CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

September 1994
Issue #50

Our 50th Issue

EMBEDDED APPLICATIONS

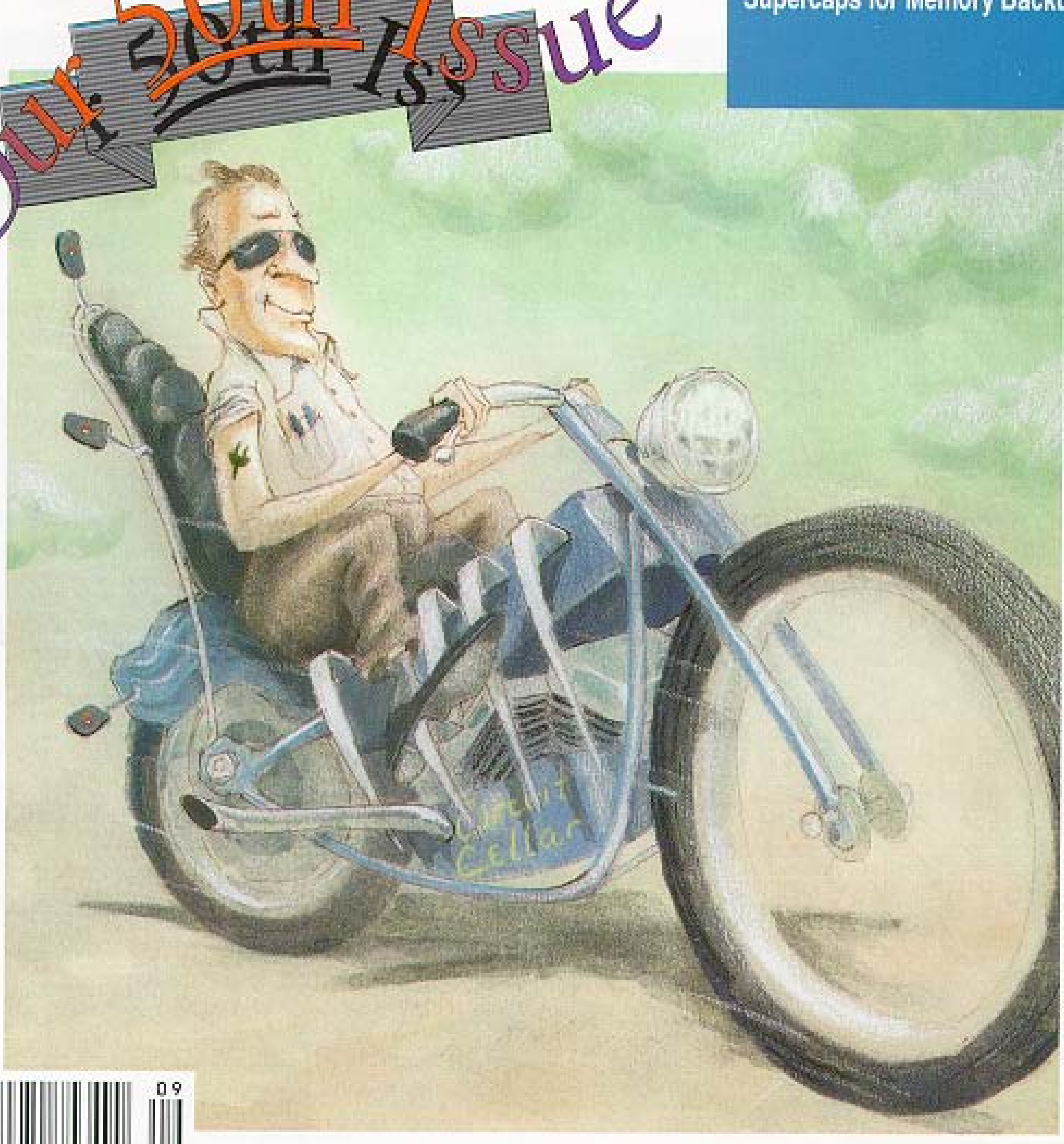
Meet the M68HC16

Add DRAM to an 8031

ISA, EISA, MCA, VL, and PCI Explained

Motorcycle Sensor System

Supercaps for Memory Backup



$3.95 U.S.
$4.95 Canada

EDITOR'S INK

The Golden Issue?

Welcome to our fiftieth issue! When we started this as a 40-page bimonthly with no ads back in 1988, I never would have dreamed I'd see the fiftieth issue go out the door as a 96-page monthly with a group of dedicated advertisers and tens of thousands of faithful readers. As a bonus, we've filled this issue with five feature articles plus our regular columns to make number 50 a bang-up issue. Thanks to our authors and readers, we can continue to bring you first-class material each month. Here's to another 50.

Continuing in the Circuit Cellar tradition of treading into new areas, I'm pleased to announce that starting with the January '95 issue, we will be putting out a quarterly special entitled Home *Automation and Building Control. HABC* won't be just another glossy production that talks down to naive consumers nor will it cover $50,000 installations in multimillion-dollar homes. What we will be covering is home automation technology that you can apply to your own home, whether it be off-the-shelf products, installation techniques, design ideas, or complete projects. We will also be looking at the commercial building control side to find out what is happening in that sector and how it might also be applied to the home. You've come to expect Circuit Cellar to get down to the nitty-gritty and tell it like it is, and *HABC* won't be an exception. There is no such thing as "too technical" when it comes to our readers, so we won't be holding anything back.

If you are already a regular reader of the *Computer Applications Journal, you* don't have to do anything special to receive *HABC.* It will be included in the center of the January, April, July, and October 1995 issues of *CAJ.*

In the meantime, I'm looking for authors to write about all aspects of the home automation industry. Whether it be your experiences with an off-the-shelf product, your own design, or tricks of the trade, we want to hear from you. While we may use the Circuit Cellar HCS as the basis for some projects we write about here, we want to cover all systems currently on the market. Unlike some other industry magazines that only seem to recognize control systems advertised in their pages, we will be providing an equal, unbiased platform for all to use.

Home automation is always a popular topic among our readers (and I know it's popular among our staff). However, it still suffers from lack of consumer awareness and lack of decent user interfaces. It's all of our job—we the pioneers-to help set the stage for acceptance by the masses. A lot of work needs to be done before that can happen, though. We strive to be the medium to carry that work, but we still need your help to fill it. Feel free to contact me with your ideas by E-mail at ken.davidson@circellar.com or using any of the other methods listed on page 6.

This should be fun....

Ken

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Janice Marinelli

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street. Suite 20, Vernon, CT 06066 (203) 875-2751. Second class postage paid at Vernon, CT and additional offices One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 696, Holmes, PA 19043.9613 or call (600) 269-6301.
POSTMASTER, Please send address changes to The Computer Applications Journal, Circulation Dept P.O. Box 696, Holmes, PA 19043-9613.

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

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

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

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

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

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (203) 871-1988; 2400/3600 bps Courier HST, (203) 671.0549

INSIDE ISSUE 50

READER'S INK

Sony Documents

In response to the request listed in "Reader's INK," *CAJ* 48, I think some readers (besides Mr. Khan) might be interested in how to get technical documents from Sony. Here are some phone numbers:

Customer Relations: (800) 282-2848
Camera Tech Info: (800) 222-7669
Document Ordering: (800) 488-7669

The sequence of calls is Customer Relations to get the phone number of the tech info line you need, Tech Info to get the name of the document you need, and then Document Ordering to actually get the document.

In Mr. Khan's case, one of the documents he wants is "Protocol of Control L/LF," Sony part number 9-972-453-11. This is a 26-page pamphlet, sold as a service manual.

Peter Lengsfeld, Addison, IL
Bill Fuhrmann, bill.fuhrmann@tstation.mn.org

Mudassir Farhat Khan is asking for Sony camcorder control protocols in "Reader's INK," *CAJ* 48. There is a file on the Circuit Cellar BBS in area 17 called SONYCTL.ZIP. It is possible to read out the protocol (or part of the protocol) from the assembly listing.

Gyorgy Komarik
via the Circuit Cellar BBS

Encrypted Program?

Got the new issue of the magazine yesterday (*CAJ* 48). Murphy seems to have been present when page 38, Listing 3 was composed.

Line 55 is missing. You need to close the FOR loop begun in line 40 with: `55 NEXT N`.

Line 130 reads: `130 FOR I = J TO MLEN`. When you execute line 130 the first time, J = 26. Thus line 130 should be: `130 FOR I = 1 TO MLEN`.

Interesting article, but the use of GOTO to exit FOR loops several times in Listing 3 on page 38 shows why the structured programming crowd dislikes BASIC. While I realize that at least one reason Microsoft GW-type BASIC is used in programs accompanying articles is that everybody who has DOS through 3.3 has it, I find the limitations of that kind of BASIC overcome the virtues-if there are any.

The use of `INPUT$` to get the encryption key letter (line 35) allows accidental input of more than one character, and the unterminated FOR loop is clumsy compared with Pascal even if the `NEXT` isn't omitted. In Pascal the use of a set makes the process easier to write and easier to comprehend when reading the source code.

```
procedure Test:
var
   C: char;
   N : short:
begin
   Read(C);
   if not (C in ['a'..'z','A..Z']) then begin
     ClrScr;        {not alpha, so bomb1
     writeln('You idiot! Follow instructions!')
   end
   else begin   {char is alpha, so continue}
      N := ord(C);
      case N of      {convert alpha to num O-251
         ord('A')..ord('Z'): Dec(N, 65);
         ord('a')..ord('z'): Dec(N, 97)
      end; {case}
   {rest of program)
   end
end;
```

Call `Test` from a main program.

Frank Kuechmann
via the Circuit Cellar BBS

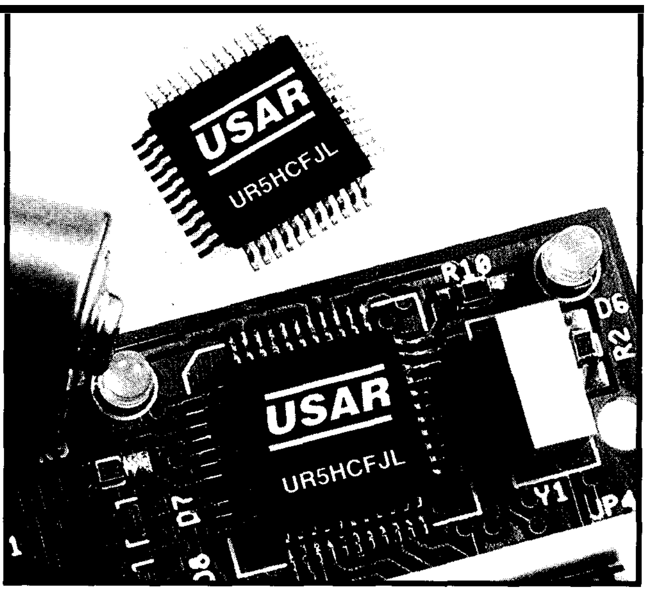
Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, The Computer Applications Journal, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (609) 786-0409. Contact our editorial offices at (203) 8752199.
Fax: All faxes may be sent to (203) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (203) 871-1988 with your modem (300-l 4.4k bps, 8N1).
Internet: Electronic mail may also be sent to our editors and regular authors via the Internet. To determine a particular person's Internet address, use their name as it appears in the masthead or by-line, insert a period between their first and last names, and append "@circellar.com" to the end. For example, to send Internet E-mail to Jeff Bachiochi, address it to jeff.bachiochi@circellar.com. For more information, send E-mail to info@circellar.com.

NEW PRODUCT NEWS

Edited by Harv Weiner



"ZERO POWER" KEYBOARD ENCODER

USAR Systems has announced the first PS/2-compatible keyboard encoder which consumes hardly any power. The GreenCoder IC extends system battery life, has a small form factor, and is easy to integrate.

The **UR5HCFJL GreenCoder** regulates power consumption based on the keyboard's activity. Active, the keyboard uses only 2 mA; inactive, it consumes less than 2 µA. To further reduce keyboard power consumption, the encoder provides an innovative LED dimming feature-when the keyboard is not in use, the LEDs gradually fade.

The UR5HCFJL offers these low-power advantages without compromising functionality or adding to the complexity of the system. The IC, which requires no software drivers or BIOS modifications, comes equipped with a full range of technical features. These include a 16-bit timer that can be used for overall system power management activities such as CPU wake-up and peripheral shutdown, a watchdog and oscillator monitor circuit for high-reliability applications, and a port for an additional input device.

Ready to connect to Fujitsu's FKB Series and other laptop or palmtop keyboards, the UR5HCFJL is available for 3-, 3.3-, and 5-V systems in DIP, PLCC, QFP, and wide-temperature-range packages for $3.45 in quantity. Evaluation kits, with the IC, sample keyboard, evaluation board, connectors, and cables, are also available for **$140**.

USAR Systems, Inc.
568 Broadway, Ste. **#405**
New York, NY 10012
(212) 226-2042
Fax: (212) 226-3215 **#500**

ROBOT KIT

Aclypse Corp. is shipping the ADR-1 Robot Kit for the hobbyist and educational markets. The ADR-1 is 27" tall with a 14" diameter and weighs approximately 16 lbs. The complete robot kit has an onboard computer system that features voice recognition, English speech output, power motor drive, and a battery with a monitoring and recharging system.

The robot has its own operating system and a built-in BASIC programming language for robot instruction. The system can be programmed by connecting to almost any computer or terminal through a serial cable. Program and data files can be sent back and forth between the robot and a personal computer at speeds up to 19,200 bps.

The onboard computer is powered by a 16-bit 8086-compatible CPU operating at 10 MHz with 256 KB of RAM (expandable) and 128 KB of ROM. A lithium battery enables data to be retained if power is lost. Two I/O ports have a total of 12 digital inputs and outputs. Expansion cards can be connected to add memory, sensors, motors, and other new devices.

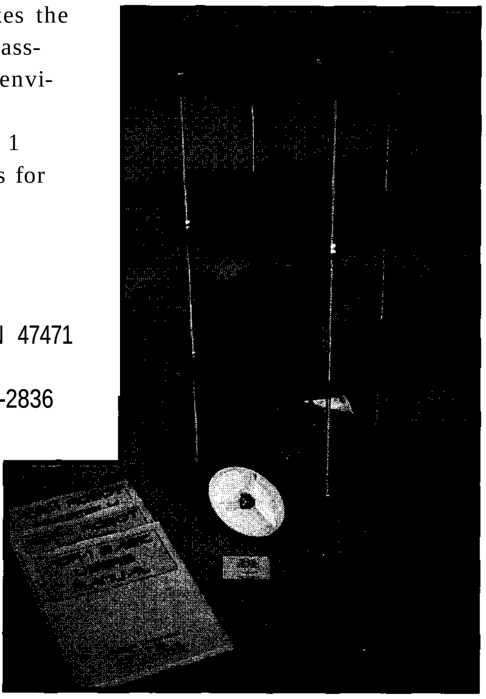
The robot is powered by a 12-V, 6-A battery pack and contains an onboard power and recharge module. The unit features 6" diameter wheels and can move forward or backward at a speed of up to 8 inches per second.

No electronics or advanced computer experience is required to assemble and use the ADR- 1. No special tools are required and assembly normally takes from 2 to 6 hours. The ease of assembly makes the kit ideal for classroom and lab environments.

The ADR- 1 Robot Kit sells for $499.

