code
CMP     AL,055h                     is it OK?
JE      @@Ct10K

@@KbdNG:
MOV     [KeyEnable],0                keyboard failure!
CALL    ConfSendString,CON_SERIAL, \
        GDT_CONST,OFFSET cMsg_KbdNG
JMP     @@Done

@@Ct10K:
CALL    KeySendCmd,CCMD_DISABLE      disable keyboard
CALL    KeyFlushOutput              discard pending codes

CALL    KeySendDataAck,KCMD_RESET    reset & test keyboard
CMP     AL,KRSP_ACK                 accepted byte?
JNE     @@KbdNG                     nope, no keyboard
CALL    KeyReadData                 fetch result code
CMP     AL,0AAh                     is it OK?
JNE     @@KbdNG                     nope, bad hardware

CALL    KeySendDataAck,KCMD_RATE     set typematic rate
CMP     AL,KRSP_ACK                 did it work?
JNE     @@KbdNG                     ; nope, bad hardware
CALL    KeySendDataAck,DELAY_250 + RATE-30 ; yup, get smokin'

CALL    KeySendDataAck,0F0h          select scan code...
CALL    KeySendDataAck,003h          Set 3
CMP     AL,KRSP_ACK                 did it work?
JNE     @@KbdNG

CALL    KeySendDataAck,KCMD_ALL_TMB  all typematic m b
CMP     AL,KRSP_ACK                 did it work?
JNE     @@KbdNG

CALL    KeySetTypes                  set individual keys

CALL    KeySendDataAck,KCMD_ENABLE   enable key scanning

;--- install the IRQ 1 interrupt handler
CallSys CGT_MEM_SETINTGATE,I8259A_VECTOR_PM+1, \
        GDT_IDT_ALIAS, \
        GDT_CODE,<OFFSET KeyHandler>,ACC_INTGATE

;--- enable 8259 keyboard controller interrupt on IRQ1
CallSys CGT_UTIL_UNMASKIRQ,1

;--- set numlock on, update the LEDs, and (en passant)
;--- enable the keyboard

OR     [ShiftState],MASK_KEY_SH_NUMLOCK
CALL   KeyUpdateLEDs

```

Each of the routines called in Listing 7 sends tracing information to the serial port. If the keyboard can't support Scan Code Set 3 or if it reports a self-test error, you'll see a few addi-

tional messages. The self-test and keyboard response delays differ among the keyboards I've tested. I've picked default timeouts long enough to handle much worse than the worst I've seen.

CPL	Current Privilege Level
DPL	Descriptor Privilege Level
EOI	End Of Interrupt (command)
FDB	Firmware Development Board
FFTS	Firmware Furnace Task Switcher
GDT	Global Descriptor Table
GDTR	GDT Register
IBF	Input Buffer Full
IDT	Interrupt Descriptor Table
IF	Interrupt Flag
IOPL	I/O Privilege Level
LDT	Local Descriptor Table
LDTR	LDT Register
NT	Nested Task
OBF	Output Buffer Full
P bit	Present bit (in a PM descriptor)
RF	Resume Flag
RPL	Requestor Privilege Level
TF	Trap Flag
TR	Task Register
TSS	Task State Segment

The end result of all this is a sensible keyboard—each key has a unique, single-byte scan code. We can surely build something interesting from that raw material!

## RELEASE NOTES

Demo Taskette 3 now displays doublewords containing the shift state, system scan code, and character for each keystroke that produces a character. The keyboard interface routines send a torrent of tracing information to the serial port on each make and break code, exposing the inner workings (and perhaps failings) of your keyboard.

I planned to wrap up the keyboard this month, but there's more code than pages. Next month, we'll look at the interrupt handler that queues scan codes and the translation routines that convert them into familiar real-mode BIOS values. □

*Ed Nisley, as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at ed.nisley@circuitcellar.com or 74065.1363@compuserve.com.*

## IRS

422 Very Useful  
423 Moderately Useful  
424 Not Useful

# Sacrifice for the Good of the Circuit

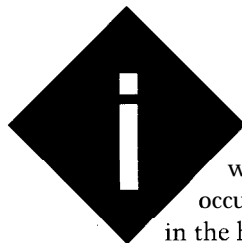
## Strengthening the Weak Link

Nothing  
beats a  
fuse for  
stopping  
overcurrent—once.

However, Jeff puts much more stock in resettable fuses. After explaining when a fuse is not a fuse, he runs us through the hoops of how to use these devices in various applications.

## FROM THE BENCH

Jeff Bachiochi



In my youth, it was a constant occurrence. Someone in the house would plug in an additional appliance and poof—we'd all be in the dark. I was always curious why this scenario always seemed to happen after daylight had faded into night.

"I must have blown a fuse," came the guilty party's cry.

"Everyone stay where you are," my Dad would say in a calming voice. He knew the routine. The most difficult part was getting to the flashlight on the pantry shelf. "I'll get the flashlight and have it fixed in a moment."

We all cringed at those words. We knew what would follow: "All right! Who took the batteries out of the flashlight?" Luckily, the darkness and our silence made a perfect cover.

Matches and candles were more prevalent back then. I think I now understand why. Burning candle in hand, Dad descended the basement stairs and searched the rows of tiny windows for the one missing link.

All the fuses looked alike—the 15s, 20s, and 30s—and could be easily interchanged. Without the correct replacement, there was a real temptation to do just that. But even temporarily, this was a real fire hazard.

Through the years, manufacturers improved the safety factor by designing screw-in fuses with different size bases. At least then, you had to use the correct size fuse. This also eliminated using the cheapest fuse replacement: the one-cent fuse.

Three cheers for the circuit breaker. Although they incur a much higher initial cost, replacements are unnecessary. Even a child can safely reset a tripped breaker without exposure to dangerous voltages.

Many electronic appliances today use protective devices—thermal and current fuses are the most popular. They are added to protect the circuitry against excessive current, voltage, and temperature. And, like the fuses which protected our homes, once they do their job, they must be replaced.

On those appliances with fuse holders, replacement is easy, although using the incorrect fuse size is still a problem. On products with fuses

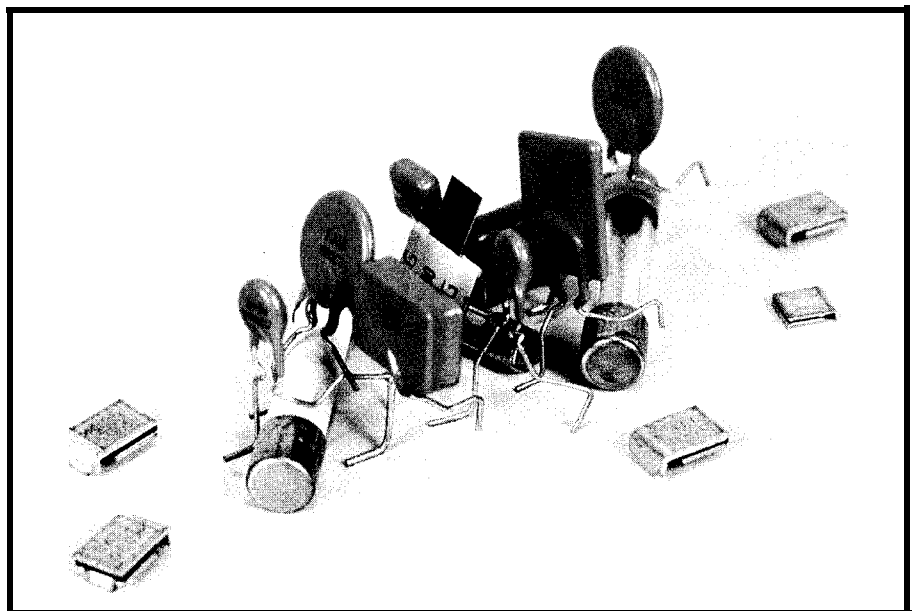


Photo 1—PTC devices are rapidly overtaking fuses on the protection race circuit

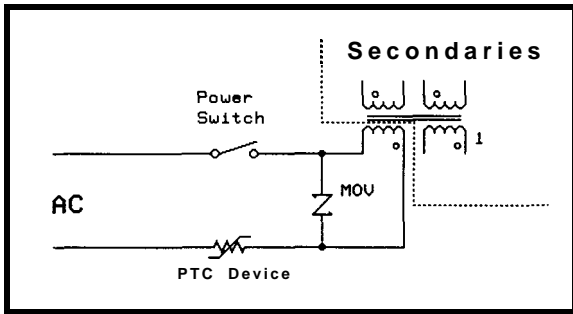


Figure 1—PTCs can replace fuses as circuit protection that doesn't require service after being tripped.

hidden somewhere inside, you're pretty much out of luck.

PTC [positive temperature coefficient) devices can change all that. Photo 1 offers a pictorial sample of the PTC takeover of the fuse.

## RECOVERABLE CIRCUIT PROTECTION

A PTC device raises its resistance in response to rising temperature. This rise can be due to increased ambient temperature or power dissipation within the device. The physical bulk of the device acts as a delay, altering its switching times. This delay enables the PTC to be unaffected by inrush, start-up, surge, and transient currents including lightning strikes.

Ceramic PTCs have been around for years. These devices range 10–200  $\Omega$  at 25°C and can rise to above 5 k $\Omega$  at trip currents. Since the full applied voltage may appear across the device, it must be rated for the maximum potential possible. Ceramic PTCs are generally limited to 100 V or less.

A new entry to the PTC market gets its protective abilities from a specially formulated composite of plastic and conductive materials. At normal temperatures, the conductive particles form low-resistance chains within the crystalline structure of the polymer. As current increases, power dissipation and temperature rises. As a result, the crystalline structure breaks into an amorphous mass, isolating the conductive particles from one another.

This isolation causes the rise in resistance responsible for limiting current flow to the circuitry. With a continuing fault, an equilibrium is established in which the limited current produces sufficient heat to keep the device in its tripped state. Once the fault is removed, the device cools and the

polymer returns to its crystalline state. The conductive particles once again form chains, reducing the device's resistance and allowing full current to pass through.

These poly PTCs service a much higher maximum working voltage (up to 600 V) in a much smaller size. Initial resistance ranges from a fraction

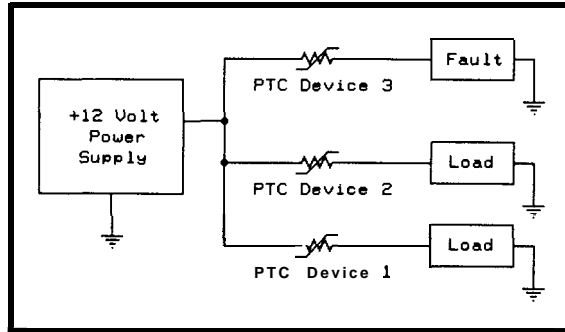


Figure 2—PTCs protect individual secondary circuit branches from one another so a failure in one won't affect the others.

of an ohm to 10 ohms at 25°C and rises to hundreds or thousands of ohms in the tripped state.

It is important to note here that, even in a tripped state, current still flows through the circuit. However, this reduced current is far below normal operating current and is based on the voltage input and power rating of the device.

Suggested applications for PTCs include instrumentation and control, security systems, medical equipment, batteries, computers and peripherals, audiovisual equipment, toys, telecommunications networks, modems, and transformers.

Let's look into a few of these to see how the PTC device aids in reducing equipment down time.