Aclypse Corp.
Rt. 2 Box 213H
Worthington, IN 47471
(812) 875-2852
BBS: (812) 875-2836

#501

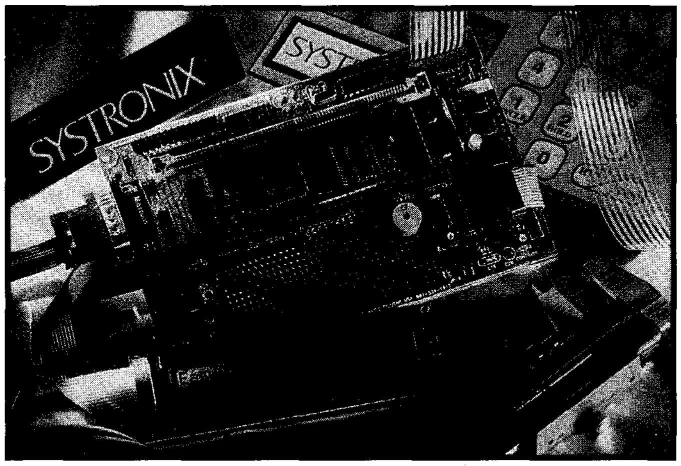


NEW PRODUCT NEWS

UNIVERSAL DEVELOPMENT BOARD

Intellix/Systronix has released a development board that supports all Dallas Semiconductor 805 1 -compatible microcontrollers. The **DPB2** is a complete, single-board computer with LCD and keypad interfaces, serial I/O, relay-driver outputs, analog-to-digital conversion, rugged voltage regulator, and a generous prototyping area.

The DPB2 is a 100 mm x 60 mm (4" x 6.4") card that contains 72-pin SIMM, 40-pin SIMM, and 40-pin DIP sockets for the Dallas microcontrollers. Crystal, serial I/O, and other ports are common to all processor sockets. Processor I/O pins are brought out to labeled headers. The board accepts unregulated 6-13 VDC or can be powered directly with regulated 5 VDC.



A bidirectional RS-232 serial I/O and RS-232 unidirectional printer output are brought out to 2 x 5 headers, and a 2 x 5–to–DE-9 adapter is included. The pinout is consistent with a standard PC/AT 9-pin serial port. A 74C922 4 x 4 keypad encoder/debouncer conditions the keypad and interrupts the processor when a key is ready to be read. A patch area is provided to map the keypad. The LCD interface accepts most 4- or 8-bit parallel LCD displays.

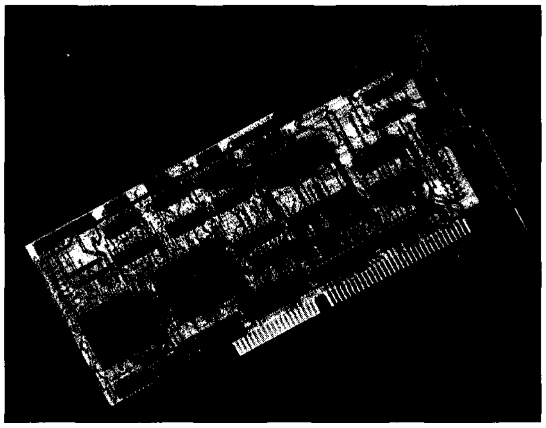
An 8-bit A/D converter with an adjustable 2.5-5.0-V reference voltage is included. Four high-current, open-collector relay drivers with snubbing diodes can directly drive relays, stepping motors, and alarms.

Serial loading and system reset are controlled by a TL7705 monitor chip and a 16V8 programmable logic device. Loading can be initiated by on-card push buttons or the DTR line of the RS-232 serial port. Serial loader software is included.

The DPB2 is assembled and tested in a variety of configurations, and is available starting at $199. A bare board with documentation is available for $49. Software tools are also available.

Intellix/Systronix, Inc.
555 South 300 East, Ste. **#21** • Salt Lake City, UT 84111
(801) 534-I 017 • Fax: (801) 534-I 019 **#502**

SOFTWARE CONTROLLER I/O CARD



Axxon has designed a new peripheral for IBM PC and compatible computers called **SOFT I/O.** This 16-bit product offers four high-speed serial ports (COM1–COM4 using the 16550 UART) plus two bidirectional parallel ports (LPT1 and LPT2).

The SOFT I/O has been engineered to be completely free of jumpers for configuring all of the hardware ports. With software, the user is able to change the address, disable any of the serial and parallel ports, and select from interrupts 3, 4, 5, 7, 9, 10, 11, 12, and 15 for any port. The hardware is completely compatible at the register level with high-speed, 16550-type UARTs with internal 16-byte buffer and IBM-PS/2 printer port specifications.

Also included is support for an onboard BIOS using software-updated flash memory for future expansion. The high-speed serial ports are ideal for use with 9600-bps or faster external modems. The bidirectional ports support scanners and aid in fast data transfer for laser printers or external printer port-driven tape drives.

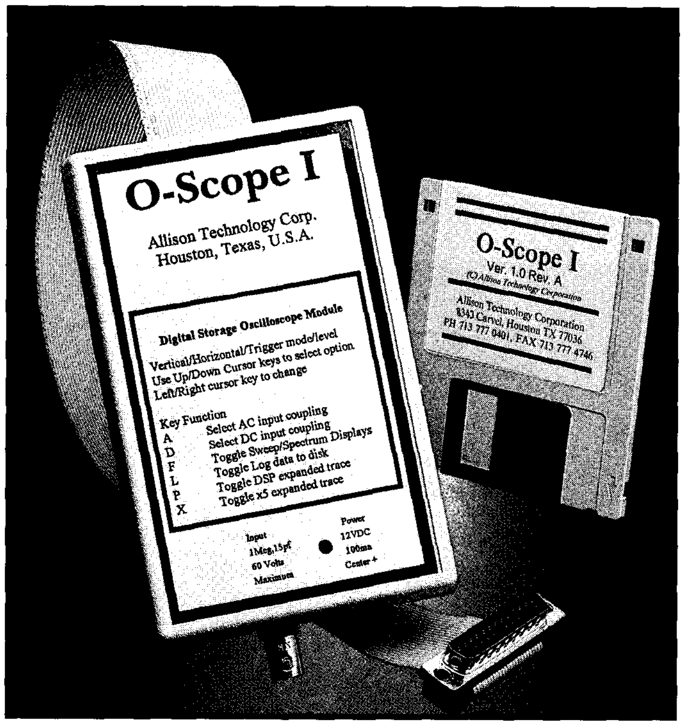
The SOFT I/O sells for $299 (CDN dollars) and includes a five-year warranty.

Axxon Computer Corp.
3979 Tecumseh Rd. East • Windsor, Ontario • Canada **N8W1J5** • (519) 974-0163 • Fax: (519) 974-0165 **#503**

NEW PRODUCT NEWS

DIGITAL STORAGE SCOPE MODULE

A low-cost, digital storage oscilloscope module has been announced by Allison Technology. The **O-Scope I** connects to IBM-PC/AT-compatible computers via the PC's printer port and converts the computer into a digital storage oscilloscope capable of capturing and displaying DC, audio, and low-end ultrasonic frequency input signals. It can be used for power supplies, audio equipment, automotive, and general analog design and repair.



O-Scope I
Allison Technology Corp.
Houston, Texas, U.S.A.
Digital Storage Oscilloscope Module
Vertical/Horizontal/Trigger mode/level
Use Up/Down Cursor keys to select option
Left/Right cursor key to change
Key Function
A Select AC input coupling
D Select DC input coupling
F Toggle Sweep/Spectrum Displays
L Toggle Log data to disk
P Toggle DSP expanded trace
X Toggle x5 expanded trace
Input 1Meg,15pf 60 Volts Maximum
Power 12VDC 100ma Center+
O-Scope I
Ver. 1.0 Rev. A
Allison Technology Corporation
8343 Carvel, Houston TX 77036
PH 713 777 0401, FAX 713 777 4746

O-Scope I is small, light weight, and portable. It draws less than 40 mA of current from a 12-VDC source. It uses standard xl and ×10 oscilloscope probes and works with both desktop and laptop computers. Trace sweeps can be frozen on screen, saved to disk to be used with other programs, or output to a printer via the DOS print-screen function. Vertical ranges of 50 mV to 10 V per division are provided. Sweep rates of 500 us (xl mode) to 100 s per division are available from most AT compatibles. The analog frequency range is DC to 22 kHz for DC-coupled input and 1 Hz to 22 kHz (-3 dB) for the AC-coupled input option.

A 128-point Fourier spectrum analyzer mode provides frequency spectrum information from DC to one-half of the current sample rate. There are 50 samples per division in the xl sweep mode. Two forms of sweep expansion are provided. Expansion modes of ×2 and x5 spread out the sweep by separating samples. Special DSP expansion modes are available from x2 to xl 6 which will expand the sweep by adding calculated samples based on the frequency content of the captured sweep signal. In addition to the sweep, O-Scope I provides voltage, frequency, and period calculations. Voltage measurements include peak-to-peak, average, peak, minimum, and RMS. If more than one cycle exists in the sweep, a frequency and period are calculated. If less than one cycle exists, a pulse period is calculated instead.

O-Scope I sells for $169.95 including an AC adapter and cable. A kit version, which is provided without the shielded case, is also available for **$119.95.**

Allison Technology Corp.
8343 **Carvel**
Houston, TX 77036
(713) 777-0401
Fax: (713) 777-4746

#504

RADIO MODEM

Monicor has announced a low-cost radio modem which eliminates the need for an RS-232 cable. The **IC-15 Radio Modems** contain a UHF radio transceiver that supports 2400 and 4800 bps. A sensitive receiver, powerful transmitter, and fast protocol support line-of-sight distances up to one mile. Greater distances are possible with optional gain antennas.

Each radio has an intelligent RS-232 communications port that can be completely configured for any terminal. Data rates are up to 19,200 bps.

The point-to-point radios are ready to use and are completely self-contained in a Lexan housing. They come complete with an antenna, a rechargeable NiCd battery, and a battery charger. Each radio weighs only 22 oz. The dimensions are 9" x 3" x 1.7".

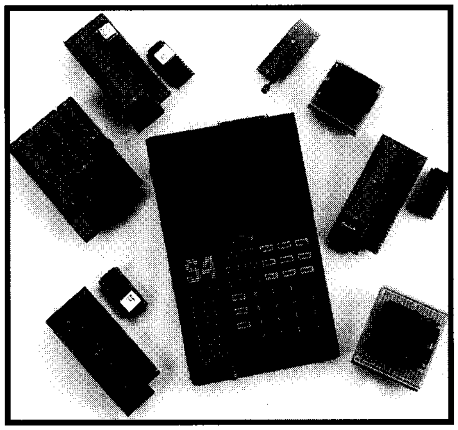
The IC- 15 Radio Modem sells for $950 each (2400 bps) and $1400 each (4800 bps).

Monicor Electronics Corp.
2964 NW 60th St. • Ft. Lauderdale, FL 33309
(305) 979-1907 • Fax: (305) 979-2611

#505

NEW PRODUCT NEWS

DATAMAN S4 PROGRAMMER



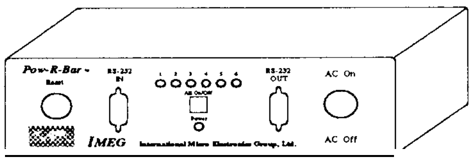
Dataman has announced improvements to their popular S4 hand-held **EPROM Programmer/Emulator.** The S4 now comes standard with 5 12 KB (4 Mbit) of RAM.

Approximately 7" x 4" x 2" and weighing just over a pound, S4 is self-powered and completely portable, making it useful for field service work. S4 can operate for several days on its internal NiCd battery or be used as a conventional desktop programmer, controlled remotely by any computer with an RS-232 port. Besides functioning as a programmer, S4 can also be a ROM emulator. Included with each unit is an emulation cable that plugs directly into the target system in place of any EPROM or ROM up to 5 12 KB x 8 bits.

Equipped with a 32-pin ZIF socket, the S4 supports EPROMs, EEPROMs, and flash memory up to 8 Mbits. It can also program PIC and 8751 microcontrollers, 8-pin serial EEPROMs, and 16-bit-wide devices using optional adapter modules. Additional socket converters are also available for a variety of surface-mount packages. The comprehensive device library is regularly updated to support new devices. Upgrades can be downloaded free of charge from Dataman's 24-hour bulletin board. Other features include a high-contrast, 80-character LCD with a wide viewing angle, 45 color-coded rubber keys (nonbreakable), and a high-impact, molded plastic case which fits comfortably in the palm of your hand.

S4 is covered by a full 3-year warranty and costs $795.

Dataman Programmers, Inc.
22 Lake Beauty Dr., Ste. 101 • Orlando, FL 32806 • (407) 649-3335 • Fax: (407) 649-3310 **#506**



Pow-R-Bar™ $149⁰⁰ (msrp)

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

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

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

IMEG International Micro Electronics Group, Ltd.

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

#104

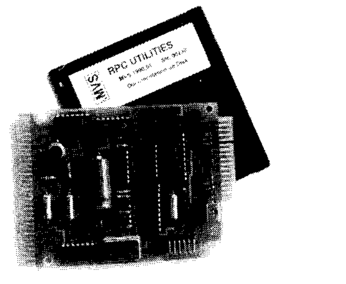
REMOTE POWER CARD!

3 VERSIONS:

RESET
WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

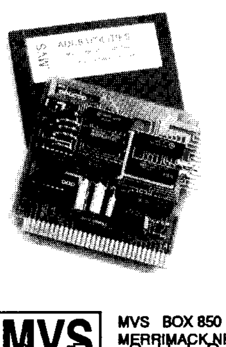
PHONE
TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER
WAKEUP OR SHUTDOWN PC, LATE NITE BACKUP / MODEM, CONTROL AC APPLIANCES



**95$** 27$ OEM

ALL MVS CARDS INCLUDE ASM, BASIC, AND C SOURCE FOR PC OR SBC



8 CHAN ADC

DATA AQUISITION, SERVO CTL, AUDIO
8-BIT RESOLUTION 22KHZ SAMPLE RATE
SHARP CUTOFF ANTI-ALIAS FILTER
CREATE STEREO BLASTER (.VOC) FILES

95$

2 CHAN DAC

VOICE MAIL, MUSIC, ALARMS, CTL VOLT
8-BIT RESOLUTION 44KHZ SAMPLE RATE
PLAYS MONO / STEREO BLASTER FILES
FUNCTIONS AS DIGITAL ATTENUATOR TOO

75$

MVS
MVS BOX 850
MERRIMACK NH
(508) 792 9507

5 YEAR LIMITED WARRANTY
FREE SHIPPING IN USA

#105

NEW PRODUCT NEWS

PCMCIA SERIAL I/O CARD

Smart Modular Technologies Inc. has developed a **PCMCIA** serial **I/O** card for use in any subnotebook, PDA, palmtop computer, or other host system equipped with a Type II slot. The card offers asynchronous compatibility with all DOS- and Windows-based software and connects to all standard serial peripherals. It also features synchronous operation for high-speed data transfers with mainframes.

The serial I/O card includes an internally buffered 16550 UART and an 8530 USART, a combination that provides the flexibility of either asynchronous or synchronous communication. It also provides the flexibility of configuring the asynchronous port as COM1–COM4 while the synchronous port is fully programmable. Its software eliminates the need to set any jumpers.