## FIRE AND SECURITY ALARM SYSTEMS

Underwriters Laboratories requires any user-replaceable fuses to be short circuited for fault-current testing. This ruling requires manufacturers to have additional calamity protection.

Under most circumstances, this extra protection involves a device (second fuse or other current-limiting device) which requires service after a short-circuit fault condition. A resettable PTC switch (see Figure 1) replaces both external and internal protection, eliminating the need for component replacement.

Distributed power for sensors and peripherals can be protected by PTC devices individually matched to their respective circuit requirements. Distributed power, although economical, causes total system failure even when the most insignificant part fails. Complete system failure can be avoided by protecting each peripheral independently. The remaining parts of the system continue operating, even when a fault exists in part of it. With the proper feedback, the system is able to intelligently report the fault instead of just grinding to a halt (see Figure 2).

Backup batteries can be easily damaged by overcurrent—both deep-cycle recharging and having to supply short-circuit currents. PTC switches interrupt short-circuit currents, preventing a potential battery explosion. Ultrathin packaging, designed especially for battery packs, enables manufacturers to replace the strap normally welded to the ends of NiCd batteries, creating total protection within the pack. While able to interrupt currents as high as 100 A, the

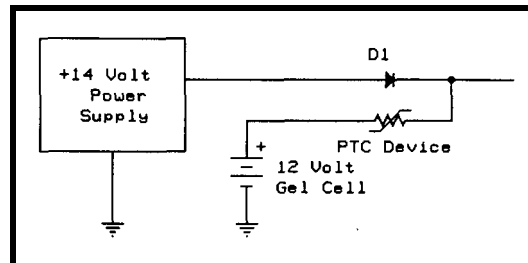


Figure 3—A PTC device limits charging current and short-circuit load current, protecting the battery from permanent damage that might otherwise occur

smallest PTC devices trip at under 3 A (see Figure 3).

## AUTOMOBILE APPLICATIONS

Power windows, door locks, and wipers all use motor and solenoid mechanisms. Without relying on a fuse, a PTC device limits current to a stalled or jammed device, preventing burnout. Stalling or jamming a device is easy enough to do. It could be caused by frozen wiper blades, an arm out the window (usually this results in needing a protective device for Mom and Dad's ears, too), or even closing a window that's already up. Individually protecting each mechanism prevents one failure from affecting other circuits sharing the same fuse.

A 1-A, 12-V DC motor might draw 5-10 times that current when stalled. A PTC device with a 1-A rating trips in less than 1 s when 8 A are passed through it.

It is important to note that wiring must be rated to survive stalled currents. Otherwise, the wire's insulation can become weakened from overheating or worse—the harnessing can become the motor's fuse.

The PTC devices drawn in Figure 4 typically interrupts up to 40 A of current without self-destructing.

## TELECOMMUNICATIONS

UL1459 requires all telco interfaces to contain protection from overcurrent which may come from downed or crossed power lines. FCC Part 68, on the other hand, requires survival from lightning-induced transients. A fault may present itself between ground and either tip or ring,

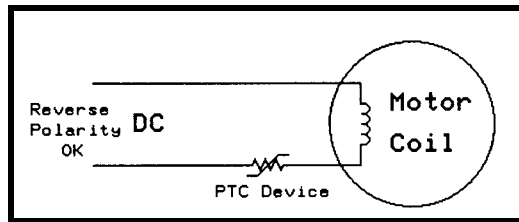


Figure 4—PTCs provide stall protection in motors without need for fuse replacement

or across the tip and ring pair. Special protection is necessary.

While MOVs are most often used across the source to limit overvoltage, they do not limit current. Series fuses offer one-shot protection. PTC switches, however, can prevent overcurrent damage and be back on-line once the fault has been corrected.

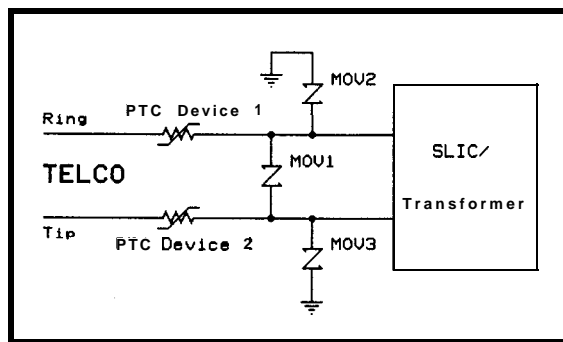


Figure 5—Telco line faults are current limited by PTCs, providing user and circuit protection.

I've designed a few telecommunication interfaces. To satisfy safety requirements, each lead (tip and ring) must have some kind of current protection. A 1-A fuse (or a 1-R resistor) suffices, but replacing either of these is a pain. The new PTC devices are perfect for fault protection here (maximum voltages of 250 and 600).

Lightning transients pass right through the PTC devices and are

absorbed by the MOVs, while line voltage faults are limited to under 300 mA by the PTC devices (see Figure 5).

## AUDIO/VIDEO APPLICATIONS

Using speakers rated below those of the amplifier, you run the risk of permanent speaker damage. PTC devices can monitor the RMS output of an amplifier and reduce current when program material crosses a maximum threshold.

When designing speaker protection, it is important to treat each parallel combination within the cabinet separately (after the crossover network). This way each speaker can be properly protected to its own rating.

Although a drop in program volume is realized when the PTC is tripped, by paralleling an LED (and properly sized series resistor) across the PTC, a visual indication of overload can be implemented.

Since the PTC devices are generally fractions of an ohm (in the untripped state) and contain no capacitive or inductive characteristics, they cause no distortion or change to the system's sound characteristics (though I'm sure some purists

would debate me on this one) under normal operating conditions (see Figure 6).

## DESIGNING WITH THE PTC SWITCH

The most important parameter to look at when using a PTC device is the absolute maximum voltage the device might ever see. In most situations, the maximum may be 15-30 V. In line fault conditions, it could be 110 or 220 V. New polymeric PTC devices can withstand up to 600 volts.

The next parameters have to do with circuit current: the maximum operating (hold) current, the minimum protecting (trip) current, and the maximum interrupting (max) current. For most PTC devices, trip current is simply twice the hold current. The hold current determines the physical size of the device, whereas max current determines the type used.

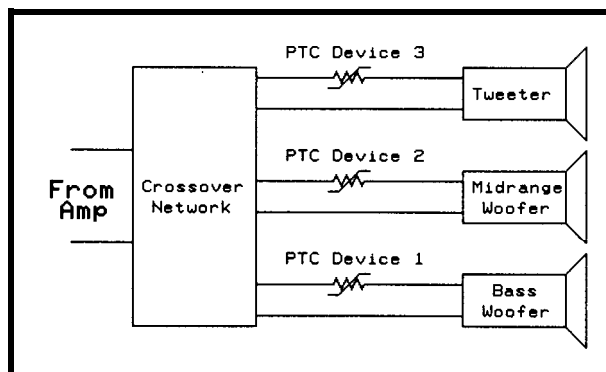


Figure 6—Each audio output device is individually limited to its maximum rating without distortion.

All of this is based on temperature—an ambient temperature of 20°C. As the ambient temperature falls, the hold and trip currents increase. Conversely, as the ambient temperature rises, the hold and trip currents decrease. This decrease is fairly consistent: 1% per °C. For instance, when the ambient temperature is 70°C (i.e., 50°C hotter), a PTC device rated to trip at 2 A trips at 1 A (i.e., 50% of 2 A).

Normal resistance for untripped PTC devices ranges from 0.05 to 20 Ω. The lower-voltage PTC devices have the lowest nominal resistances. Once tripped, the resistance is much higher and can be determined by:

$$\frac{V^2}{P_d}$$

where  $V^2$  is the applied voltage squared and  $P_d$  is the device's power dissipation.

Typically,  $P_d$  ranges from 0.5 to 4 W, depending on the device. Take, for example, a 0.1-Ω, 1-W PTC device that is dropping 12 V and has been tripped

by a 3.7-A current. The device has a tripped resistance of 12<sup>2</sup>/1 or 144 Ω, limiting the current to about 80 mA.

The PTC device remains in the tripped state as long as the fault remains. In this state, the limited current is sufficient to keep power dissipation at a high enough level to keep the device tripped. Only removal of the fault condition enables the PTC device to cool enough to recrystallize to its low-resistance arrangement.

### SO WHAT'LL IT BE?

In many applications, the designer has multiple solutions for current protection. Device selection is based on a combination of price and size. PTC devices offer small size, UL recognition, and automatic resetting. The nuisance of having to replace blown fuses alone makes this an easy choice. Quantity pricing ranges from 50¢ to \$1. □

*Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on Circuit Cellar INK's engineering staff.*

*His background includes product design and manufacturing. He may be reached at [jeff.bachiochi@circellar.com](mailto:jeff.bachiochi@circellar.com).*

### SOURCES

Poly PTC  
Raychem Corporation  
Polyswitch Division  
300 Constitution Dr.  
Menlo Park, CA 94025-1 164  
(415) 361-3333  
Fax: (415) 361-7667

Ceramic PTC  
Midwest Components, Inc.  
P.O. Box 787  
1981 Port City Blvd.  
Muskegon, MI 49443  
(616) 777-2602  
Fax: (616) 773-4307

### IRS


425 Very Useful  
426 Moderately Useful  
427 Not Useful

**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 8051/803 1 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 compatible
- Compatible with all 8051 variants
- **BXC51 \$ 295.**

508-369-9556  
FAX 508-369-9549

Binary Technology, Inc.  
P.O. Box 541 • Carlisle, MA 01741



**\$79** Micro Genius™  
QTY 1

Smaller than a credit card and C-programmable! Ask for our free Dynamic C™ demo disk. Call our AutoFAX 916-753-0618 from your FAX. Request catalog #18.

RS232/RS485  
Flash EPROM  
Power Supply  
Real-time clock  
Parallel I/O



1724 Picasso Ave.  
Davis, CA 95616  
916.757.3737  
916.753.5141 FAX

**ZWORLD**  
ENGINEERING

# MIPS for the Masses

## SILICON UPDATE

Tom Cantrell



he RISC versus CISC war rages on in the computing market, though the

technical arguments (on which there is a surprising amount of agreement underneath the fervent rhetoric) have largely been supplanted by marketing and software issues.

As the skirmish spreads into the embedded arena, it's important to note that the battlefield is somewhat different. While marketing and software are still important, technical issues such as price, performance, and power consumption remain dominant. Also, without the compatibility chain around their neck, embedded designers are free to choose whichever chip best suits their needs.

Traditionally, embedded RISCs break into two camps. The first includes stripped-down versions of desktop RISCs such as PowerPC, MIPS, and Sparc. The second group hinges on chips not designed for

desktop use (or at least they never were marketed as such) like the '960, 29k, and ARM.

Generally, the latter, not dizzied by the vapors of the high-end performance-at-any-price mentality, have converged most rapidly on the true needs of embedded applications (which is why I've covered them in previous articles).

As for the former, I'll admit having a few laughs at the expense of those purporting to sell (or claiming anyone needs): an embedded mainframe, "low-power" chips featuring a 10-minute battery life, "low-cost" chips with three-digit price tags, "easy-to-use" chips calling for a sea of TTL and PAL glue logic.... Ho-Ho!

However, the only thing that doesn't change in the IC business is the fact that everything changes. Inexorable market and technology trends require continual reevaluation of conventional wisdom.

Thus, now is the time to take a reality check on the embedded-RISC concept. Though the pitch is still a bit rocky, you may be surprised to find that yesterday's high-end RISCs may make sense in tomorrow's low-end applications.

### THE BIG GAME

The persistence of Silicon Valley in the face of a variety of challenges owes a lot to the talent pool fed by Stanford and Berkeley. Drifting in a fertile sea of venture capital, innovations and innovators continue to spawn.

The historic rivalry between the two comes to a peak at the "Big Game" every year, accompanied by the requisite frenzy and folly of each side's boosters. Who can forget the time the Stanford band ran onto the field prematurely, quite effectively blocking their own team as Berkeley ran in a last-second winning touchdown? The fact that recent tryouts for team mascot have resorted to setting themselves on fire shows how hot emotions can get.

Naturally, the rivalry extends from the football field into the lab (though no one's set themselves on fire—yet) with Berkeley originally



It was only a matter of time before

traditional workstation-style RISC processors started making the move toward the embedded arena. Tom checks out the latest MIPS offering.

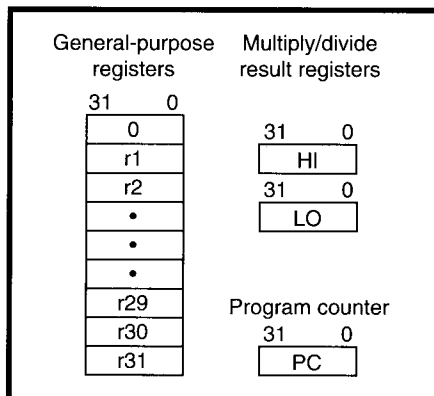


Figure 1—Since all RISC ops (except load and store) work only between registers, the MIPS architecture offers a large register set. Separate HI/LO multiply and divide result registers let other instructions proceed in parallel without stalling the CPU.

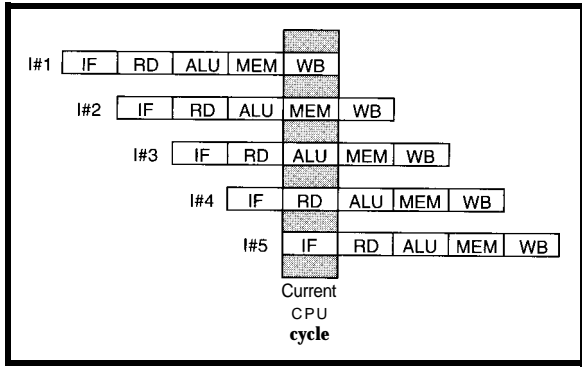


Figure 2—The MIPS R3000 pipeline consists of five stages: instruction fetch (IF), register file read (RD), execution (ALU), data load/store (MEM), and register file write (WB). Data (load) and control (branch) hazards are handled by the compiler (delay slots) rather than interlock hardware (stalls).

fielding what has become the Sun Sparc architecture and Stanford responding with the SGI-blessed MIPS. Competition between the two has been and continues to be intense, resulting in incredibly powerful chips as each side fights for bragging rights.

While most of the fireworks take place in the ethereal world of workstations, I've particularly noticed that the MIPS camp seems to be making slow but sure progress toward the embedded market. Most striking is the openness of the MIPS architecture with an ever-growing lineup of suppliers including IDT, LSI Logic, NEC, Toshiba, Siemens, Sony, and Philips. Indeed, the sun never sets on the MIPS empire.

The era of plug-compatible second-source CPUs is long gone. It never seemed to work for either the source or sourcee-witness the Intel and AMD debacle and the short-lived 68k Alliance of the early '80s. As far as I know, there are no multivendor clones for embedded RISCs.

Nevertheless, the MIPS bandwagon does seem to guarantee a degree of competition, if not at the socket, at least at the architectural level. Also, compared to purely sole-sourced designs, the MIPS camp can offer both wider and finer coverage of the price-performance spectrum.

A notable feature of the MIPS alliance (shared only by the ARM) is that unlike other open (i.e., available to license) architectures, sourcees needn't worry about running up against their source in the purchasing lobby. Neither MIPS (now part of SGI) nor ARM manufactures or sells chips, defusing the love-hate relationship that always seems to drive other CPU marriages of convenience to

the divorce court.

### RISC101

It's somewhat ironic, given the crosstown rivalry, that the captains of each team (John Hennessy at Stanford and David Patterson at Berkeley) have jointly authored some of the best books on computer architecture including *Quantitative Analysis of Computer Architecture* and *Computer Organization and Design*. If you want to understand the motivations and ideas behind the RISC revolution, check them out.

In my opinion, the original (circa mid-'80s) MIPS R3000 CPU core is arguably one of the best 32-bit integer CPU cores around—thanks to close

cooperation between the hardware (CPU) and software (C compiler) designers. Though the workstation world has moved on to fancier superscalar and speculative execution techniques, I doubt they match the R3000's bang per buck.

Though other RISC ideas have blurred in the smoke and mirrors of marketing battles, load-and-store architecture remains the red badge of RISCiness. The term refers to the fact that instructions operate only on registers, not memory. Naturally, this scheme calls for lots of registers (32 x 32 bit), dedicated high and low registers for multiply and divide, and a PC (Figure 1).

With the exception of load and store, nearly every instruction executes in a single cycle, including multibit shifts and rotates—thanks to an on-chip barrel shifter. The only notable exceptions are multiply (32 x 32 with 64-bit result) and divide (32 ÷ 32 with 32-bit quotient and 32-bit remainder), which take 12 and 35 clock cycles, respectively.

To speed those clock cycles, pipelining is a must. The million-Hz question is, "How many stages?" Too few and the opportunity for high clock rates is left on the table. Too many and the pain from increased complexity starts to exceed the gain (which itself diminishes due to the dreaded conditional branch]. The 5-stage MIPS pipeline of Figure 2 seems just about right. It's longer than most, but not too long.

To further simplify (and thus speed] the pipeline, the MIPS architecture dismisses hardware interlocks (i.e., stalls) for handling certain pipeline hazards in favor of compiler scheduling. Load and branch instructions are followed by a delay slot, which the compiler attempts (successfully about half the time) to fill with useful instructions.

Even those pesky loads and stores can often be dispatched in a single

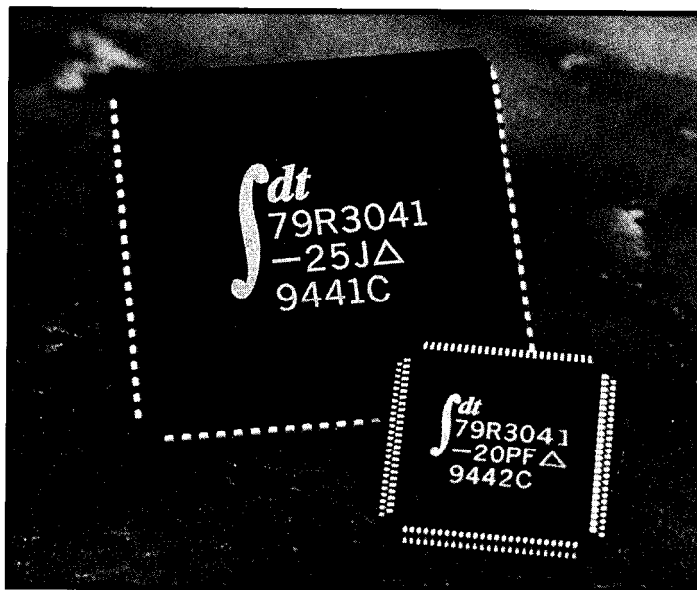


Photo 1—The IDT 3041 packs workstation performance into a 100-pin TQFP package smaller than most 8-bit micros. It's also available in an 84-pin PLCC package that is plug-compatible with more powerful members of the IDT 30xx family.

cycle, thanks to on-chip cache. A key feature of the MIPS architecture is Harvard cache organization (i.e., separate instruction and data caches can be accessed simultaneously). In furtherance of the need for speed, MIPS traditionally relies on simple, direct-mapped, write-through caches.

In response to the baroque, arbitrary instruction length of CISCs, MIPS adheres to a brutally simple fixed 32-bit instruction format (see Figure 3). There's some validity to the argument that code density suffers from such a scheme. However, it's my experience that the effect diminishes with the size and 32-bitness (i.e., >64-KB programs with long integers and pointers to big memory) of the application. In fact, the code-density issue may be hidden in the ever-increasing capacity and coarser granularity of practical memory setups.

## BIG RISC IN A SMALL PACKAGE

One of the most embeddable MIPS chips available today is the 3041 from Integrated Device Technology. It combines an R3000 integer core with 2-KB instruction and 512-byte data caches in a tiny 100-pin TQFP (Thin Quad Flat Pack) plastic package (see Photo 1).

The 3041 is the entry point to a complete lineup of 99% plug-compatible parts that includes versions with larger caches (e.g., 3051 and 3052), TLB (translation lookaside buffer) for virtual memory (E versions), and built-in IEEE floating point (308 1). However, the 3041 is clearly the leanest and meanest of the bunch, and includes specific features for low-end applications.

Thanks to smallish cache size and derated clock (only up to 33 MHz!), the 3041 is the lowest-power family member. Full-speed, 5-V operation calls for almost 2 W, which is still kind of hot for most embedded

applications, though nothing compared to the latest desktop chips. However, the truly power conscious can hop on the 3.3-V bandwagon, which cuts power to a more acceptable 0.5 W or so.

Getting full 32-bit addresses and data into and out of such a tiny chip does call for some compromise, namely a multiplexed address/data bus (see Figure 4). Besides address latches, tight bus timing typically dictates the use of data transceivers to avoid bus contention as well. However, the 3041 uniquely offers slow bus turnaround and extended address-hold timing options, giving the designer the option of dispensing with the transceivers at the expense of some performance.

Further reflecting the realities of the embedded world, the 3041 has programmable bus width (8-, 16-, and 32-bit) capability. Notably, this feature enables use of a single x8 boot (EP)ROM rather than the parade of four chips that accompanies 32-bit-only RISCs.

When it comes to the Big Endian versus Little Endian controversy, the 3041 says have it your way, supporting not only selection at reset, but even

dynamic switching between modes. Needless to say, the latter practice could lead to some rather interesting debugging sessions, so take care.

The 3041 even includes an on-chip 24-bit timer that's a likely candidate for a variety of timing duties such as triggering DRAM refresh or generating an RTOS tic. The single output line can be configured to pulse or latch low on timeout.

## QUIT STALLING

When things are going smoothly, the 3041 cranks an instruction through the pipeline on every clock. Except for overtly uncacheable references (such as I/O) and a multiply or divide from time to time, the only things that stand in the way of 1-CPI [clock per instruction] nirvana are cache misses and writes (remember the data cache is write through).

To deal with the latter, the 3041 includes a four-deep write capture buffer. The buffer can hold up to four pending writes captured at the processor rate. The information is then stored in slower external memory later. In the meantime, the processor continues to execute from cache.

Thus, the write penalty is largely hidden, though an in-order memory-update policy does dictate that a cache miss stalls until the write buffer drains.

Instruction-cache misses take advantage of DRAM burst mode to speed refill. Given the locality of code, the I-cache line size is four words, meaning the 3041 always loads four words sequentially in response to an I-cache miss. *Streaming* is also supported, which means that the processor exits the stall as soon as the needed word appears, even as the cache continues to refill.