The card's power-down mode conserves power and extends battery life. In the standby mode, which is automatically entered when no activity has been detected for a period of time, power is removed from most of the card's internal circuitry. In the sleep mode, the host computer essentially turns off the card's power. This mode is entered and exited under the control of the host computer using the PCMCIA interface.

The Serial I/O card sells for $145 in quantity.

Smart **Modular Technologies**
45531 Northport Loop
West, Bldg. **3B**
Fremont, CA 94538
(510) 623-l 231
Fax: (510) 623-l 434

#507

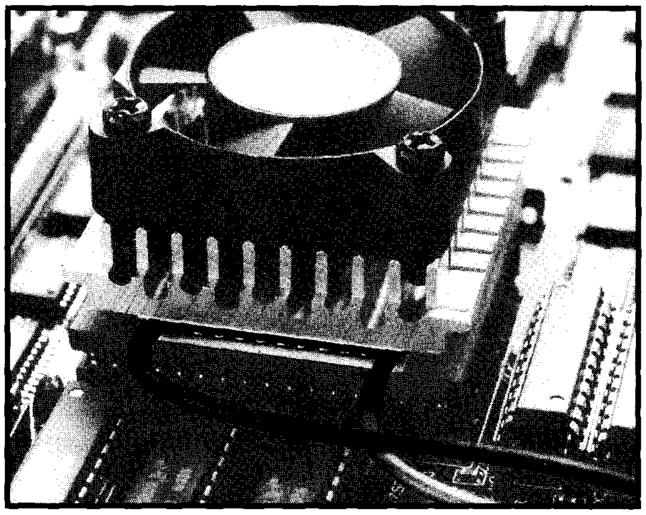
CPU COOLING DEVICE

The Turbo Chip **Cooler (TCC)** from Discovery Data Systems keeps CPUs almost 60°F cooler at room temperature than typical fan/heatsink chip coolers. CPUs will last longer and operate with wider timing margins for all computers. The device is especially effective with higher speed CPUs.

The solid-state refrigerator used in the TCC literally extracts heat from chips without the need for the heatsink gels, special clips, or clamps required by other types of chip coolers. In most cases, when combined with a typical fan and heatsink combination, it maintains the CPU at or below room temperature.

The TCC can be used with any CPU, including 286, 386,486, Pentium, 680x0, and other processors (even DIPs and socketed types). Special heat-conductive mounting pads, provided with the TCC, allow it to conveniently mount with any chip or heatsink combination. A single model fits all devices and can be installed without tools.

The Turbo Chip Cooler sells for $34.95.



Discovery Data Systems
12572 Westmont Dr. • Moorpark, CA 93021
(805) 529-1325 • Fax: (805) 523-8153

#508

FEATURES

Time to Meet Big Brother: Exploring the M68HC16

The M68HC11's big brother—the M68HC16—has been on the market for a short time now. Take a look at how the siblings compare and find out what the more powerful of the two has to offer for your next application.

FEATURE ARTICLE

Dana Romero

Many articles have been written about applications of the 8031/8051 family of micros, a family long established in the field of embedded control. As an alternative, I would like to discuss Motorola's approach to microcontroller design with the M68HC16, their "next step" after the M68HC11.

Having worked with the M68HC11 in the past and being familiar with its architecture and command set, when I read that Motorola was offering a development kit for the M68HC 16, I bought one immediately. The package turned out to be well worth the money-loads of software, programming examples, documentation, and even an introductory book on digital signal processing. In fact, the board and software are oriented toward prototyping A/D and D/A conversions for use in DSP. A much simplified diagram of the board is shown in Figure 1. Note that all the signals shown are also sent to 20-pin connectors, which have through holes for extending them to a wire-wrap area (Photo I). I've left out many signals that this new micro provides, but aren't used in this basic introduction.

Shown in the photo is a socket for a Burr-Brown serial D/A converter, which I didn't have access to and wouldn't have suited my purposes anyway. Both parallel and serial ports are provided. The parallel port, P1, is the primary communication path with a PC via its printer output; the serial port, P2, is secondary and is intended for a dumb terminal or another PC with communications software. Last, note that the components R3 and C3

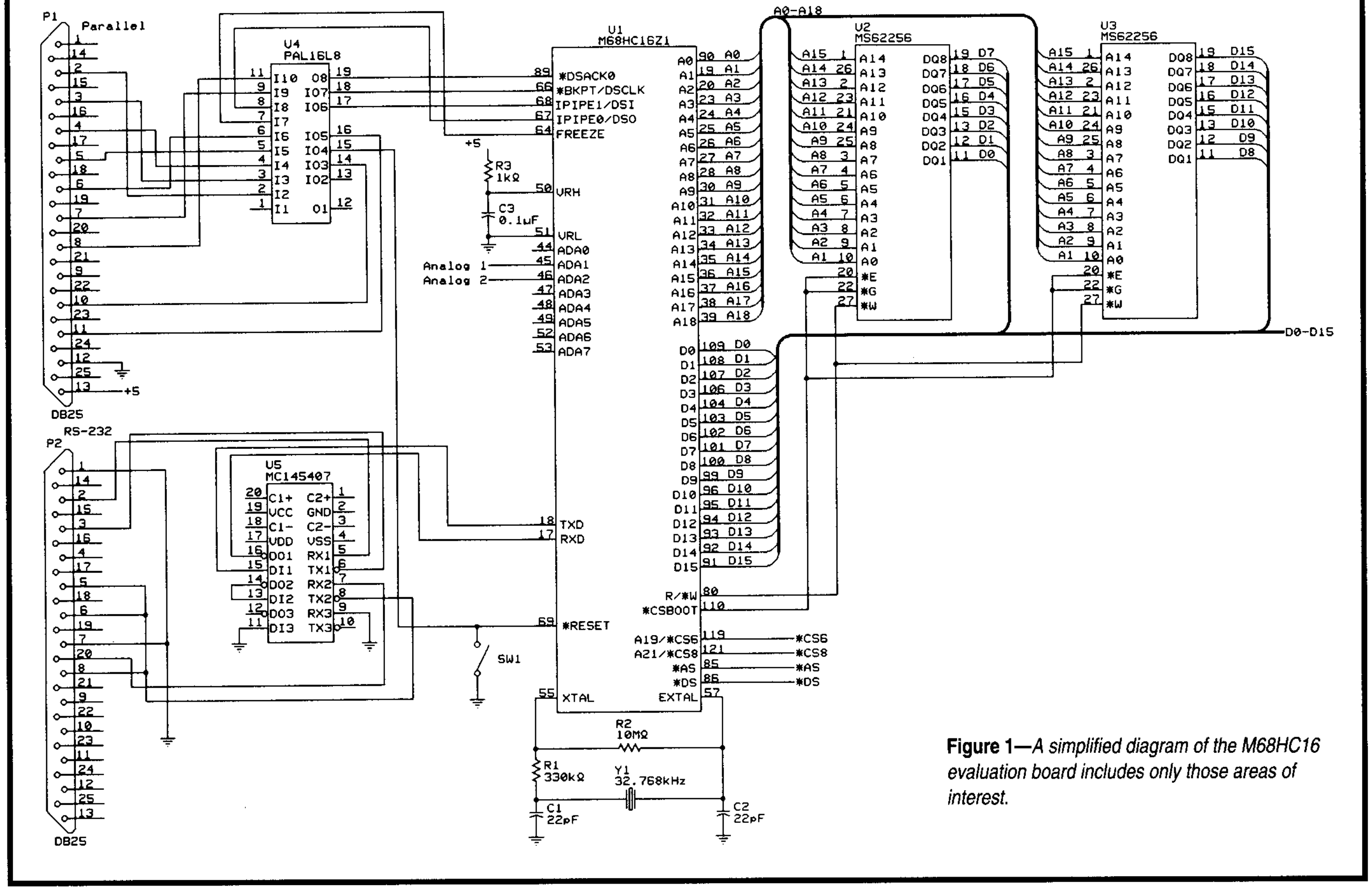


Figure 1—*A simplified diagram of the M68HC16 evaluation board includes only those areas of interest.*

are shown, but are not provided on the board. These are required to set a 5-V reference for the A/D converter section. I was stumped for quite a while because I assumed the ADC would get its reference from within the chip. Wrong. The reference voltage can be set to a value between 0 and 5 V, with the restriction that 0 < VRL < 2.5 V and 2.5 V < VRH < 5 V.

ON LESS FIRM GROUND...

Before discussing applications, I should cover some software and firmware that is essential to a discussion of the M68HC16. Lest I get in over my head (which I found wasn't hard with a device of this complexity), readers should check reference manuals for details. Also see the sidebar for a short comparison of the 'HC16 and 'HC11.

Motorola's background debug mode (BDM) introduced a new concept to me. It is an alternative operating mode in which debugging support is incorporated into the microcode of the CPU. It permits viewing and altering registers or memory, single-stepping, and so forth, with the processor in the circuit being tested. According to the *CPU16 Reference Manual*, BDM is enabled by the status of the /BKPT signal on the rising edge of /RESET and remains enabled until another system reset. Referring to the upper left corner of Figure 1, when BDM is activated, the FREEZE signal is activated by the CPU. IPIPE0 and IPIPE1 change mode from instruction status

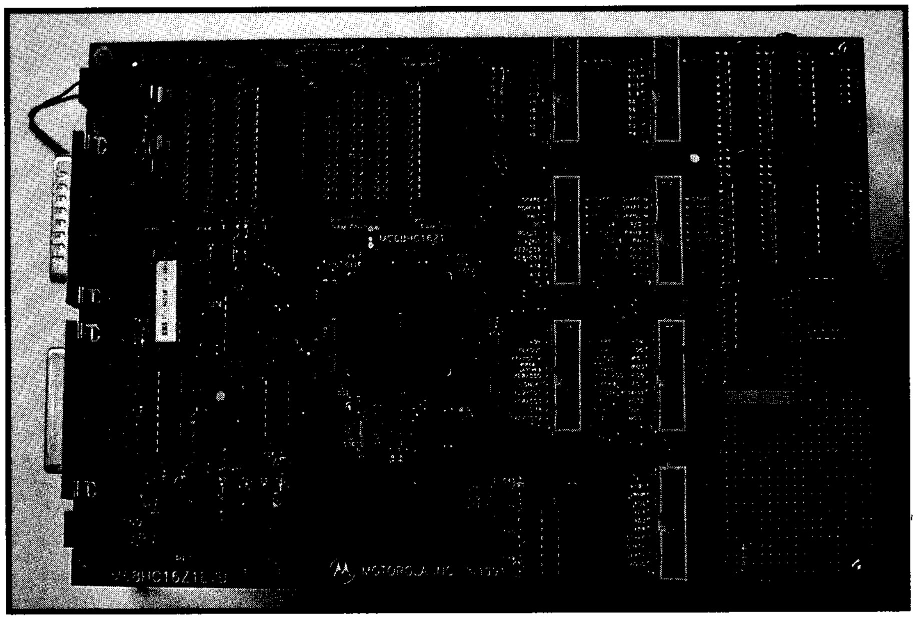
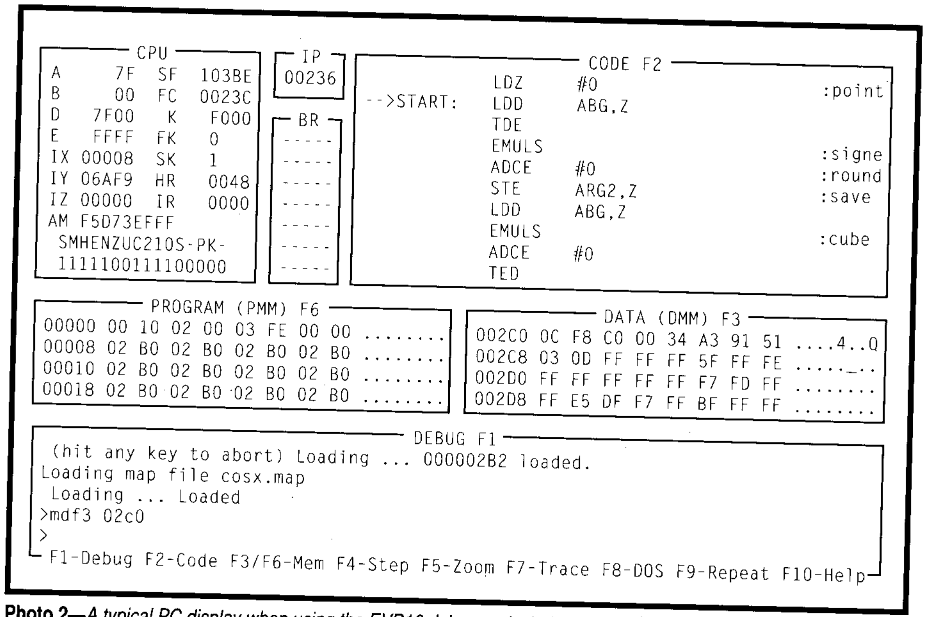


Photo 1—*The Motorola M68HC16 evaluation board includes a prototyping area where the DACs and support circuitry are built up.*



```
┌─── CPU ──────────────┐ ┌ IP ┐ ┌──────────────── CODE F2 ─────────────────────┐
 A       7F  SF  103BE   00236
 B       00  FC  0023C               LDZ    #0                    :point
 D     7F00   K   F000   ─ BR ─ -->START:  LDD    ABG,Z
 E     FFFF  FK   0      -----       TDE
 IX 00008    SK   1      -----       EMULS                        :signe
 IY 06AF9    HR   0048   -----       ADCE   #0                    :round
 IZ 00000    IR   0000   -----       STE    ARG2,Z                :save
 AM F5D73EFFF            -----       LDD    ABG,Z
  SMHENZUC210S-PK-       -----       EMULS                        :cube
  1111100111100000       -----       ADCE   #0
                                     TED

──── PROGRAM (PMM) F6 ──────────────    ───── DATA (DMM) F3 ───────────────────
00000 00 10 02 00 03 FE 00 00 ........  002C0 0C F8 C0 00 34 A3 91 51 ....4..Q
00008 02 B0 02 B0 02 B0 02 B0 ........  002C8 03 0D FF FF FF 5F FF FE ......_..
00010 02 B0 02 B0 02 B0 02 B0 ........  002D0 FF FF FF FF FF F7 FD FF ........
00018 02 B0 02 B0 02 B0 02 B0 ........  002D8 FF E5 DF F7 FF BF FF FF ........

──────────────────────────────── DEBUG F1 ─────────────────────────────────
 (hit any key to abort) Loading ... 000002B2 loaded.
Loading map file cosx.map
 Loading ... Loaded
>mdf3 02c0
>
 F1-Debug F2-Code F3/F6-Mem F4-Step F5-Zoom F7-Trace F8-DOS F9-Repeat F10-Help
```

Photo 2—*A typical PC display when using the EVB16 debugger includes memory, register, and program information.*

signals to DSO and DSI, respectively. In conjunction with /BKPT—now interpreted as a clock signal—these two signals form a full-duplex synchronous serial interface. The PAL at U4 then magically completes the connection to the PC printer port.