Since data usually exhibits less spatial locality than code, the data-cache line size is one

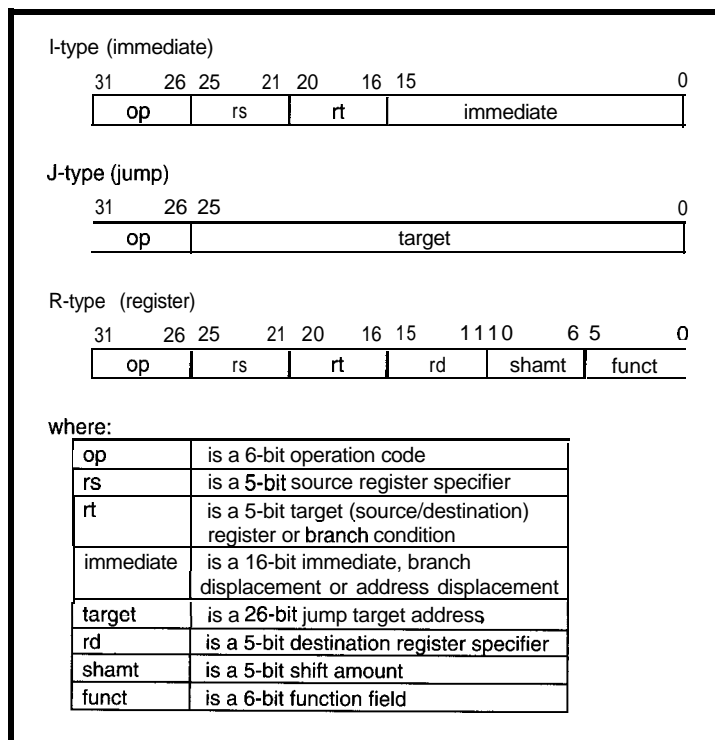


Figure 3—The MIPS instruction set strictly adheres to the RISC fixed-length (32-bit) instruction tenet. Furthermore, besides loads and stores that miss the cache, every instruction (except multiply and divide) executes in a single clock.



(i.e., only a single word need be loaded to satisfy a miss). Nevertheless, sequential data operations (such as array, buffer, or bitmap accesses) can take advantage of a four-word data-cache refill option. Indeed, the refill-size selection can be made dynamically, giving the programmer overt choice in the matter.

Burst refill places demands on the system designer to drive two inputs: RdCEn\* (Read Clock Enable) and Ack\* (Acknowledge). The former signals the availability of each word in the burst, while the latter signals the end of the transfer.

You may wonder why both are needed after all, the CPU can certainly count the number of RdCEn\*s to determine when the cycle is finished and generate its own Ack\*. Indeed, the 3041 offers this option, thereby simplifying the required memory interface logic.

However, true speed freaks can take advantage of overt Ack\* control to wring a little more parallelism out of the bus interface. RdCEn\* actually clocks data into a read buffer while Ack\* dumps it into the cache and releases the stall. As it turns out, you can treat Ack\* as a promise to deliver the data on time and issue it early, starting the cache transfer and possibly ending the stall even before the last word is delivered from memory.

## EXCEPTING THE INEVITABLE

One common complaint about workstation-like

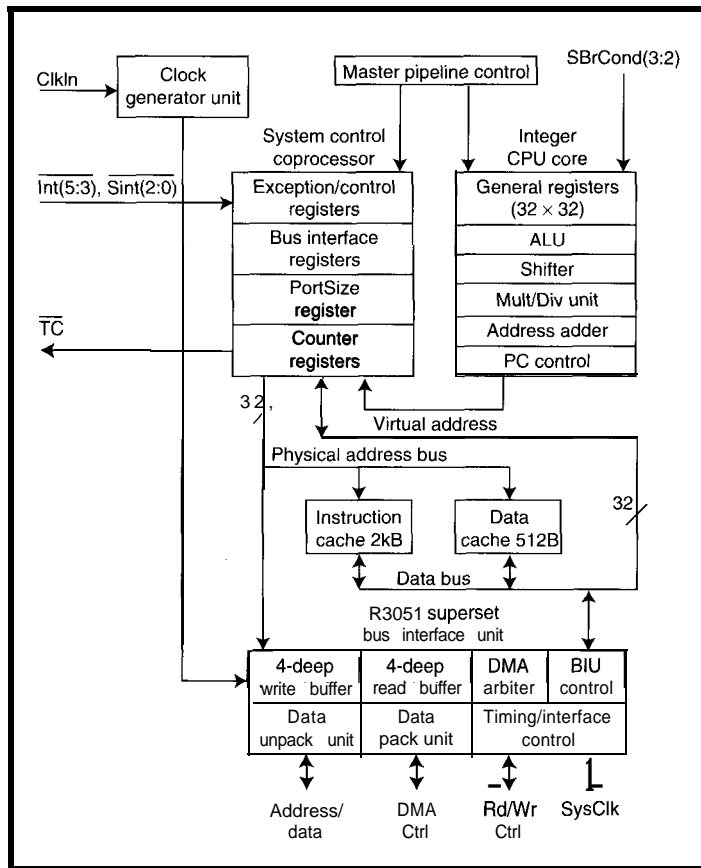


Figure 4—The IDT 3041 combines a MIPS R3000 core with 2-KB instruction and 512-byte data caches. To cut pin count, a multiplexed address/data bus is used.

RISCs is their cavalier attitude toward interrupts. I must say, at first glance, it's tempting to criticize the MIPS architecture in this regard.

First, almost all exceptions—address and bus error, SYSCALL, breakpoint, unimplemented instruction, overflow-not to mention the six

hardware and two software interrupts are mushed together and funneled through a single exception vector. It's up to the programmer to examine the so-called CAUSE register to figure out what's going on.

The 3041 stacks nothing in memory in response to an exception. Instead, it just shoves the old PC into a register (EPC) and leaves the rest to you. Sure, you don't have to stack the PC, but be real careful that your exception handler doesn't itself generate an exception-at which point you're toast.

Reflecting the true spirit of RISC, support for interrupt priority is rather reduced. In fact, there isn't any! An exception simply turns off all interrupts, and if you want to handle nested or prioritized

interrupts, feel free to diddle with the individual and global interrupt-enable bits.

They even forgot to provide a real NMI\* (i.e., errant software can disable all interrupts). Oh well, the '51 has gotten by for many years without one, so it must not be that big a deal.

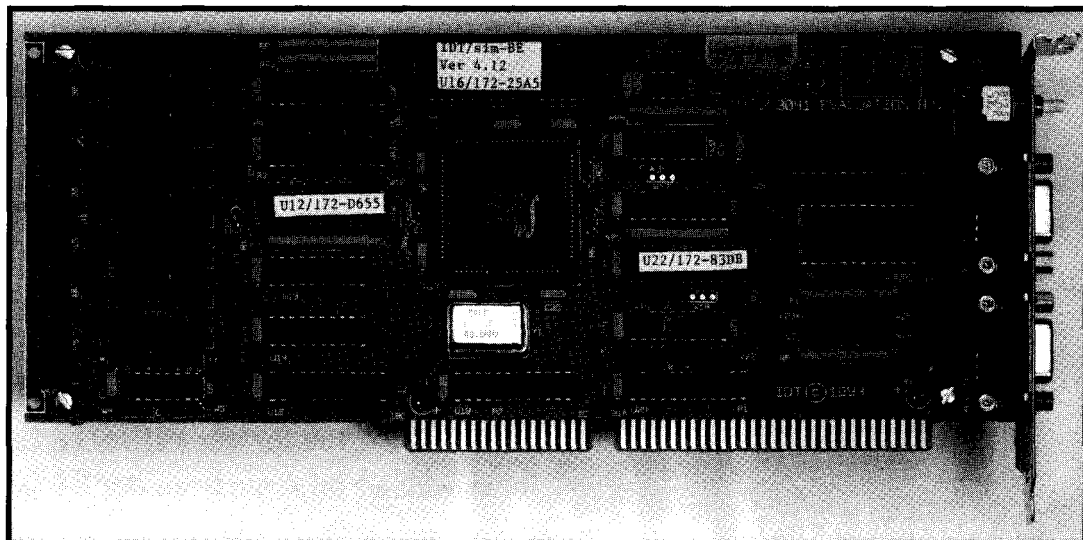


Photo 2—The 3041 Evaluation Board packs a 20-MHz 3041, 128 KB of EPROM (with the IDT/SIM monitor), and up to 4 MB of DRAM into a few dozen chips. Communication with a PC host takes place via either the PC/AT bus (bottom edge) or RS-232 serial ports (right edge).

Figure 5—As the 3041 evaluation board shows, designing in a workstation-class RISC is easier than ever. Compared to earlier MIPS chips, the amount of glue logic required is significantly reduced.

It all sounds rather grim, but actually I've become less critical as time goes on—so what if it takes a few dozen instructions to clean up after interrupts? After all, that's only about a microsecond or so.

Furthermore, I kind of like the idea that I can implement any darn interrupt scheme (priority, nesting, how much gets stacked, etc.) I want without stumbling over someone else's preconceived notions. And as for NMI\*, I imagine there are ways to fall back on other exceptions (perhaps bus error or, in a pinch, there's always RESET).

## LESS FILLING, TASTES GREAT

Photo 2 shows the IDT79S341 Evaluation Kit for the 3041. Any of you who have designed with workstation-class RISC chips must be impressed with remarkable downsizing. In fact, thanks to 3041 streamlining features like the slow bus turn (i.e., no data buffers) and programmable bus width (one EPROM instead of four), the 3041 board achieves basically the same specs as IDT's earlier 3051 board with half the chips. It's nice that, when it comes to recognizing the realities of embedded applications, the RISC folks are finally starting to get it.

Figure 5 shows the block diagram of the board. Notice the tiny amount of glue logic required—little more than address latches, DRAM address mux, and a memory-control PAL.

If the form factor doesn't give it away, communication with the board can take place over the PC/AT bus (for highest speed downloading) or via the more conventional serial port.

The ROM contains IDT/SIM, a rather full-featured monitor. Besides the usual commands (memory dump, go, etc.), it includes a quick-and-dirty assembler/disassembler (great for

fiddling with small routines without hauling out the entire arsenal of tools) and the commands needed to deal with the cache (read, write, flush, etc.). It even offers a number of runtime entry points (such as C-like `printf`, `putchar`, `strcpy`, etc.) that you can wrap your code around.

The package also includes MicroMonitor, a minimal kernel for debugging your own design. All you need to run MicroMonitor is the CPU, EPROM, and a serial port. It's just the thing for debugging your DRAM interface, the area that usually causes the most head scratching.

The latest version of the ROM also includes an ICE-like trace facility that tracks execution based on qualifiers such as instruction, data read, data write, memory range, and so on. Sure it isn't real time, but it is real cheap!

The budget-conscious will be further pleased to find that the evaluation kit includes a complete DOS-based cross-development toolset from the increasingly popular GNU technology. It consists of an ANSI-compliant compiler (GCC), an assembler (GAS), linker (GLD), and debugger (GDB). The debugger works with IDT/SIM to offer remote (i.e., via the PC/AT bus or serial port) C source-level debugging.

Besides the standard C runtime library and a variety of example routines (cache initialization, exception handling, etc.), a key extra is a floating-point library.

Traditionally, floating point on integer MIPS chips has been handled by emulating the original MIPS FPU

(the 3010) using software traps. The good thing about such a scheme is that a single binary can run transparently on a system with or without FPU.

However, truly mimicking the FPU is quite messy and slow since even minor operations not only incur trap overhead, but have to save and restore a zillion FPU registers and intricately fake a variety of status and control bits. So, IDT provides a floating-point math library that dispenses with FPU emulation in favor of dedicated code that runs as much as 10 times faster.

If it isn't plain to see that the era of 32-bit embedded RISC is close at hand, let me spell it out for you. The EV kit is only \$795 and 3041s start at \$9.60. If you're ready for life in the fast lane, hang on to your logic probe and hitch a ride on a MIPS chip. □

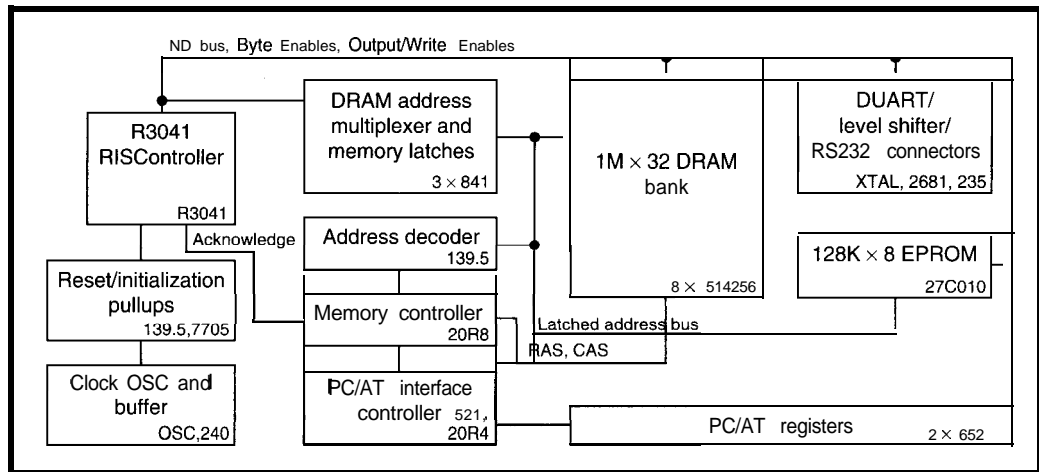
*Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached at (510) 657-0264 or by fax at (510) 657-5441.*

## CONTACT

Integrated Device Technology, Inc.  
2975 Stender Way  
Santa Clara, CA 95054-3090  
(408) 727-6116  
Fax: (408) 492-8674

## I R S

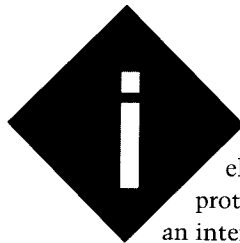
428 Very Useful  
429 Moderately Useful  
430 Not Useful



# Circuit Protection

## EMBEDDED TECHNIQUES

John Dybowski



In principle, electrical circuit protection provides an intentionally weak link at a circuit's key point. If the current exceeds what the weak link can bear, it breaks down, protecting the circuit from excess current.

Generally, the condition guarded against is overcurrent, although overvoltage and thermal-fault conditions can also be targeted. Protection can be system-wide (i.e., complete electrical assemblies are protected) or selective (i.e., protection is added to an electrically delicate subsection).

Most commonly, protection is provided with a fuse that melts under specified overload conditions. The fuse's disintegration can be invisible or spectacular, depending on a number of factors. Fundamentally, however, for a fuse to provide effective circuit protection, a number of operational parameters must be understood and specified.

Rather than inundate you with data that may be of little immediate use, let me touch on some more important points of effective circuit protection. Once I've covered these subtleties, I'll end by looking at an active electronic fuse. The capabilities of this device are more interesting in light of the preliminary information.

### FUSE PARAMETERS

#### • Speed

Perhaps the most notable parameter of a fuse is how rapidly it reacts to various

current overloads. Three general categories exist: very fast acting, fast acting, and slow blowing.

Very-fast-acting fuses do not tolerate even short-duration overcurrent and are frequently used for extremely sensitive circuitry. Fast-acting fuses are not as unforgiving and are appropriate for brief overcurrent conditions. Slow-blowing fuses have extra thermal inertia. They tolerate the heavy inrush current common to a variety of electrical devices, especially those using large filter capacitors.

#### • Rated Current

As a general rule, fuses should not be operated at more than 75% of their nominal capacity at 25°C (fuses are temperature sensitive). Their ratings are established at an ambient temperature of 25°C. The internal temperature generated by the current passing through the fuse increases or decreases with ambient temperature change. Observing a fuse's current rating is extremely important since excessive surges have a cumulative effect.

#### • Shorting Current

In addition to a fuse's nominal current-carrying capacity, there is a short-circuit specification. This parameter defines the maximum current that a fuse can safely interrupt at rated voltage. During a short-circuit fault, a

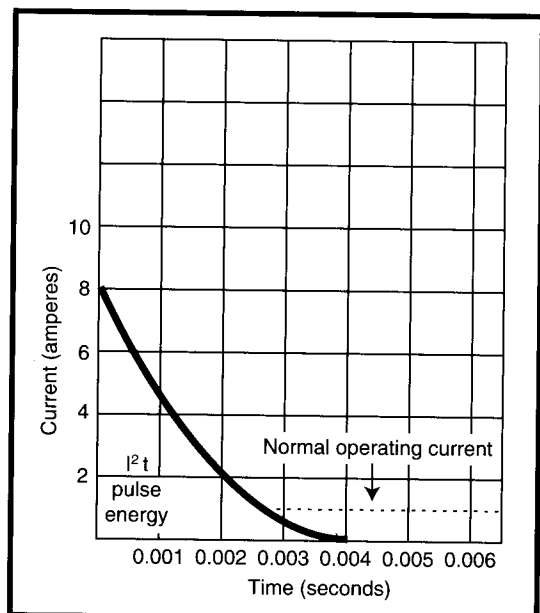


Figure 1—A typical inrush current can be many times the nominal current.

John picks up where Jeff

leaves off with other popular forms of circuit protection. He combines various active and passive solutions—from the wire that melts to something managing power overload actively.

fuse may receive an instantaneous overload current many times greater than its normal operating current. The short-circuit parameter (also called the *interrupt rating*) sets the upper limit of current that the fuse can safely handle. If this limit is not exceeded, the fuse remains intact and clears the circuit.

• Temperature

Most slow-blow fuses use materials that melt at lower temperatures than fast-acting and very-fast-acting fuses. As a result, they are more sensitive to changes in ambient temperature.

• Resistance

An often neglected fuse parameter is its resistance since, in many cases, resistance is not significant at all. However, with low-voltage applications served by fractional-ampere fuses, you may no longer be able to disregard this parameter.

Values of several ohms are common. Most fuses are manufactured from materials that have positive temperature coefficients, so it is common to give cold and hot resistances (voltage drop at current). As you'd expect, operation typically falls somewhere in between. Cold resistance is the resistance obtained using a measuring current of no more than 10% of the fuse's nominal rated current while hot resistance is calculated from the stabilized voltage drop across the fuse with current equal to the normal rated current.

• Voltage

A fuse's voltage rating indicates that it can safely interrupt (i.e., no rupture, explosion, or other undesirable side effects) a voltage that does not exceed its rating. Since fuses are sensitive to current and not voltage, this parameter usually indicates the safe upper limit for preventing unanticipated related events.

Under certain conditions, it is acceptable to exceed the voltage rating.

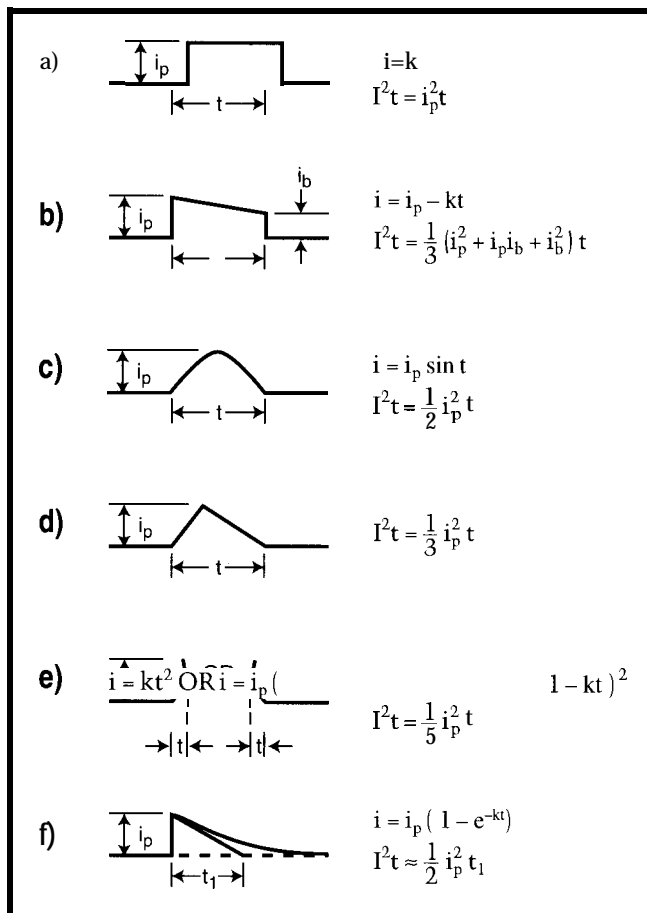


Figure 2—Here's a representative collection of transient pulse current waveforms and their associated equations for calculating pulse Pt.

For example, electronic equipment with relatively low-output power supplies have circuit impedances which limit short-circuit currents to 10 times the current rating of the fuse. In such a case, it is common to specify fuses with 125- or 250-V ratings for secondary circuit protection of 500 V or higher.

To clarify, a fuse may always be used in a circuit with voltages less than its rated voltage. It may also be used with voltages higher than its rated voltage if the maximum power level the fuse is subjected to under short-circuit conditions only produces a low-energy, nondestructive arc.

• Nominal Melting:  $I^2t$

Tests are performed on a fuse to determine the amount of energy required to melt the fusing element. This energy is described as the *nominal melting  $I^2t$*  and is expressed as  $A^2s$  (ampere squared seconds).

To determine this parameter, the time is measured for a pulse of current

applied to the fuse to cause melting. If melting does not occur within 8 ms or less, the pulse current is increased. This test is repeated until the fuse melts within the specified time frame. The stipulation of 8 ms for melting guarantees that the heat does not have sufficient time to be thermally conducted from the fuse element.

PICK ONE

Fuse selection involves a survey of the operational conditions surrounding the electrical system. These conditions could include:

- normal operating current—assuming operation at 25°C, the current rating of the fuse would be derated 25 % typically.
- AC or DC application voltage—except for special conditions, the fuse's voltage rating should be equal to or greater than the available circuit voltage.

- ambient temperature—the higher the ambient temperature, the hotter the fuse operates. A fuse also runs hotter as the operating current approaches the fuse's rating. However, a fuse should last indefinitely if operated at room temperature at no more than 75% of its nominal rating.
- overload current and overload time—these are the parameters fault protection is specified for. In other words, fault conditions can be specified in terms of the current or of the current and time till damage.
- maximum fault current—the maximum fault current should fall below the fuse's maximum short-circuit specification to ensure safe operation.
- inrush and pulse current—inrush or pulse current encompasses the broad category of wave shapes that includes inrush and start-up currents, surge currents, and other current transients. Pulse currents produce thermal cycling that affect the life of a fuse.

Since start-up inrush currents are unavoidable in a variety of electrical circuits, thermal delay fuses, such as the slow-blow type, are designed to help them survive such pulse phenomena while still providing adequate protection during continued fault conditions.