From here, the development system provides a debug program for PCs called EVB16 which displays registers, memory, code, breakpoints, and a main command window (Photo 2). Both EVB16 and the PAL were developed by P&E Microsystems. I suppose there are other tools for developing, uploading, and downloading programs for the M68HC16, but this system is hard to beat. If you're not an experienced programmer, it's nice to start writing and testing a small subroutine without worrying about everything crashing and leaving no hint about what went wrong.

MACMATH

To illustrate some of the 'HC16 commands, especially the MAC and EMULS commands, let's consider the Taylor series expansion:

$$\cos(x) \cong 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

This equation is derived from the expression for $\exp(ix)$, and the fact that $\exp(ix) = \cos(x) + i*\sin(x)$. By itself, this isn't going to do much in the way of DSP; but, it is a start in that direction.

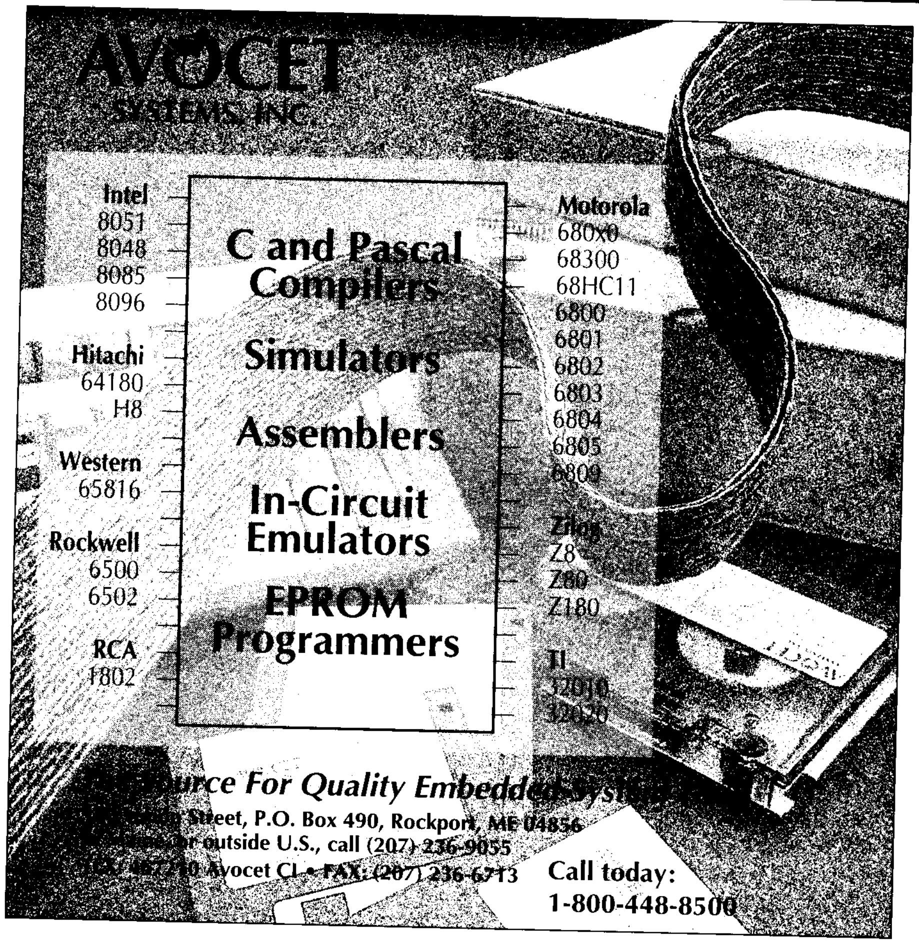
It would be nice to calculate $\cos(x)$ from $x = -\pi$ to $x = \pi$, but the `MAC` command is a lot more tractable if we keep $-1 < x < 1$. Since a user is probably going to scale the final result anyway, let's translate $\cos(x)$ to:

$$\cos(\pi x) \cong \pi^2 \times \left[\frac{1}{\pi^2} - \frac{x^2}{2} + \frac{\pi^2 x^4}{24} - \frac{\pi^4 x^6}{720} + \frac{\pi^6 x^8}{40320} \right]$$

Now if $-1 < x < 1$, an entire cycle can be computed while, at the same time, keeping all of the terms between -1 and 1. A program to calculate $\cos(\pi x)/\pi^2$ for a given x is `COSX.ASM` (see Listing 1). I have precalculated the five coefficients and called them `CF0–CF4`.

At this point, we need to understand the formats for the `MAC` command and the `EMULS` (extended signed multiply) command. For `MAC`, the signed 16-bit numbers to be multiplied have their decimal points between bits 15 and 14 (bit 15 is the sign bit). The product in the 32-bit register, E:D, has its decimal point between bits 31 and 30 (bit 31 is the sign bit). Thus, a multiplicand of \$8000 is equal to –1, \$2000 to ¼, \$C000 to –½, and so on. More specifically, to translate $\pi^2/24$ = 0.411234, multiply by the range 32768 to get 13475 or \$34A3, which is the coefficient `CF2`.

The `EMUL` command is similar, but the range is now 65536. \$8000 represents –½, \$4000 is ¼, and \$1000 is ¹⁄₁₆.



#109

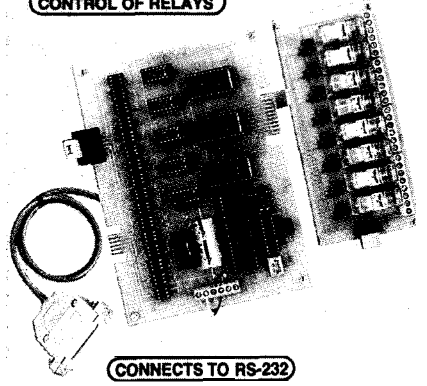
Listing 1--The *COSX.ASM subroutine calculates* $\cos(\pi \times X)/(\pi^2)$. *Enter with the argument in location ARG and the result is* returned in *the* E register.

```
CF0      EQU      $02E0
CF1      EQU      CF0+2
CF2      EQU      CF0+4
CF3      EQU      CF0+6
CF4      EQU      CF0+8
ARG      EQU      $02F0
ARG2     EQU      ARG+2
ARG4     EQU      ARG+4
ARG6     EQU      ARG+6
ARG8     EQU      ARG+8

         ORG      $02E0

         DC.W     $0CF8         one over PI squared
         DC.W     $C000         -1/2
         DC.W     $34A3         +PI^2 over 24
         DC.W     $9151         -PI^4 over 720
         DC.W     $0300         +PI^6 over 40320

         ORG      $0200

***** Initialization Routines  *****

         INCLUDE  'EQUATES.ASM'
         INCLUDE  'ORG00000.ASM'  reset vectors
         INCLUDE  'ORG00008.ASM'  interrupt vectors
         INCLUDE  'INITSYS.ASM'   set EK=F, XK=0, YK=0, ZK=O
                                  set sys clock at 16.78 MHz
                                  disable COP
         INCLUDE  'INITRAM.ASM'   initialize and turn on SRAM
                                  set stack (SK=#$1, SP=#$03FE)

***** Set up Port F as an output *****

         LDAB     #00
         STAB     PFPAR         ;configure port F as general I/O
         LDAA     #$FF
         STAA     DDRF          ;now set port F as an output port
         STAB     PORTFO        ;set initial value to zero

***** Compute powers of x     *****

         LDZ      #0            ;point to bank zero
START:   LDD      ARG,Z
         STAB     PORTFO        ;write odd value of ARG to port F
         TDE
         EMULS                  ;signed multiply E*D>>E:D
         ADCE     #0            ;round E:D into E
         STE      ARG2,Z        ;save square of x  n ARG2
         LDD      ARG,Z
         EMULS    ;cube of x
         ADCE     #0

         TED
         EMULS
         ADCE     #0
         STE      ARG6,Z         sixth power of x
         LDD      ARG2,Z
         TDE
         EMULS
         ADCE     #0
         STE      ARG4,Z         fourth power of x
         TED
         EMULS
         ADCE     #0
         STE      ARG8,Z         eighth power of x
```

(continued)

RELAY
INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

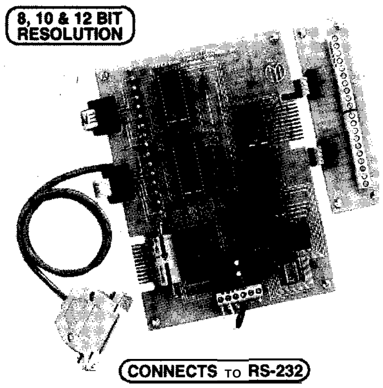


CONNECTS TO RS-232

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

ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER* (16 channel/8 **bit)..$** 99.95
ADC-8G A/D CONVERTER* (8 channel/IO **bit).$124.90**
Input voltage, amperage, pressure, energy usage, joysticks **and a** wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately).
ADC-8E TEMPERATURE INTERFACE' (8 **ch)..$ 139.95**
Includes term. block & 8 temp. sensors (-40˚ to 146' F).
STA-8 DIGITAL INTERFACE* (8 channel).........$ 99.95
Input on/off status of relays, switches, HVAC equipment. security devices, smoke detectors, and other devices.
STA-8D TOUCH TONE INTERFACE*................ **$ 134.99**
Allows callers to select control functions from any phone.
PS-4 PORT SELECTOR (4 channels **RS-422)....$ 79.95**
Converts an RS-232 port into 4 selectable W-422 ports.
CO-485 (RS-232 to RS-42WRS-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 800 number to order FREE INFORMATION PACKET. Technical Information (614) 464-4470.

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

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

ELECTRONIC ENERGY CONTROL, INC.
360 South Fifth Street, Suite 604
Columbus. Ohio 43215-5438

#110

PIC16C5x Real-time Emulator

Introducing RICE16-5x and RICE5x-Junior, real-time in-circuit emulators for the PIC16C5x family microcontrollers: affordable, feature-filled development systems from **$599***

* Suggested Retail for U.S. only

RICE16 Features:

Real-time Emulation to 20MHz

- PC-Hosted via Parallel Port
- Support all oscillator types
- 8K Program Memory
- 8K by 24-bit real-time Trace Buffer
- Source Level Debugging
- Unlimited Breakpoints
- External Trigger Break with either "AND/OR" with Breakpoints
- Trigger Outputs on any Address Range
- 12 External Logic Probes
- User-Selectable Internal Clock from 40 frequencies or External Clock
- Single Step, Multiple Step, To Cursor, Step over Call, Return to Caller, etc.
- On-line Assembler for patching instruction

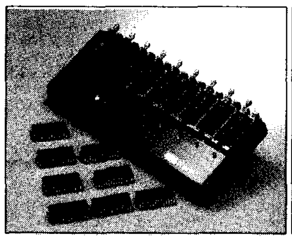
- Support 16C71, 16C84 and 16C64 with Optional Probe Cards
- Easy-to-use Windowed Software
- Comes Complete with TASM16 Macro Assembler, Emulation Software, Power Adapter, Parallel Adapter Cable and User's Guide
- 30-day Money Back Guarantee
- Made in the U.S.A.

RICE5x-Junior supports PIC16C5x family emulation up to 20 MHz. It offers the same real-time features of RICE16 without the real-time trace capture.

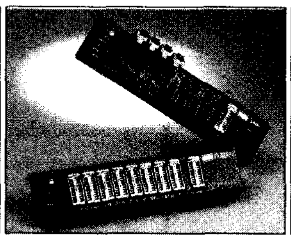
PIC Gang Programmers

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

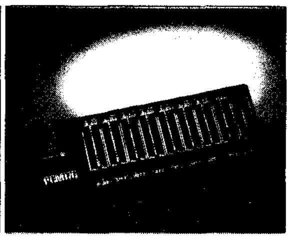
■ Stand-alone COPY mode from a master device ■ PC-hosted mode for single unit programming ■ High throughput ■ Checksum verification on master device ■ Code protection ■ Verify at 4.5V and 5.5V ■ Each program cycle includes blank check, program and verify eight devices ■ Prices start at **$599**



PGM16G: for 16C5x family
Throughput: 16C54: 4 seconds
(0 devices) 16C57: 10 seconds



PGM47: for 16C71/84
16C71: 10 seconds
16C84: seconds



PGM17G: 17C42
17C42: 13 seconds

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

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

TransdAtA

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

See us at Embedded Systems West booth #749 #111

For this reason the stored arguments, ARG2–ARG8, need to be divided by two to be in the proper format for the MAC command. The four LSRW instructions just before the MAC section handle this.

Before running this program from within EVB16, be sure to first set the data window (F3) with the command MD F3 0 2 E 0 to display the coefficients CF0-CF4 and the values ARG-ARG8. (You can watch the ARGs being changed if you trace through the program with the Tn command, where *n* is the number of steps to trace.) To watch the timing on a scope, preload ARG with an odd value using DMM. W 0 2 F 0 from within the Fl (debug) window. The odd value sets Port F's bit zero to one for the duration of the loop. Since port F has been configured as an output port, the signal MODCLK is now redefined as F0. Your scope probe on this line will show the time that it remains positive.

I measured 100 µs for one loop. Although that seems slow, there are a lot of 16- and 32-bit calculations going on. Interestingly, eliminating the eighth-power term of the approximation only reduces the calculation time to 90 us. Timing would be a concern if one wanted to output a waveform.

D/A EXPERIMENTS

One of the first things that I wanted to do with the M68HC16 board was to experiment in generating audio waveforms that would be difficult or impossible to produce with analog circuitry alone. My ultimate goal was to produce unique tones. Using the prototyping area on the board, I added two 12-bit DACs whose outputs are then sent to a summing junction as shown in Figure 2.

I summed the two DAC outputs so that one could provide a fundamental or low-frequency tone while the other provided a harmonic or nonharmonic, high-frequency tone. Alternatively, one DAC might help generate an envelope of sorts for the other's tone, and so on. (Note that, just because the circuit uses DAC 1222 converters, you're not restricted to the slower response of 12-bit resolution. Instead, through software, a 16-bit

Listing l-continued

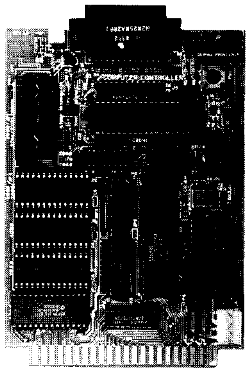
```
          LSRW     ARG2,Z           ;change format for MAC command
          LSRW     ARG4,Z
          LSRW     ARG6,Z
          LSRW     ARG8,Z

***** Multiply and accumulate  *****

          ORP      #$0010           ;set saturation mode in MAC reg
          CLRD
          TDMSK
          LDE      CF0,Z
          TEM                       ;first term is one over PI^2
          LDX      #ARG2
          LDY      #CF1
          LDHI
LOOP:     MAC      2,2
          CPY      #$02EA             last coefficient?
          BNE      LOOP
          TMER                        return with rounded result in E
          STE      $02EA,Z
          RTS                         remove this to measure timing
          CLRD
          STAB     PORTFO             all bits set to zero
          LDAB     #08
DELAY:    DECB                        negative state for scope timing
          BNE      DELAY
          JMP      START

BDM:      BGND
```

output port can be used for 12-,10-, or 8-bit conversions.)