In any case, start-up pulses should be defined and compared to the  $I^2t$  rating of a fuse. Once this rating is known, the resulting wave shape should be compared to the fuse manufacturer's time-current curve. Combining this information with normal operating currents, derating, and ambient operating temperature gives the selection criteria for the fuse.

For example, suppose a fast-acting fuse is required for a given application. Assume the fuse must be capable of withstanding 100,000 current pulses of the wave shape shown in Figure 1. The ambient operating temperature is 25°C and the nominal operating current is 0.75 A.

Reference to the fuse manufacturer's wave shape table shown in Figure 2 reveals that wave shape (e) should be used for the subsequent calculations. The peak pulse current ( $I_p$ ) and time (t) yields the following calculation:

$$\begin{aligned} I^2t &= \frac{1}{5} I_p^2 t \\ &= \frac{1}{5} \times 8^2 \times 0.004 \\ &= 0.0512 \text{ A}^2\text{s} \end{aligned}$$

This value yields the pulse  $I^2t$ .

Figure 3 indicates that a figure of 22% should be used for a value of 100,000 occurrences of the calculated pulse  $I^2t$ . This  $I^2t$  pulse is now converted to the required value of nominal melting  $I^2t$ :

$$\begin{aligned} \text{Nominal Melt } I^2t &= \text{Pulse } I^2t \times \frac{1}{0.22} \\ &= \frac{0.0512}{0.22} \\ &= 0.2327 \text{ A}^2\text{s} \end{aligned}$$

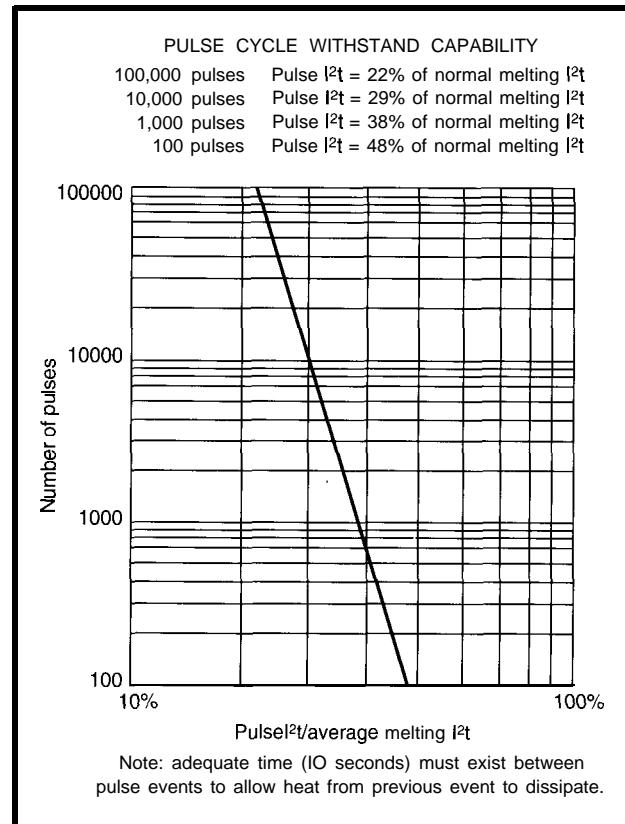


Figure 3-The effects of pulse-current phenomena must be considered because they are cumulative.

Examining the manufacturer's  $I^2t$  rating data for the Littlefuse Pico II 125-V very-fast-acting fuse reveals that the 1-A 2510001 part is rated at 0.281  $A^2s$ , which is the minimum rating that satisfies the calculated 0.2327- $A^2s$  value. This fuse also meets the 0.75-A nominal current criteria when derated by 25% for operation at an ambient temperature of 25°C.

## ELECTRONIC BREAKING

Undoubtedly, fusible links are the most common method of providing reliable, inexpensive circuit protection. However, a number of alternative technologies (most notably mechanical circuit breakers) are an advantage when your needs are more specialized.

With most of these devices, their various operating parameters (melting, thermal inertia, resistance, etc.) are linked by their manufacturing process. When precise control of the individual parameters is necessary, it's best to combine various active electrical components.

Linear Technology's LTC 1153 Auto-Reset Electronic Circuit Breaker

provides highly integrated precise protection. This 8-pin IC drives a low-cost N-channel MOSFET that supplies (or interrupts) current to an external electronic load. The trip current, trip delay, and autoreset period are programmable and can operate over a wide range of values.

An external PTC thermistor can be added when thermal faults must be guarded against. As a bonus, a digital on/off pin enables external logic to control power to the external load. An open-drain status pin pulls low to indicate that the electronic breaker has tripped and that the protected system is in fault status.

Figure 4 shows the LTC 1153 in block form. From this, you can see that the IC contains a built-in voltage regulator, current-sense amplifier, various timing circuits, and an N-channel

MOSFET charge pump.

Certainly, these elements are familiar to engineers. As is often the case, it's the way these capabilities are combined in a single piece of silicon that makes the device unique and useful. To gain an appreciation for the device's refinement, let me briefly touch on the design's various function blocks.

- input and shutdown pins-The LTC 1153 has an active-high control pin that activates all protection and charge-pump circuitry when asserted. The shutdown pin breaks the circuit when used in conjunction with a PTC thermistor if a secondary overtemperature fault is detected. These pins switch at 1.3 V with about 100 mV of hysteresis and, as a result, can handle a variety of logic family outputs.
- auto-reset timer-Using an external timing capacitor, a ramp voltage is generated each time a fault condition is detected. When the timing ramp reaches 2.5 V, the switch is turned back on and the capacitor is

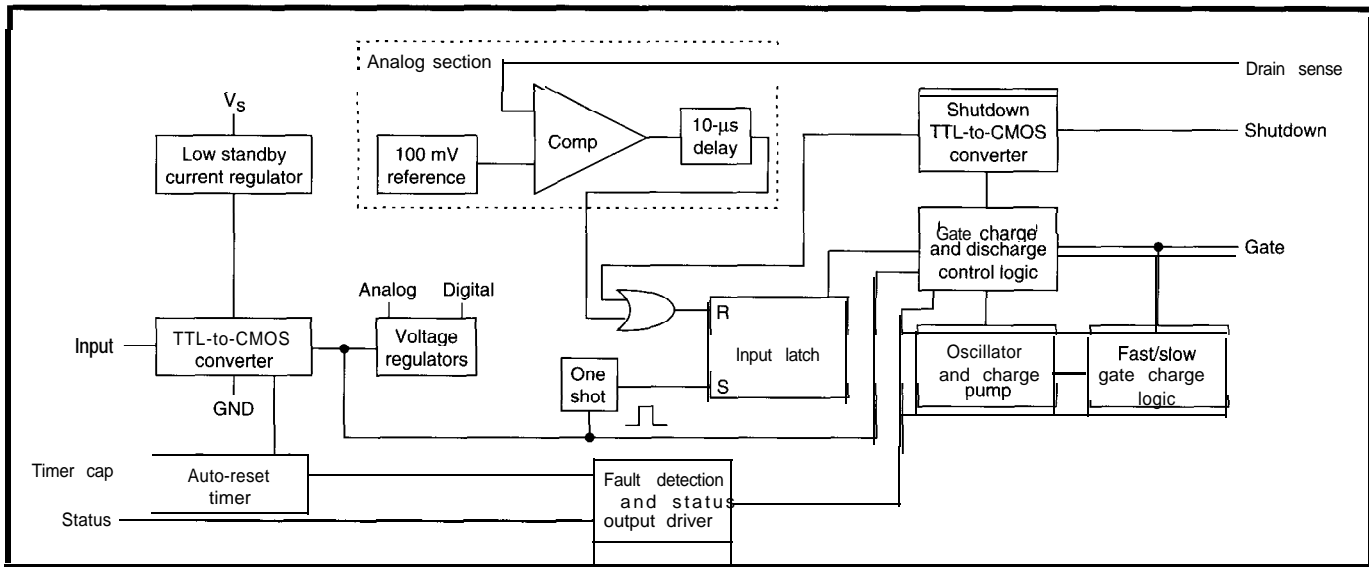


Figure 4—The LTC1153 Electronic Circuit Breaker is a unique combination of standard functions.

discharged. This can be used to automatically reset the breaker after a fixed fault duration.

- internal voltage regulator—The output of the internal TTL-to-CMOS converter drives the regulated supply which powers the low-voltage CMOS logic and analog blocks. The

outputs of the regulator are isolated so that noise generated by the charge pump does not couple into the 100-mV reference or the analog-comparator section.

- gate charge pump—The gate drive for the N-channel MOSFET switch is generated by an adaptive charge-

pump circuit. The resulting gate voltage is substantially higher than the power-supply voltage, enabling the designer to use an inexpensive external N-channel device as a high-side switch. The charge-pump capacitors are included on-chip to further save space.

- drain current sense—Current flowing into the drain of the N-channel MOSFET switch is monitored using a low-value external sense resistor. The voltage drop across this resistor is compared to an internal 100-mV reference. If the voltage drop exceeds 100 mV, the input latch is reset and the gate drive discharged.
- controlled gate rise and fall times—When the input is switched on and off, the gate is charged and discharged by the internal charge pump in a controlled manner. The charge and discharge rates are set up to minimize RFI and EM1 emissions during normal operation. Under overload conditions, the gate discharges very quickly by a large N-channel transistor turning off the external N-channel MOSFET as quickly as possible.
- status output driver—Status circuitry continuously monitors the input and gate-charge control logic. The active-low open-drain status indicator is driven when the breaker is latched off (fault condition). The status circuitry resets with the auto-reset circuitry.

**Q** How do you know you're getting the most from your development tool purchase?

**A** Compare Avocet Systems with the competition.

- A Broad Line of High-Quality Products at Competitive Prices
- Free On-Line Technical Support for Registered Users. No Voicemail!
- Attractive Multi-User Discount Prices & Our "50%+" Educational Discount Plan!
- Unconditional 30-Day Money-Back-Guarantee

Now call the obvious choice!

**AVOCET**  
SYSTEMS, INC.