I wired both DACs for bipolar, rather than unipolar, operation. I adapted the circuitry, not from National Semiconductor, the manufacturer, but from Analog Devices' specification for the AD7541A. (As far as I can tell, the chips are functionally the same.) The outputs for given input codes are listed in Table 1. Note that V_{OUT} as referred to in the table is the voltage that will appear at the output of either U11 or U12; op-amp U13 will produce the negative sum of these voltages at its output. Also, V_{IN} is set by voltage reference D1 and potentiometer R14 and goes to V_{REF} of both DACs.

With this hardware in place, the real complexity comes in the software. The fact that decoded chip selects come directly from the microcontroller means more work in software. A program that produces a basic audio waveform is SPIKE.ASM (see Listing 2). This program shows how to set up the chip selects, output to the

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

INPUT/OUTPUT
- Console RS232 – autobaud detect
- Line printer RS-232
- Three 8-bit parallel ports
- EXPANDABLE!
  - Compatible with 12 BCC expansion boards

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

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

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

CALL FOR OEM PRICING

MICROMINT, INC. 4 Park Street, Vernon, CT 06066
In Europe: (44) 0285-658122 • In Canada: (514) 336-9426 • In Australia: (3) 467-7194
Distributor Inquires Welcome!

#119

▶ GoodStuff ◀

Bar Code Sensor
Battery Controllers
Clock/Calendars
Digital Power Drivers
DTMF & Phone Interfaces
Firmware Furnace Widgets
HCS-II Hard-to-find Parts
I^2C Bus ICs
IR LEDs & Photodiodes
IR Data Link Parts
IR Remote Control
Laser Diode Controllers
Linear Hall Effect Sensor
Multiplexers & Crosspoints
Power Op Amp
Remote Temperature Sensor
Stepper Motor Drivers
Watchdogs & Power Monitors
8051 Information
and more!

Use a soldering iron? Get the parts!

UPS: Ground/2nd day $6 5019.00 to 48 US states, COD add $4.50. PO Boxes and Canadian addresses. $6 for USPS mail. Check, MO, or COD only; no credit cards, no open POs. NC residents add 6% sales tax. Quantity discounts start at five parts. Data sheets Included with all parts.

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

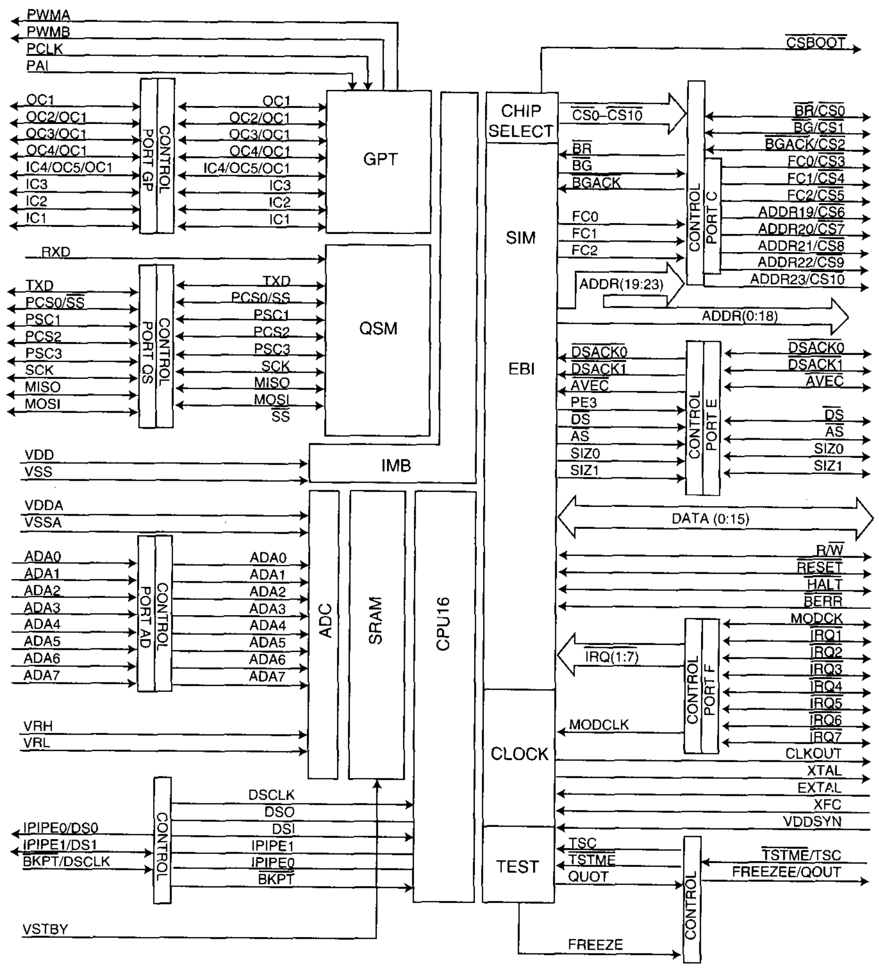
Pure Unobtainium

▶ Your unusual part5 source ◀

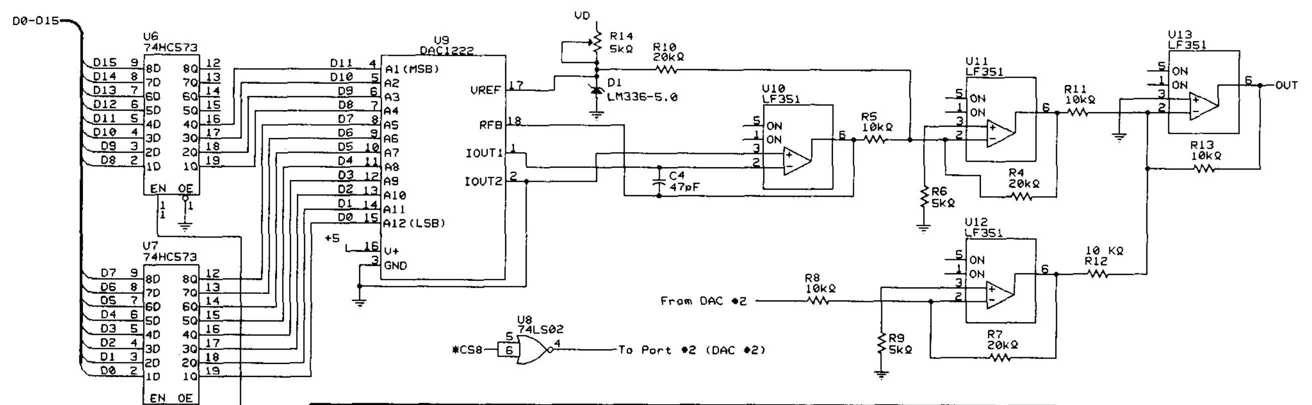
13109 Old Creedmoor Road Raleigh NC 27613-7421
FAX/voice (919) 676-4525

Meet the M68HC16

The M68HC16 is Motorola's 16-bit step after the M68HC11 line of microcontrollers. Additional features make it more suitable for digital signal processing. The M68HC16 architecture is based on modules similar-but not identical-to those used in the M68HC11. These modules include a CPU core, RAM, 10-bit analog-to-digital converter (ADC), general-purpose timer (GPT), queued serial module (QSM), system integration module (SIM), clock control, and port or chip selects. All these modules are running on one IC at 16.77 MHz.



Although the basic modules of the M68HC 16 are named differently than in the M68HC11, most of the features of the 'HC11 are present in the 'HC16. Of course, the data path is 16 bits wide, and it has 1 MB of data space and 1 MB of code space. Although the documentation implies that the address bus is 24 bits wide, ADDR20–ADDR23 follow the output of ADDR19. According to the ***Technical Summary,*** these are brought out to pins only for test purposes. This summary also claims that the 'HC16 is upwardly code compatible with the 'HC 11. However, the 'HC16 has no instructions to increment or decrement any of the 16-bit registers directly as the 'HC11 does with the INX and INY instructions. The only instructions that comes close are AIX and AIY: add immediate to X or Y. The

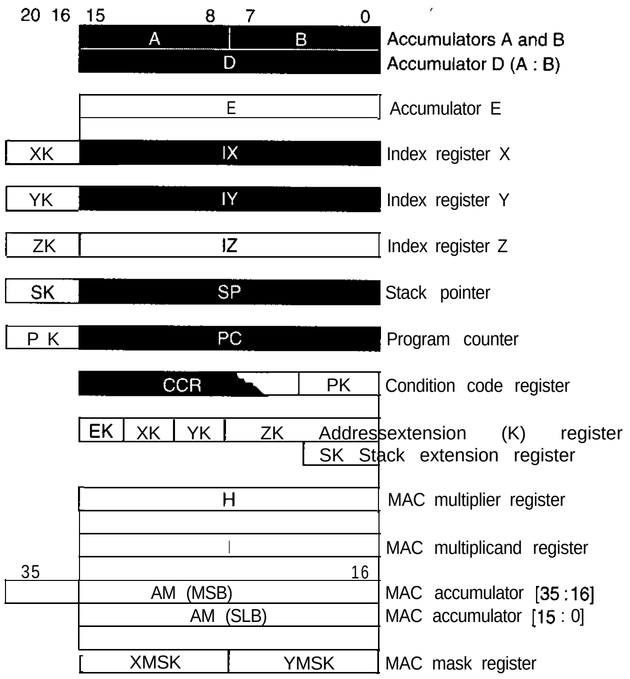


Part	+5V	GND	+10V	-10V
U1	8,21,40,61,84,98,113,124	9,22,41,62,83,97,112,125		
U2-U3	28	14		
U4	20	10		
U5	19	2	17	4
U6-U7	20	10		
U8	14	7		
U10-U13			7	4

Figure 2—*A two-channel DAC and some filters on the outputs can be used to generate tones under processor control.*

processor would be more accurately said to be "upwardly register compatible."

The *CPU16 Reference Manual* devotes a whole appendix to the comparison of the two command sets. Looking at the registers found on the 'HC16, you'll see that the shaded registers in the diagram are also found in the 'HC 11, but that the condition code register is only partially shaded. This partial shading is because the 'HC16 adds three new flags, devoted to the multi-ply-and-accumulate (MAC) register, and a field of three bits to mask eight interrupts. The "K" extension registers are four bits each and concatenate with their corresponding 16-bit registers to form 20-bit addresses, with the exception of the EK register. The EK register concatenates with a word following an opcode in the extended addressing mode.



A significant conceptual difference between the 'HC 16 and the 'HC 11 comes with the addition of the MAC section, which consists of a 36-bit accumulator (AM), multiplier registers (H and I), and 8-bit mask registers (XMSK and YMSK). This section was specifically designed for DSP calculations. XMSK and YMSK are used for modulo addressing. Readers should study these for their own applications; but for now, it is sufficient to say that setting both to zero disables modulo addressing.

With this in mind, a single MAC instruction of the form MAC x0, y0 performs the following sequence:

1. A 16-bit signed fraction in the H register is multiplied by the same in the I register; the product is shifted left once to align the decimal and then placed in the 32-bit register, E:D. DO is set to zero.

2. The aligned product is added to the current contents of AM, and flags in the CCR are set accordingly.

3. The X and Y registers are incremented by x0 and y0, respectively.

4. Contents in H are saved in the Z index register.

5. The word pointed to by XK:IX is loaded into H and the word pointed to by YK:IY is loaded into I.

Before using the MAC command, place values into H and I with LDH I, which loads H with the word pointed to by XK:IX and I with that pointed to by YK:IY. See Listing 1 for an example.

188SBC

Use Turbo or MS 'C

Intel 80C188XL
Two 1 meg Flash/ ROM sockets
Four battery backed, 1 meg RAM
16 channel, 12 or 16 bit A/D
8 channel, 12 bit D/A
2 RS-232/485 serial, 1 parallel
24 bits of opto rack compatible I/O
20 bits of digital I/0
Real-time clock
Interrupt and DMA controller
8 bit, PC/104 expansion ISA bus

Power on the quiet, 4 layer board is provided by a switcher with watchdog and power fail interrupt circuitry.

The 188SBC is available with Extended Interface Emulation of I/O - a Xilinx **Field Programmable Gate Array** and a breadboard area. Define and design nearly any extra Interface you need we'll help! 188SBC prices start at $299.

Call right now for a brochure!

552SBC

The 80C552 is an **8051** with:

8 ch. 10 bit A/D	**2 PWM outputs**
Cap/cmp registers	**16 I/O lines**
RS-232 port	**Watchdog**

We've made the **552SBC** by adding:

2-RS-232/485 multi-drop ports

24 more I/O	**Real-time Clock**
EEPROM	**3-RAM & 1-ROM**
Battery Backup	**Power Regulation**
Power Fail Int.	**Expansion Bus**

Start with the Development Board - all the peripherals, power supply, manual and a debug monitor for only $349. Download your code and debug it right on this SBC. Then use OEM boards from $149.

True Low-cost In-circuit Emulation

The DryICE Plus is a low-cost alternative to conventional ICE products. Load, single step, Interrogate, disasm, execute to breakpoint. Only $448 with a pod. For the 8051 family, including Philips and Siemens derivatives. Call for brochure!

8031 SBC as **low** as $49

Call for your custom product needs. Quick Response.

VISA MasterCard

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

Since 1983

(619) 566-1892

70662.1241 @compuserve.com

#112

Listing 2—*The SPIKE.ASM sample program produces an audio waveform by shifting the D register.*

```
        INCLUDE  'EQUATES.ASM'  ;table of EQUates for common regs
        INCLUDE  'ORG00000.ASM' ;initialize reset vector
        INCLUDE  'ORG00008.ASM' ;initialize interrupt vectors

        ORG       $0200         ;start prog after interrupt vectors

*****   Initialization Routines *****

        INCLUDE  'INITSYS.ASM   ;set EK=F, XK=0, YK=0, ZK=0
                                ;sys clock at 16.78MHz, disable COP
        INCLUDE  'INITRAM.ASM   ;initialize and turn on SRAM
                                ;set stack (SK=#$1, SP=#$03FE)

***** Start of main program *****

        LDD      #$03FF          set CS6-CS10 as 16-bit chip selcts
        STD      CSPAR1
        LDD      #$0200          block size = 2k
        STD      CSBAR6          CS6 active when address = $02XXXX
        LDD      #$0300          block size = 2k
        STD      CSBAR8          CS8 active when address = $03XXXX
        LDD      #$7430          chip selects are asynchronous,
        STD      CSOR6           write only with data strobe,
        STD      CSOR8           in user space

***** Set DAC #2 for offset voltage *****

        LDX      #0000          ;essential for proper chip select
        LDAB     #03            ;CS8
        TBXK
        LDAA     #0C            ;8 gives zero volts
        LDAB     #00            ;$0C gives DC offset
        STD      $0000,X        ;(offset value: $0-$F or 0-15)

***** Increment and send word  to DAC #1 *****

START:  LDAA     #00
        LDAB     #02            ;CS6
        TBXK
        LDAB     #04            ;start with D= 0004

OUT1:   STD      $0000,X
        ASLD                    ;multiply by two
        CMPA     #$09           ;stop when A>8
        BMI      OUT1
        CLRB
        LDAA     #08            ;reset A & B

OUT2:   STD      $0000,X
        ASRD                    ;divide by two
        BCC      OUT2
        JMP      START

*****   Exceptions/Interrupts   *****

BDM:    BGND                    ;exception vectors point here
                                ; and put the user in bckgrnd mode
```

DACs, and use the D register for 16-bit shifts.

An oscilloscope on the output of U13 shows what first appears to be a rectified sine wave (Figure 3). But a close examination of the program and the waveform show that it is far removed from a sine wave-change a few parameters and the results look quite different. Specifically, DAC #2 has been held at a constant offset. At the next level, we can vary this according to external input such as the A/D section or an interrupt, for example.

Binary into DAC MSB	LSB	Analog Output
1111	1111 1111	$+V_{IN}\left(\frac{2047}{2048}\right)$
1000	00000001	$+V_{IN}\left(\frac{1}{2048}\right)$
1000	00000000	0 V
0111	1111 1111	$-V_{IN}\left(\frac{1}{2048}\right)$
0000	00000000	$-V_{IN}\left(\frac{2048}{2048}\right)$

Table 1--**The** *DAC1222 digital-to-analog converter produces a bipolar output referenced to V,,*

From here, it's up to your imagination. If you plan to work with the M68HC16, you really should have the following: the *CPU16* Reference ***Manual*** (software descriptions), the *M68HC16* ***User's Manual*** [for voltage and timing specifications), and the MASM16 assembler from Motorola. Also, the EVB16 debugger and PAL firmware from P&E Microsystems work quite well with MASM16 and greatly speed learning the ins and outs of this device.

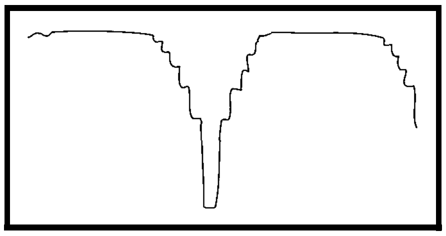


Figure 3—*Looking at U13 pin* **6 when** *running SPIKE .A SM produces something like a rectified sine wave.*

Dana Romero holds a B.S. in Mathematics from the University of Utah.

CONTACT

P&E Microsystems
P.O. Box 2044,
Woburn, MA 01888
(617) 944-7585
Fax: (617) 353-9205

Motorola
(602) 952-4103
Fax: (602) 952-4067

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

DRAM on an 8031: It's Not as Hard as You'd Think

FEATURE ARTICLE

Hugo Cheung

Although the 8051 family is not designed to use DRAM, it can be done. Hugo shows us how to connect the 8051 family CPU to 256 KB or more DRAM.

Many microprocessor applications such as data logging, printer buffering, and serial and parallel data conversion require large amounts of data memory. Typically, most applications need a few hundred kilobytes to a few megabytes. SRAM, which is expensive and takes up a large PCB area, does not provide a practical solution. DRAM, on the other hand, costs about $30 per megabyte making it a good candidate for this kind of application.

However, using DRAM is inconvenient because of its need to be refreshed with burst modes or cycle stealing. Burst refresh requires the CPU to stop accessing the refreshing DRAM. Cycle stealing, which I use in this application, refreshes the DRAM during the CPU instruction fetch cycle: Unlike the methods used for the IBM PC, XT, or AT, these refresh techniques have no software overhead and the CPU doesn't need to halt while DRAM refreshes.

After looking at the timing for DRAM, I'll investigate the timing for the 8051 (or any 8051 family device). Specifically, I'll look at how we can fit DRAM into an 805 1 application. Since understanding the detail timing of a system is critical to implementing DRAM in an application, I'll provide a lot of timing diagrams along with circuit diagrams.

OVERVIEW OF THE 41256 DRAM

To implement my data logger, I decided to use eight 41256 memory chips. The 41256 is organized as 1 x

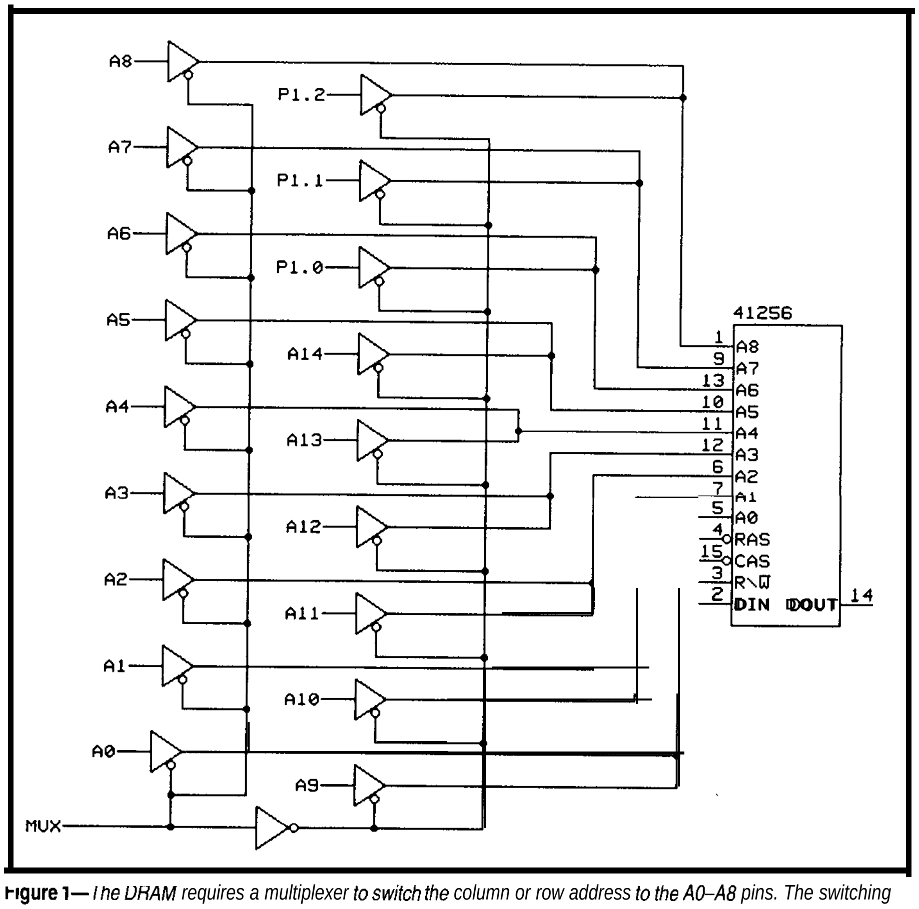


Figure 1—*The DRAM requires a multiplexer to switch the column or row address to the A0–A8 pins. The switching is controlled by the /MUX signal.*

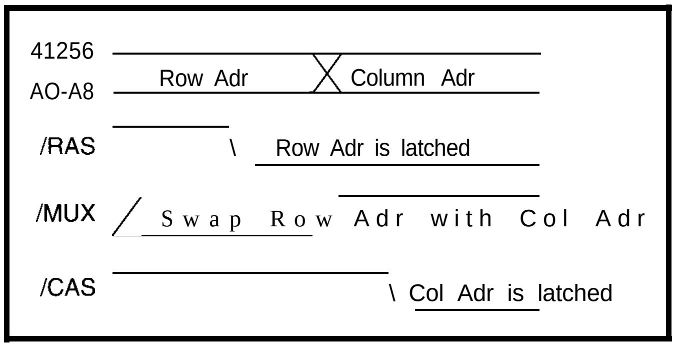


Figure 2—/RAS and /CAS latch the row and column addresses into the DRAM. The address multiplexing signal (/MUX) has to change between /RAS and /CAS.

256 Kb of RAM and uses separate data input and output lines. It requires 18 address lines to give it a total of 256K address locations. The address lines are multiplexed to nine pins (AO-AS). The active-low multiplexing control signals are /CAS (column address strobe) and /RAS (row address strobe). All together, the 41256 has a total of 16 pins, including the /WE (active-low write-enable signal) and two power pins (+5 V and ground).

When /RAS is asserted, the information on the nine address lines is latched internally and is used as the lower 9 bits of the total 18-bit address. When /CAS is asserted, the information on the nine address lines is used as the upper 9 bits of the complete 18-bit address.

ADDRESS MULTIPLEXING

The 8031 has 64 KB of address space for data memory and I/O. In my application, 32 KB from OOOOH to 7FFFH is allocated for the DRAM. The 256 KB DRAM requires 18 address lines. However, the CPU can only provide 15 address lines (AO-A14) from the address bus. To get the remaining lines, I used three bits of the CPU's Port 0 to make up a total of 18 address lines for the DRAM.

Figure 1 is a simplified circuit diagram of the address multiplexer. When /MUX—the control line to the multiplexer-is low, the microprocessor's AO-A7 and P1.O are routed to the AO-A8 address input pins of the 41256. When /MUX is high, the microprocessor's A8–A14, **P1.l,** and **P1.2** are routed to the 41256's **AO-A8** pins.

Figure 2 shows the timing for /RAS, /MUX, and /CAS. /RAS first goes active when /MUX is low, indicating the low-order address bits are being presented to the DRAM. Next, /MUX goes low and /CAS is asserted to tell the DRAM that the high-order address bits are being presented. These three signals are generated by a PAL 22V10. I will describe the details of the PAL's contents later.

HOW TO READ AND WRITE DRAM

The 805 1 has /RD and /WR signals for data memory read and write timing control. /WE, which is the write-enable signal for the DRAM, has to be valid after /MUX and before /CAS become active. Figure 3 shows DRAM read and write timings. The DRAM read/write sequence of events is as follows:

1. Al5 is low (address = OOOOH to 7FFFH) and /MUX is low
2. /WR is low to write; /RD is low to read
3. /RAS goes low to latch the row address
4. /MUX goes high to switch the DRAM address bus from the column to row address
5. /WE goes low to write to DRAM or it goes high to read from DRAM
6. /CAS goes low to latch the column address

HOW TO REFRESH DRAM

The information contained in the internal storage cells of dynamic RAM must be accessed periodically to keep it valid. Typically, the information in a DRAM storage cell remains valid for only a few milliseconds. If the cell is not refreshed, the data is lost.

When data is read from the 41256, an entire row of the internal cell is refreshed in parallel. An entire RAM chip can be refreshed by accessing A0–A7 during a 4-ms refresh period. To do this, we need to have a refresh counter, which can be implemented in software or hardware. However, to avoid the software or hardware overhead necessary to implement such a refresh counter, we can instead use the */CAS before IRAS* refresh method.

Most DRAM currently available on the market has a built-in refresh counter and refresh-timing generator. The internal refresh clocks and refresh counters can be initiated by special timing in the /CAS and /RAS signals. From Figure 2, we know that read or write timing normally requires that /RAS be active before /CAS. If /CAS is active before /RAS, though, the DRAM internal decoding logic will trigger the internal refresh clock and counter. Consequently, a simple

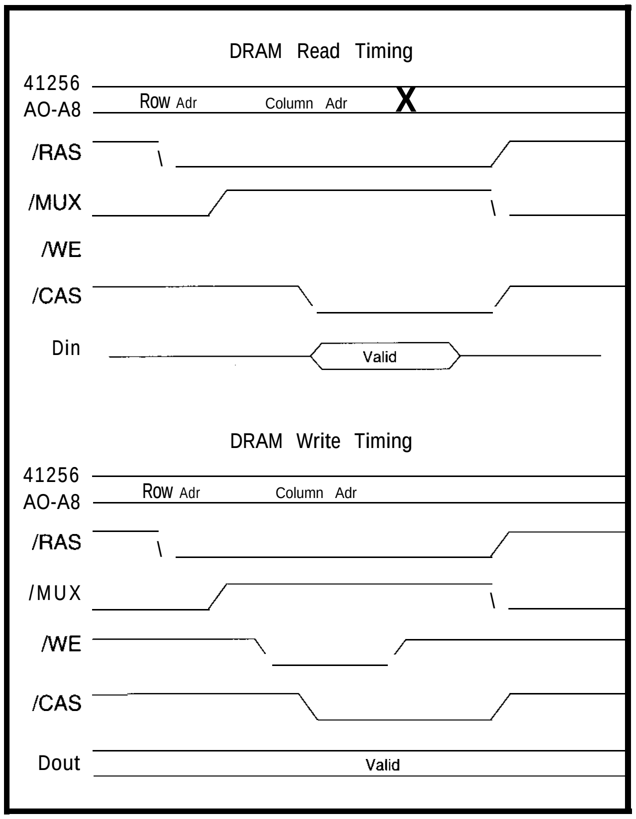
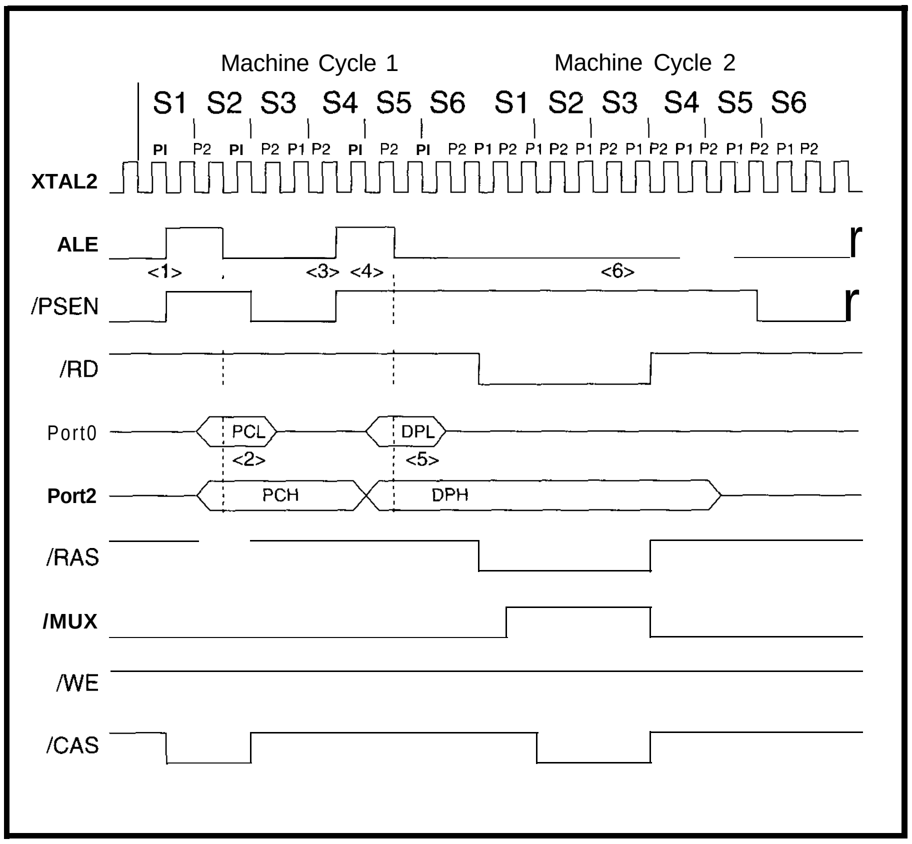


Figure 3—To access the DRAM, we have to prepare the row address, activate /RAS, switch to the column address, set up /WE, and activate /CAS.



Machine Cycle 1
Machine Cycle 2
S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6
PI P2 PI P2 P1 P2 PI P2 PI P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2
XTAL2
ALE
<1>
<3> <4>
<6>
/PSEN
/RD
Port0
PCL
DPL
<2>
<5>
Port2
PCH
DPH
/RAS
/MUX
/WE
/CAS

Figure 4—*When* **DRAM is read by the CPU, the** *signals—/RD, /RAS, /MUX, and /CAS—are generated by a state machine and depend on the machine cycle and its state and phase.*

DRAM refresh circuit can be realized by carefully arranging the /CAS and /RAS timing.