The Best Source for Quality  
Embedded System Tools

(800) 448-8500

(207) 236-9055 Fax (207) 236-6713

**ANSI-C COMPILERS**

8051 • 68xxx • 68HC11  
6805 • Z80/180/64180  
6801 • 6809  
H8/300 & More

**MACRO ASSEMBLERS**

8051 • 68xxx • 68HC11 6805  
• Z80/180 • Z8 • 8096/196  
6800/6801 • 6809  
H8/300 • 8048 & More

**SIMULATORS/DEBUGGERS**

8051 • 680xx • 68HC11 6805  
• Z80/180 • 6800  
6809 • 8048 • 6502  
8085 & More

**AND HARDWARE**

EPROM Programmers &  
Emulators by EETools, Tribal,  
Softaid, Cactus Logic

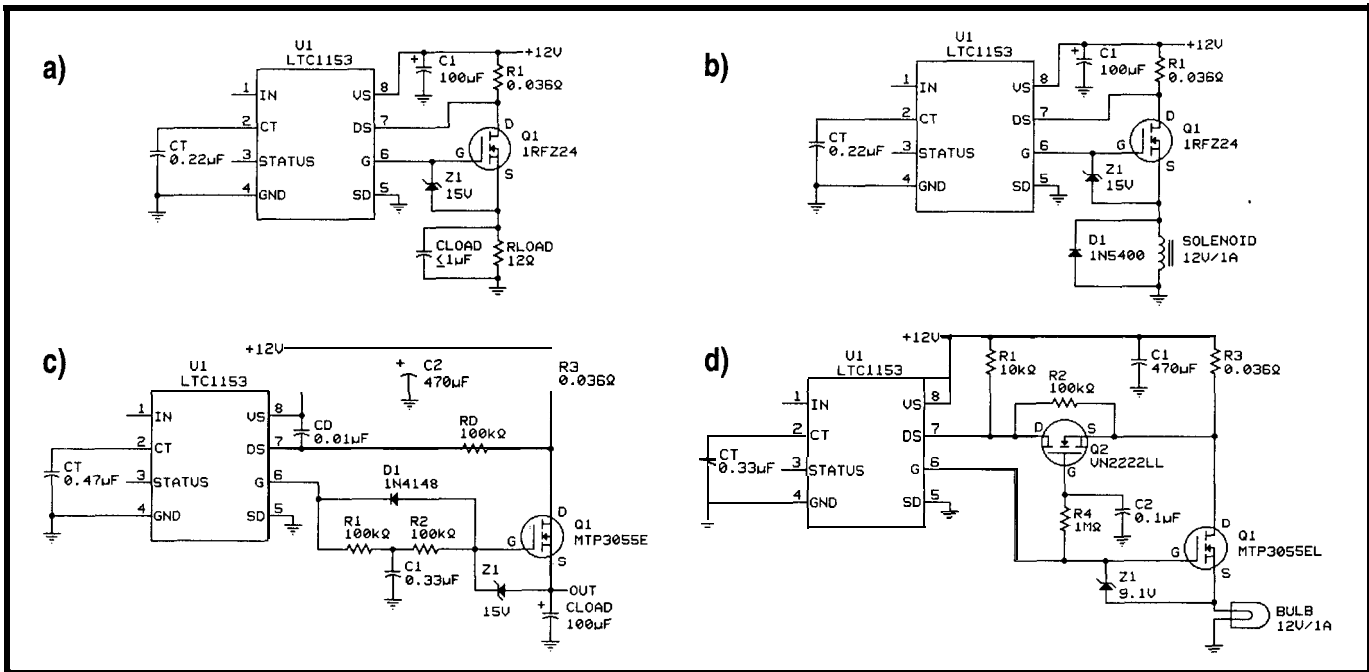


Figure 5—Different loads have different protection requirements. The LTC1153 easily adapts to a number of conditions: resistive (a), inductive (b), capacitive (c), and incandescent lamp (d).

## IT'S HOW YOU USE IT

The LTC1153 offers a great deal of flexibility in the selection of various parameters such as initial hold-off delay, break current, and autoreset time. Additionally, the IC is easily configured to guard against thermal faults and overvoltage. Linear Technology's application notes provide endless ways to configure the electronic breaker to handle just about every imaginable fault condition.

Let's briefly look at several different configurations that illustrate how to handle typical load circuits:

- resistive and inductive loads—These loads require as short a delay as possible to minimize the amount of time the load and the MOSFET switch arc subject to overcurrent. The LTC 1153 drain-sense circuitry has a fixed 10- $\mu\text{s}$  fault delay that prevents false tripping under power supply or load transients. This delay adequately masks short load-current transients and small capacitive start-up surges. Figures 5a and b depict this simple circuit.
- capacitive loads—Circuits with large bypass capacitors are traditionally protected using slow-blow fuses since they can absorb heavy inrush current at powerup.

Rather than just surviving, the LTC 1153 lets you balance circuit parameters so you can fix the problem. Figure 5c shows how the gate drive to the MOSFET is passed through an RC delay network that controls the turn on ramp of the switch. As the MOSFET source voltage follows the gate voltage, the load is gradually powered up, reducing power-supply transients.

Once the system comes to full power, the LTC1153 reacts within the nominal 10- $\mu\text{s}$  fault delay for a short-circuit condition. Note that diode D1 provides a direct path to the LTC1153 discharge control circuitry for controlled or fast turn off.

- Lamp Loads—Lamp turn-on currents can be 10–20 times higher than the rated current. Figure 5d illustrates how the trip current can be shifted up by a factor of 11:1 for 100 ms when the lamp turns on. Following this surge, the trip current drops to the nominally configured level.

Circuit protection takes on many forms. With more protective components appearing on the market, the selection process can be even more confusing. A wide range of factors must be weighed when selecting a protective device for an application.

But, in your concern to protect, don't forget there are cases when it is perfectly appropriate not to provide any protection at all.  $\square$

*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.*

## CONTACT

Pico II  
Littlefuse, Inc.  
800 E. Northwest Highway  
Des Plaines, IL 60016  
(708) 824-0400

LTC1153  
Linear Technology  
1630 McCarthy Blvd.  
Milpitas, CA 95035-7486  
(408) 432-1900

431 Very Useful  
432 Moderately Useful  
433 Not Useful

# CONNECT TIME

conducted by Ken Davidson

The Circuit Cellar BBS  
300/1200/2400/9600/14.4k bps  
24 hours/7 days a week  
(203) 871-1 988-Four incoming lines  
Internet E-mail: [sysop@circellar.com](mailto:sysop@circellar.com)

Since *this month's issue includes the Home Automation & Building Control insert, I decided to start off this month's column with an X-10-related thread. Although the original question is simple, the answer can be very complicated if you want to know all the background theory.*

*In the other thread, we risk starting a language religious war by asking, "Which is better: Forth or C?" The responses are remarkably civilized and consistent.*

## X-10 Fan Control

**Msg#:21339**

**From: Darrell Norquay To: All Users**

Greetings. I have a problem with a recently installed ceiling fan/light combination. It is presently controlled by an X-10 wall switch module, and when on, the thing is horrendously noisy. With only the fan switched on, the noise is a little less than with both fan and light, but it's still obnoxious. Also, the X-10 unit will turn on the unit with the fan only, but not off. With the light switched in, on/off control seems to work fine.

I assume that the on/off switching problem has something to do with the inductive load of the motor, possibly a suitable snubbing network across the switch will take care of this, but what about the noise? I have seen low-noise fan speed controls in the local electrical supply houses. How do they work? Is it possible to modify the X-10 wall switch to eliminate the noise problem? Is there a special X-10 module for fan speed control?

**Msg#:21489**

**From: Ken Davidson To: Darrell Norquay**

The X-10 wall switch module may be used to control lights *\*only\**. You're going to burn out the switch, the fan, or both by having it connected the way you do.

Leviton does have a true fan-control wall switch module that can be used in your setup. I've heard lots of complaints about it being noisy as well. You might want to try it, though. Contact any of the usual module sources such as Home Control Concepts, HAL, and so forth (Radio Shack can special order it).

**Msg#:36771**

**From: Pellervo Kaskinen To: Darrell Norquay**

While I have no personal experience with X-IO, I gather it is time to discuss the two problems you have described from a "theoretical" point. I added the quotes there to emphasize that I try to present educated guesses.

Your first point is that the X-10 can only switch the fan off if the light is connected in parallel. This is not unique to X-10. Any triac circuit has fundamental difficulties in turning off when it has an inductive load. And that is something that I *\*do\** have experience with.

Starting with the data sheet for any triac, we find at least two relevant pieces of information. One is the noise immunity limit: how fast can the voltage rise between the main terminals without the triac turning on. It can be given as a single number with a fixed load between the gate and the "low end" main terminal, or it can be given as a graph versus the gate impedance.

The second important issue is a discussion about snubber networks required under inductive loads. If you work the given data back to the rate of voltage rise, it turns up being related to the first issue, but not the same....

There is the need to turn off, while the first number was given as a guidance against false triggering. Internally, there are some charge carriers that have to migrate from the saturated interfaces to the bulk material or to the external connections before there is again a voltage being applied. This migration takes time. With more time, the saturation characteristics are better, and there is more current just prior to the turn-off attempt.

You have heard that inductive current lags the supply voltage. You may even remember that it lags by  $90^\circ$ . Now, what does it mean for the triac? It means that the current may be very close to the maximum at exactly the point where the voltage reverses. Not the kind of gradual decay that a light bulb load gives with a sine wave supply. In fact, the inductive portion of the current is quite adamant about continuing just as it is, regardless of whether a triac somewhere in the loop wants to turn off.

Note that I say "inductive portion." If you have a light bulb connected in parallel with the motor winding, you get a resistive portion and an inductive portion. If each is 1 A, the vector sum is 1.4 A, with  $45^\circ$  of phase lag. If the



# CONNECT TIME

resistive load is 2 A and the inductive remains at 1 A, then the combined current is 2.2 A at a phase lag of some 26°.

Pick your choice for any combination of resistive and inductive current; you always have a phase lag \*and\* a remaining current at the voltage zero-crossing point. In fact, that current remains the same whether you change the resistive load or not. Why then does the light bulb help the triac!

It simply gives another path for the inductive current to divert when the triac is able to muster a little resistance. The lower the resistance (higher the load in watts or kilowatts), the better bypass route there is.

You mention a possibility of a suitable snubbing network taking care of the turn-off. It may or may not, depending on how much money you are willing to spend on that. A use in snubber duty is not easy on capacitors, which means that the garden variety polyester capacitors fail in short order, sometimes bursting into flames. You need polypropylene or oil paper AC capacitors to stand a chance.

Now, a different viewpoint: If you are using capacitors in general, don't they compensate for the phase lag of the inductive loads?

They do, although not in the usual way of connecting a snubber network over the triac. And not too much with the small-value capacitors that normally are used. But, if you put a real capacitor over your inductive load-something that also takes 1 A from the power line (when alone)-you are truly compensating the inductive load. Wouldn't the triac love that?

Maybe. But there is a complication or two. First, is the fan motor current fixed so that you could accurately compensate for it? Second, you may have to deal with a nice resonance at the primary frequency plus some possibilities of hitting resonances at the harmonic frequencies.

Now, did you want to control the speed of the fan with the triac? If so, you are dealing with a mighty harmonics generator. If you put in enough capacitance to resonate at the fundamental frequency, the harmonics would appear to be attenuated and consumed by the same capacitor. But, you better check if the capacitor can take all that without overheating.

What about the noise, your second concern?

You know that transformers "hum." And yet, they have very tight magnetic laminations and often the coils are embedded so they are locked in place as well. Moreover, they operate at a very pure sinusoidal voltage, right? The fact is, there is a considerable portion of the fifth harmonic in the current even at the sinusoidal excitation. The 300-Hz content in the sound level may actually exceed the 60-Hz component.

Back to the phase-steering speed control with a triac. It has \*much\* more harmonics than the transformer. More-

over, the fan motor has moving parts, unlike the transformer. When they start to vibrate, you sure can hear them.

You can try to filter the supply to the fan motor, but all the common radio noise filters fall into a different frequency range than our 120-1000 Hz. And again, you have to deal with the component heating and bursting danger.

One last caveat: The ordinary phase control systems such as lamp dimmers tend to pump large amounts of DC through the load, as they do not fire symmetrically on positive and negative half cycles. That works okay with incandescent light bulbs, but drives any asynchronous motor crazy. The motor magnetic elements saturate and lose their inductance. After that, our assumption of a fixed inductive current is not worth too much!

The repetitive current jolts really kick your fan into a special Lo-Fi sound emission mode.

How about connecting a capacitor in series with the beast? That would at least cure the DC pumping. But we have just created another potential monster: series resonance. A good series-resonant circuit can draw immense amounts of current and increase the voltage over the motor on one hand and over the capacitor on the other hand to astronomical values. One or the other is bound to fail very soon, and sooner the closer we are to any resonance (base frequency or harmonics).