BUS TIMING IN 8051

Read and write timing for program and data memory is critical for the 805 1 controller. To accomplish DRAM refresh by cycle stealing, we have to understand the timing of the 8051 bus; otherwise, we cannot determine where to steal cycles.

Unlike other CPUs, the 805 1 is optimized for control applications. It has 64 KB of program memory address space and another 64 KB of data memory address space. Because the 805 1 doesn't store program code in data memory, a program opcode only has to be retrieved from program memory. During an opcode fetch, data memory is not used. We can, therefore, steal opcode cycles for DRAM refresh.

The 8051 bus timing is divided into machine cycles. A machine cycle consists of a sequence of six states, numbered S1–S6. Each state time lasts for two oscillator periods. Thus, a

Everything You Need for Embedded 80x86 Development

Debug code from your existing C/C++ compiler

- **Soft-Scope supports compilers from Microsoft, Borland, Watcom, MetaWare and Intel**
- **For real- and protected-mode 16- and 32-bit embedded applications**
- **Supports entire 80x86 family including 80486 and i386EX**
- **Available as a DOS .exe or Windows™ application**

Embed a DOS or Windows executable file

- **CSi-Locate supports C/C++ compilers from Microsoft, Borland and Watcom**
- **Creates HEX, binary and absolutely located files**

Royalty-free monitor with complete source code

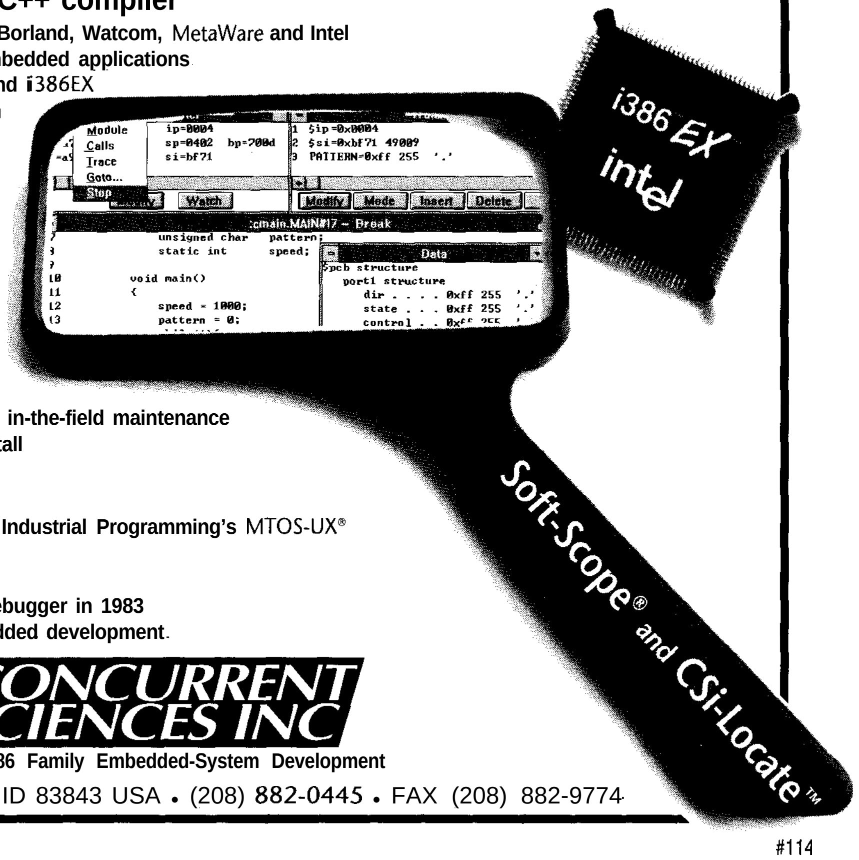
- **Include the monitor in your application for easy in-the-field maintenance**
- **Configuration utility makes monitor quick to install**

Debug real-time applications

- **Supports Intel's iRMX®, JMI's C EXECUTIVE® and Industrial Programming's MTOS-UX®**

Since 1983

- **We introduced the original source-level 8086 debugger in 1983**
- **We are experts in 32-bit protected mode embedded development.**



```
ip=0004                   1 $ip=0x0004
sp=0402  bp=700d          2 $si=0xbf71 49009
si=bf71                   3 PATTERN=0xff 255 '.'
:cmain.MAIN#17 - Break
     unsigned char   pattern;
     static int      speed;     $pcb structure
                                  port1 structure
   void main()                      dir . . . . 0xff 255 '.'
   {                                state . . . 0xff 255 '.'
     speed = 1000;                  control . . 0xff
     pattern = 0;
```

CSi CONCURRENT SCIENCES INC

Bringing Insight into 80x86 Family Embedded-System Development

530 S. Asbury • P.O. Box 9666 • Moscow, ID 83843 USA • (208) 882-0445 • FAX (208) 882-9774

#114

machine cycle takes 12 oscillator periods or 1 µs if the oscillator frequency is 12 MHz.

Figure 4 illustrates the external-data-memory read timing. XTAL2 represents the oscillator clock timing. ALE (address latch enable) outputs a pulse to grab a low byte of the address during access to external memory. At point <1>, ALE latches the CPU program counter low (PCL) byte. This combines with the program counter high (PCH) byte at <2> to provide a 16-bit program memory address. EPROM data (opcodes) will be sampled at P1 of S4 (<3>). At <4>, ALE latches the DPL (data pointer low) byte value or register bits for other instructions. Combining with the DPH (data pointer high) byte data at <5>, a 16-bit data memory address is ready. The external data memory (DRAM) will be sampled at P1 of S3 (<6>). /RD is active for the data read control.

Figure 5 shows external-data-memory write timing. When /WR is active, data to be stored in memory is present on Port 0.

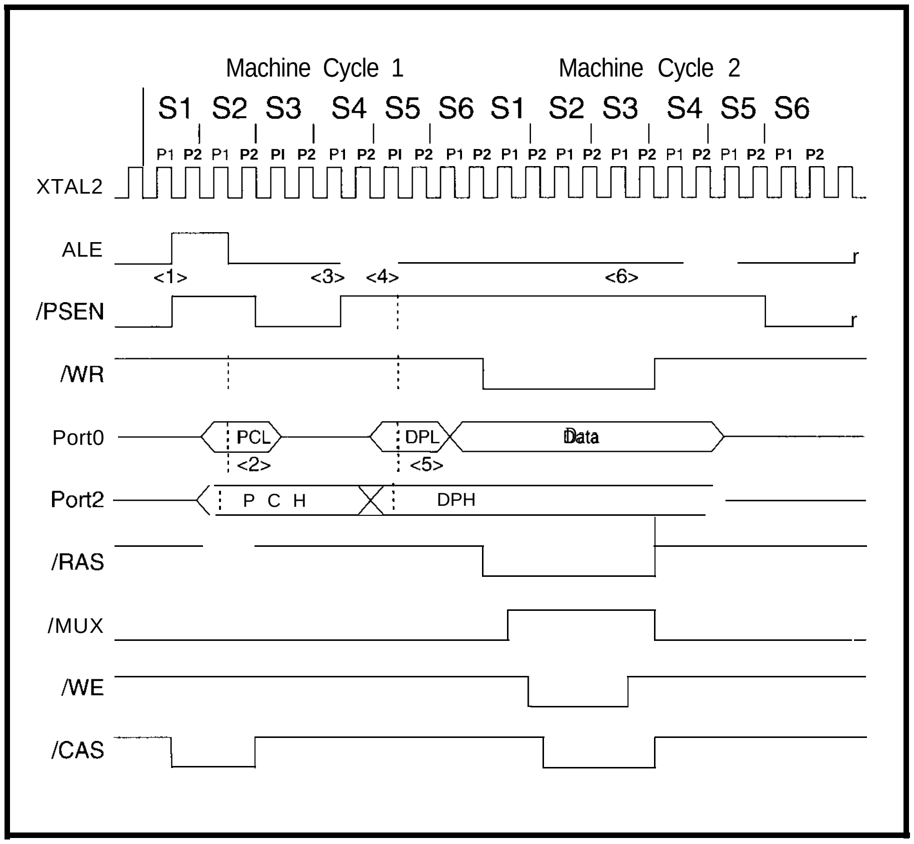


Figure 5—The data memory write cycle is similar to the read cycle, but the DRAM/WE signal is asserted to indicate that data is being written to memory.

LAST CHANCE DESIGN CONTEST

ENTRIES ARE DUE

SEPTEMBER 16, 1994

To receive an Entry Form and a complete set of rules, write or call Rose at:

CIRCUIT CELLAR DESIGN CONTEST

4 Park St., Suite 90 • Vernon, CT 06066
Tel: (203) 875-2199 • Fax: (203) 872-2204

Enter as many projects as you like, but each entry must be accompanied by an official Entry Form.

DRAM READ/WRITE TIMING AND EQUATIONS

In order to make the PAL equations easy to match up with the timing shown in Figures 4 and 5, I define the bus states as S11 for P1 of S1, S12 for P2 of S1, and so forth through S62 (see Table 1). I'll discuss how to generate S1 **1** through S62 later. The PAL equations are shown in Listing 1.

First I define a general-purpose DRAM chip select signal for use inside the PAL as follows:

CSDRAM = /A15 • /WR + /A15 * /RD

During a data memory access, /RAS is the first signal we need to activate and is programmed to be active during S11, S12, S21, S22, S31, and S32 of machine cycle 2. The equation for /RAS is:

/RAS = CSDRAM * (S11 + S12 + S21 + S22 + S31 + S32) + REFRAS

where REFRAS is the /RAS refresh condition (to be discussed later). In Figure 6, /RAS is connected to eight 41256 chips.

/MUX is changed one clock cycle after /RAS. If a 12-MHz oscillator is used, this clock cycle is 83 ns long and is known as the /RAS address hold time (t_{CAH}). Normally, DRAM requires a minimum t_{CAH} of 10–20 ns, so we are well within spec. From this analysis, we can program /MUX to be active during S12, S21, S22, S31, and S32 of machine cycle 2. The following equation represents the /MUX signal:

/MUX = CSDRAM • (S12 + S21 + S22 + S31 + S32)

As you can see in Figure 6, /MUX controls three 74HC244s. It switches the DRAM's AO-A8 between the processor's AO-A7, P1.0 and A8–A14, P1.l, P1.2.

If the data memory access is a DRAM write, /WE is asserted next (see Figure 5). The DRAM spec calls for a minimum delay from the leading edge of /MUX to the leading edge of /WE, t_{WCS}, to be 0 ns. Programming /WE to be active during S21, S22, and S31 (one

Listing I--The *PAL source listing describes the equations for a PAL22V10 which is used to implement a 13-state, Gray-code state machine that is decoded to generate control signals for the DRAM access circuit.*

```
TITLE  DRAM8051
PATTERN  DRAM  REFRASH  FOR  8051
REVISION  0.1
AUTHOR  HUGO  CHEUNG
COMPANY
DATE  1-31-93

; THIS  PAL  GENERATE  CAS  RAS  WE  SIGNAL  FOR
 ;  DRAM  41256  READ  WRITE  ACCESS  AND  ALSO
; CAS  BEFORE  RAS  REFRESH  SIGNALS  FOR  uP  8051
 ;    P0 = 0  0  0  0  0  0    0  1  1  1  1  1  1
;   P1 = 0  0  0  1  1  1  1  1  1  1  1  0  0
;   P2 = 0  1  1  1  1  0  0  0  0  1  1  1  1
;   P3 = 1  1  0  0  1  1  0  0  1  1  0  0  1

CHIP DRAM8051 PAL22V10

;PINS
;1      2   3   4   5        6   7   8       9   10    11  12
PALCK  ALE  RD  WR  PSEN  A15  A14  RESET  NC  NC     NC  GND

;13     14  15  16  17     18  19  20      21  22     23  24
NC      PO   P1 P2  P3     MUX RAS CAS     WE  ADCE   NC  VCC

GLOBAL

STRING CSDRAM '(/A15*/WR+/A15*/RD)'  ;P ST N ST
STRING  IDLE   '(/PO*/P1*/P2* P3)'   ;0001 0011
STRING S12     '(/PO*/P1* P2* P3)'   ;0011 0010
STRING  S21    '(/PO*/P1* P2*/P3)'   ;0010 0110
STRING S22     '(/PO* P1* P2*/P3)'   ;0110 0111
STRING S31     '(/PO* P1* P2* P3)'   ;0111 0101
STRING  S32    '(/PO* P1*/P2* P3)'   ;0101 0100
STRING S41     '(/PO* P1*/P2*/P3)'   ;0100 1100
STRING  S42    '( PO* P1*/P2*/P3)'   ;1100 1101
STRING S51     '( PO* P1*/P2* P3)'   ;1101 1111
STRING  S52    '( PO* P1* P2* P3)'   ;1111 1110
STRING S61     '( PO* P1* P2*/P3)'   ;1110 1010
STRING S62     '( PO*/P1* P2*/P3)'   ;1010 1011
STRING S11     '( PO*/P1* P2* P3)'   ;1011 0001
STRING REFCAS '((S12+S21+S22)*RD*WR)';CAS REFRESH SIGNAL
STRING REFRAS '((S21+S22)*RD*WR)'    ;RAS  REFRESH  SIGNAL

EQUATIONS

;PRS  STATE:  IDLE    S12 S21 S22 S31 S32  S41  S42  S51 S52 S61 S62 S11

PO  := (                           S41+S42+S51+S52+S61+S62     )*/RESET
P1  := (              S21+S22+S31+S32+S41+S42+S51+S52             )*/RESET
P2  := (IDLE*/WR+S12+S21+S22                    +S51+S52+S61+S62+S11)*/RESET
P3  := (IDLE             +S22+S31         +S42+S51         +S62+S11)+ RESET

/RAS = CSDRAM*(S11+S12+S21+S22+S31+S32)+REFRAS; RD,WR & REFRESH RAS
/MUX = CSDRAM*(S12+S21+S22+S31+S32)
/WE  = CSDRAM*(S21+S22+S31)*/WR
/CAS = CSDRAM*(S22+S31+S32) +REFCAS            ; RD,WR & REFRESH CAS

SIMULATION
TRACE-ON  PALCK  ALE  RD  WR  RAS  MUX  WE  CAS  ADCE  A15  A14  PO  P1 P2  P3  RESET
SETF      /PALCK /ALE RD WR /A15 /A14 RESET
CLOCKF    PALCK
SETF      /RESET
        CLOCKF  PALCK   ;S41 PHASES BEFORE RD OR WR
        SETF    ALE  PSEN
        CLOCKF  PALCK   ;S42
```

(continued)

clock cycle later than /MUX) gives a t_{WCS} of 83 ns, so again is well within spec. The equation for the /WE signal is:

/WE = CSDRAM * (S21 + S22 + S31) • /WR

During a DRAM read, /WE is not active. As shown in Figure 6, /WE from the PAL is connected to /WE inputs of the eight 41256 chips.

The last step involves asserting /CAS and is done at the same time during both read and write cycles. The equation for /CAS is:

CAS = CSDRAM * (S22 + S31 + S32) + REFCAS

Similar to the /RAS signal, /CAS is connected to all eight 41256 chips.

DRAM REFRESH TIMING AND EQUATIONS

According to the DRAM specification, the refresh rate has to be 256 times per 4 ms. That is, the refresh must occur once every 4 us. Can this be done using just software?