X-10 or no X-10, I trust you see that the problem is quite universal.

---

## Forth vs. C

**Msg#:16062**

**From: Calvin Krusen To: All Users**

Having worked only in C and only seen Forth, I would like to know if there are any advantages Forth may have over C. My main interest is the Motorola 68HC11 development systems. I have seen development systems with the standard Motorola BUFFALO monitor and systems with a Forth interpreter. The short (20-minute) demo of Forth led me to believe that this was an old language with few advantages (if any) over C.

Is Forth a compilable language or is the interpreter ROM always required? Does Forth support pointers (or another way to access interrupt service routines), different data sizes, structures, or linkable object modules?

**Msg#:17174**

**From: Lee Stoller To: Calvin Krusen**

Any computer language can be compiled or interpreted, but different ones are \*usually\* done one way or another. Forth is almost always interpreted, and generally there are

# CONNECTIME

special versions of various microcontrollers that are manufactured with the interpreter in ROM.

It's only my opinion, but I'd say that the Forth craze has pretty much run its course, while the C craze still has some years left in it. Adding the fact that C is generally compiled, you're probably better off going with C over Forth.

## Msg#:17246

From: James Meyer To: Calvin Krusen

Here's why I like Forth:

In an embedded system, often there isn't a formal statement as to what the system should do. If I have a system that has a Forth interpreter onboard, I can use the system as its own development system. By that, I mean I can write small portions-subroutines-of a larger complete system and immediately execute them to see if (how) they work. I can build up the system in small, debugged steps until I have something that does what I want it to do.

> Is Forth a compilable language or is the interpreter ROM  
> always required?

Forth can be completely compiled. In a final product based on Forth, you could simply include in the object code, only the used parts of what would have been in the interpreter ROM.

> Does Forth support pointers (or another way to access  
> interrupt service routines), different data sizes,  
> structures, or linkable object modules?

Forth supports any and all of those, in the same way that a brick supports the building of an outhouse or a White House.

Learning and using more than one language paradigm makes you a better programmer in \*both\* languages. Forth is sufficiently different from most other languages that it gives you the exercise you need.

## Msg#:17562

From: Don Meyer To: Calvin Krusen

Well, my serious Forth days are quite a while back (but I've always retained a soft spot for it), but here's my two cents.

If you need or want to have the development system (or a good portion thereof) resident on your target system, Forth is great. For fast interactive hacking, it's great. A good language to bring up hardware with in the lab. There \*are\* some various interactive C development systems out there, but I haven't had the chance to play with them any so can't comment on those.

My opinion is that today Forth is much less useful than it was when it was conceived. The power (CPU speed, low memory cost and great size, and large cheap mass storage) of today's computers makes C or equivalent very attractive. Back when 16 KB was a lot of memory and 8" floppy disks were the cutting edge, Forth really shone-there was no way you could get a C compiler with equal functionality on the same hardware.

Now some other stuff for your information. Forth can be compiled, which really consists of creating a target image without the development code (compiler, editor, etc.) and removing the text of each word (SWAP, DUP, etc.) that is normally part of the executable. The resulting executable can be as small as that produced by a C compiler in most cases.

Forth certainly can deal with interrupts and function pointers. There are some object-oriented flavors, which I have no real experience with-especially in recent products.

Just to repeat some old chestnuts:

Forth is an amplifier-a good programmer can produce \*great\* code, a bad programmer can produce \*horrible\* code.

Forth is a religion.

Forth is quite powerful and much less "safe" than C (that should really chill the Wirth folks who think C is dangerous :-)). If you think it might be for you, grab a public domain version and play with it. Call Forth Inc. (really) to see what the high-end, expensive, dead-serious development tools are like today. Drop by the Forth forum on CompuServe or the Usenet newsgroup on the Internet.

## Msg#:18647

From: Kenny Trussell To: Calvin Krusen

I agree with the statements made by James and Don about Forth. I would add one other note. The style of program development that you learn if you do serious Forth programming will make you a much better programmer in ANY language. Buzz phrases like "top-down design," "bottom-up coding," "data-hiding," and so on will make a lot of sense and definitely help you organize and program in any language. *Thinking Forth* (author's name slips my mind) is a good book about the "style" of development which Forth promotes. (Note: it is NOT a tutorial on Forth per se.)

I haven't used Forth in several years, but I "think Forth" a lot.

## Msg#:22781

From: Pellervo Kaskinen To: Calvin Krusen

Being one of the Forth converts of yesteryear, I can only tell one side of the story. And even that has mostly been covered by James.

# CONNECTIME

It is my understanding that any real use of C depends on availability and use of good libraries. For Forth, there are almost no libraries unless you build your own.

The low-level stuff that determines the system throughput have to be in native code of the processor, or as close to it as possible. A compiler with good assembly-generated libraries can do the job in C. In Forth, you can initially create the whole thing in high-level fashion and then turn into CODE statements the portions that seem to be used most heavily.

Besides being extensible ("you build your own language"), Forth allows debugging without a special debugger and in module-by-module fashion (i.e., word by word or in groups as you deem fit). It is like incremental compiling.

Be warned that long threads cause slow-running programs. There are some traditional ways of minimizing that effect, most notably the hashing into parallel vocabularies. The implementation of the language (dialect) you pick can have quite large an effect to the speed.

There are never any forward references in any ordinary Forth statement, unlike in assembly language. Everything that your statement needs has to be predefined. That, on the other hand, is a marvelous way of debugging. When you enter a new statement, it is checked immediately for complete validity. And, because you have not accumulated a mountain of possible problems, you are led immediately to the offending word, not through some lengthy compile and search cycle.

The metacompilers that generate headerless and/or native code to a microprocessor of your choice can be hard to come by, unless you like the old work horses. And there are *some* differences from the plain Forth that you have to learn before you can get through the process of compiling. In that respect, there is fairly little difference from the process you have to go through changing from, say, C to assembly and, correspondingly, from a C compiler to MASM or similar.

Last, if you are looking for somebody else to help or to later support the product, you may have a hard time finding competent Forth programmers. C programmers, on the other hand, are readily available, although few are fluent in the details of programming embedded controllers.

I have tried to be as objective as I can, given the fact that I've never gotten past one feeble attempt of learning C. If I do any programming at all, it is either in Forth (bit twiddling) or in BASIC (number crunching). So, take my opinions with whatever amount of salt they require.

We invite you to 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](mailto:info@circellar.com).

## ARTICLE SOFTWARE

Software for the articles in this and past issues of *Circuit Cellar INK* 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: Circuit Cellar INK, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (203) 8752199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

## IRS

434 Very Useful

435 Moderately Useful

436 Not Useful

## This Parts List Software is for Engineers / Designers

Not MRP: P&V actually assists development!

Windows™ program helps you stay organized

ID	Description	Status
1	2 4334-01	DWG Connector, waferboard (2 position, 4 angle, D44)
2	3 314-004	PS Compression nut Item C44a
3	4 315-001	DWG Sleeve, shaft 100-very close F3E1, OS 01
4	2 6146-02	PS Capacitor, tantalum 33uF 25-volts PC pkg CA 05
5	6 7054-02	PS Grounding at end of lead Item 250
6	8 4485-1B3	PS Resistor, carbon film, 10K RT12B17
7	1 3795-01	PS Integrated circuit BC317A2B U2
8	2 3768-02	PS Integrated circuit BC25128 EPROM U1
9	1 2304-01	PS Power pack PC mount right angle 2
10	4 4108-01	PS Capacitor, 100
11	4 3762-02	PS Integrated cir
12	4 PS 3232-14	PS Wire, #22AWG
13	1 3684-01	PS Transistor, pnp
14	1 3771-02	PS Variable imped
15	2 314-003	PS Compression r
16	2 1296-01	PS Screen, Range
17	4R 1878-01	PS Lubricant, oil
18	1 312-03	DWG Label
19	4 6146-01	PS Capacitor, tan
20	1 6178-04	DWG Bracket, elec

Keep track of:  
 - Drawings & Specs  
 - Suppliers & quotes  
 - Product material costs  
 - Engineering stock

## Parts Vendors

...for independent and departmental projects. Use alone or in a workstation.

\$99 + shipg

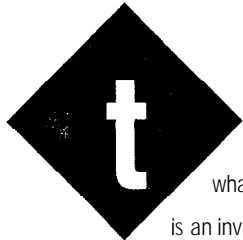
Call 800.280.5176

Requires 486, 8meg min. RAM, Windows™

Trilogy DESIGN™ voice.916.273.1985 fax.916.477.9106 P.O. Box 2270, Grass Valley, CA 95945

# STEVE'S OWN INK

## Your Computerized Future



he worst part about being at the leading edge of any revolution is that you can't predict with certainty what the long-term consequences will be. For some people, this is an invitation to head for cover. For others, it is an invitation to make money gambling on where and when to duck.

There has been a lot of speculation on how computers will eventually affect our lives and whether it will be good or bad. How long will it be before we tell our refrigerator in the morning what we want for dinner that night? Will the refrigerator take inventory, create a shopping list, modem an order to a grocery-delivery service, and perhaps cook it too?

Since I don't have a crystal ball, I can only look at history to see how technology becomes integrated. Obviously, revolutionary change is not instantaneous. No matter how quickly we engineers have adapted to computerization, history suggests it takes a good 30 years before society *starts* being reoriented as the result of the "new" technology. The application of the new technology prior to the "real" changes generally revolves around just doing the old stuff better and more efficiently.

A specific example of this is the transition from steam-powered manufacturing to electric motors. Steam-powered factories were designed as single long buildings with a powered drive shaft. Machines derived power through a pulley-and-belt system connected to the drive shaft. Adding a new machine was easy as long as there was room along the drive shaft.

The advent of the electric motor extinguished the noisy, smelly steam system almost overnight. Initially, however, the electric motor merely replaced the steam engine by turning the long central drive shaft. It wasn't for a number of years that people built individual electric-motor-powered machines to replace the single line. As a result, the long open manufacturing building made way for compartmentalized manufacturing and physically separate production departments. Of course, the real benefit of electric motors came as they were eventually applied to entirely new disciplines.

In my opinion, the real computer revolution hasn't occurred yet either. Yes, we balance our checkbook in 100 ns, have 300 volumes of classics on a tiny plastic disk, and teleconference with the Dalai Lama, wherever he is. We have succeeded in enhancing traditional procedures and processes with the speed of a computer.

For the future, rather than just finding a new way to drive the power shaft, machines with vision, interactive speech, and unlimited control capability will have to result in a behavioral change on our part. Of course, interacting with a robot at Motor Vehicles might be a welcome change from the surly human purser who's typical verbal response is, "Sorry, you're in the wrong line!"

As an engineer, I look forward to building some of the "toys" that bring us into the revolution. But, as a human being who knows that invention shouldn't be devoid of responsibility, I think we have to keep in *perspective* what we have wrought.

When you have a friendly machine that exactly tailors your personal environment, tracks your habits, and presents you with your favorite electronic information, foods, and products, will you prefer its holographic assistance to a real person? Will the information superhighway enlighten and educate people so that they interact more intelligently or lead to isolation and depersonalization?

We've discarded the steam engine and started applying the electric motor. So far, we're just cranking faster. The real revolution comes when this technology is applied to things we never thought possible. It's the social revolution that might have to go along with it that I fear.