Executing M 0 V X instructions continuously would result in DRAM accesses every other machine cycle, or once every 24 oscillator periods (remember, there are 12 oscillator periods per machine cycle). That translates to an access every 2 us. While we could conceivably keep memory refreshed using this technique, it wouldn't leave much time to do any useful work. Let's let the PAL do the refresh for us.

The only time data memory is accessed during the second machine cycle of an instruction (as shown in Figures 4 and 5) is with the MOVX instruction. With all other instructions, the CPU never accesses data memory during clock cycles S11, S12, S21, and S22, so we can do our refresh during these cycles. This technique is known as cycle ***stealing.***

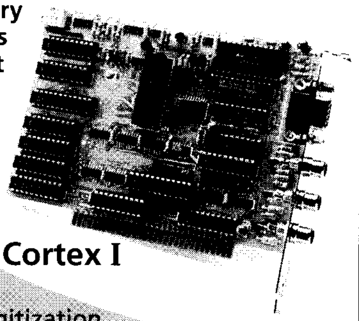
We need some way to detect when instructions other than MOVX are being executed so we can start a refresh cycle. /RD and /WR can be active only during S1 l-S32 of the second machine cycleofaMOVX,sowecanuse/RD=O

Listing l-continued

```
        CLOCKF  PALCK    ;S51
        SETF    /ALE
        CLOCKF  PALCK    ;S52
        CLOCKF  PALCK    ;S61
        CLOCKF  PALCK    ;S62

        SETF    /WR      ;INIT PHASE COUNT AND WRITE TO DATA MEMORY
FOR I:= 1 TO 6 DO BEGIN;S11-S32
        CLOCKF  PALCK
END

        SETF   WR
FOR I := 1 TO 5 DO BEGIN
        CLOCKF  PALCK    ;S41 FETCH OR PREFETCH
        SETF    ALE PSEN
        CLOCKF  PALCK    ;S42
        CLOCKF  PALCK    ;S51
        SETF    /ALE
        CLOCKF  PALCK    ;S52
        SETF    /PSEN
        CLOCKF  PALCK    ;S61
        CLOCKF  PALCK    ;S62
END

        SETF    /WR A15 ; WRITE TO ADC
FOR I:= 1 TO 6 DO BEGIN;S11-S32
        CLOCKF  PALCK
END
TRACE-OFF
```

/Video Frame Grabber

- $495 Including Software with "C" Library
- Half Slot Card for Compact Applications
- Real Time Imaging with Display Output
- 8 Bit (256 Gray Levels)



FEATURES:

- Single 512 X 484 Image or Four 256 X 242 Images
- External Trigger
- Input Look Up Table
- Low Power Option Available
- Elegant Software Interface
- <10 nsec Pixel Jitter Means Accurate Digitization
- EISA (PC) Bus and STD Bus Products Available
- RS-170 and CCIR Video Formats Available
- Binary and TIFF File Formats

The Cortex I is used in machine vision, industrial control, medical, security and scientific applications around the world.

ImageNation strives to delight customers with quality products and personal service at a competitive price.

Call today for volume pricing or to discuss your application.

ImageNation Corporation
Providing Imaging Solutions

P.O. Box 276
Beaverton, OR 97075
(503) 641-7408

FAX (503) 643-2458 • (800) 366-9131

#115

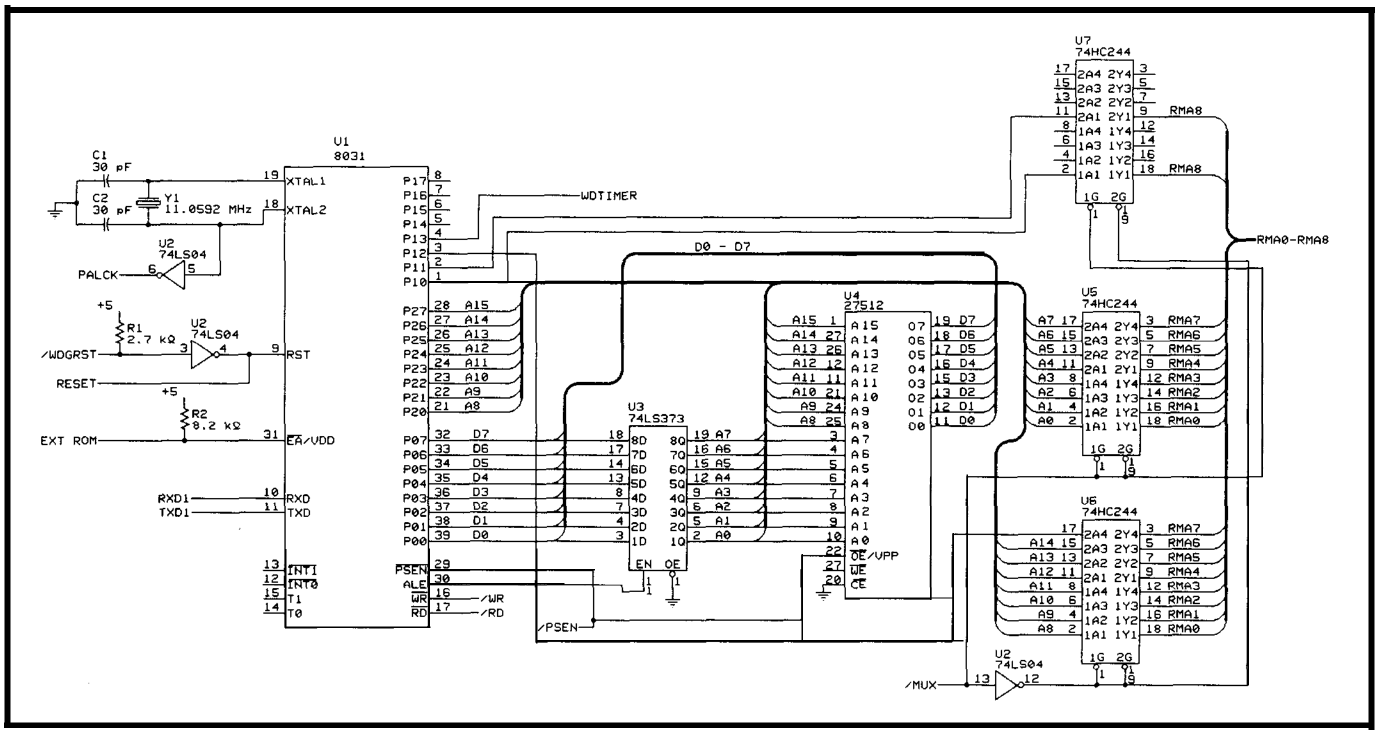
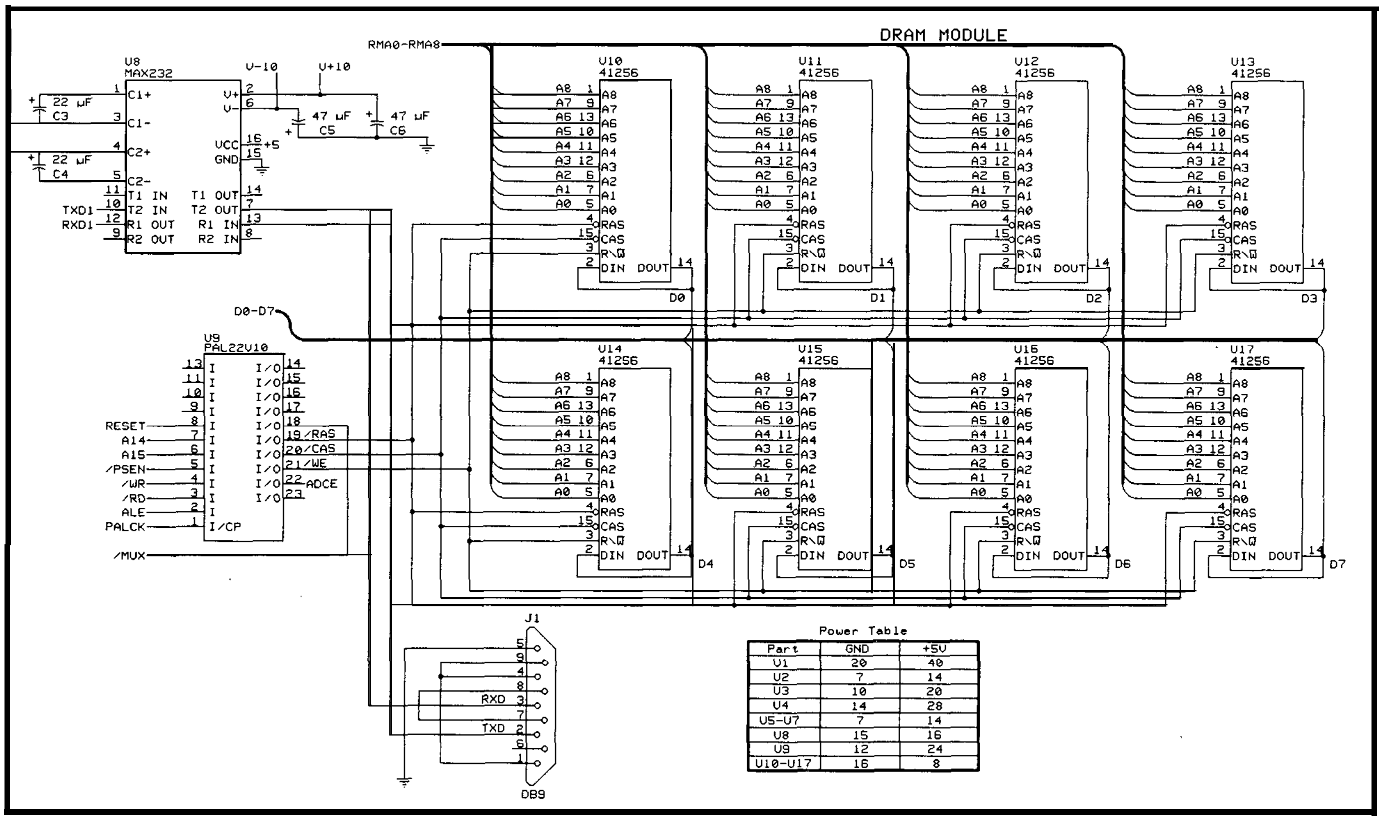


Figure 6a—*This* is a common 8031 circuit with a DRAM address multiplexer. Three *I/O* bits and *15* address *lines are* used to generate the DRAM *row and column* addresses. Six *more pins on U7 are* available *for further DRAM expansion.*

or /WR = 0 to indicate "no refresh." When both /RD and /WR are high, we know we can perform a refresh.

As I mentioned before, /CAS has to be active before /RAS for "/CAS before /RAS" refresh. The /CAS refresh signal, REFCAS, is active during S12, S21, and S22. The /RAS refresh signal, REFRAS, is active during S21 and S22. So the final equations for REFCAS and REFRAS are:

REFCAS = (S12 + S21 + S22) * RD * WR



Power Table

Part	GND	+5V
U1	20	40
U2	7	14
U3	10	20
U4	14	28
U5-U7	7	14
U8	15	16
U9	12	24
U10-U17	16	8

Figure 6b—*Eight* 41256 *DRAMs are implemented* in *this application. Other DRAM sizes or* configuartions *(e.g., two 4 x 256 Kb* DRAMs *or DRAM modules) can be used with minor modifications.*

CPU States	S1	S2	s3	s4	s5	S6
OSC Phases	P1/P2	P1/P2	P1/P2	P1/P2	P1/P2	P1/P2
Detail States	SI I/S12	S21/S22	S31/S32	S41/S42	S51/S52	S61/S62

Table I--The twelve states of *each* 8031 machine *cycle are named S11–S62. With these new names, if is easier to follow the structure of fhe state machine and PAL source code.*

and

REFRAS = (S21 + S22) * RD * WR

OSCILLATOR CYCLE STATE MACHINE

The 803 l's twelve clock states, S1 l-S62, are tracked within the PAL using a state machine (see Figure 7). 1 tap off the processor's main oscillator (XTAL2), buffer the signal with a 74LS04, and pass that signal to the PAL to step the state machine on each falling edge of the oscillator.

Now that 1 have a raw clock input, how do 1 synchronize the state machine with the CPU so 1 know what state the processor is in?

After the power-on reset, the state machine is idle. It will exit its idle state when it receives a /WR signal from the processor. /WR is only active on Sl **1,** where other signals such as ALE or /PSEN can be active during multiple states (S12 or S42). Once started, the state machine counts through all the other states and repeats. Your software must issue a dummy instruction such as "MOV X @R0 , A" right after reset to generate a /WR signal and start the state counter.

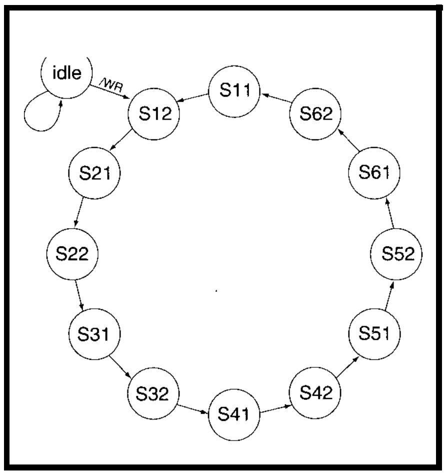


Figure *7—This state machine, which is triggered by fhe /WR strobe, tracks the* CPU's *twelve machine cycles and generafes DRAM read, write, and refresh control signals.*

In order to avoid possible race conditions and glitches in the PAL's latches that are used to count the state machine, 1 use Gray code values. Using Gray code, you are assured that only one bit will change at a time when stepping from one value to the next.

The PAL's reset input is connected to the system reset, and, when this signal is active high, the state machine will be forced into an idle state.

1 developed the PAL code using AMD's PALASM and include a test program in the source file.

IS IT ENOUGH?

Ten years ago, when the Apple 11 was very popular, 64 KB of DRAM was a lot of memory for applications of the day. 1 could run WordStar, Pascal, and many other programs. Today, 1 complain about the 640 KB on my being too small. In contrast, some people may think that 256 KB or more RAM for 803 1 is too much. However, no matter how much memory the hardware can provide, there are always some applications, such as a printer buffer or data logger, that will consume it all and then some. So, good luck on experimenting with beefing up your DRAM. ❑

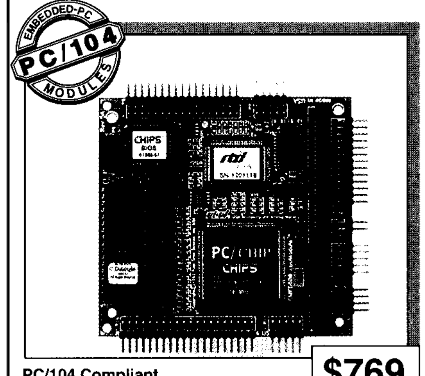
Hugo Cheung is a Ph.D. student at the University of Southern California and is currently a systems engineer at Rockwell Telecommunications located in Newport Beach, CA. His interests include embedded controls, DSP applications, ASIC design, and ASIC testing. He may be reached at cheung@nb.rockwell.com.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

Replace Four Conventional PC/l 04 Modules with *One* CMF8680 cpuModule™

Embedded PC/XT Controller with Intelligent Power Management



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

$769

- PC/XT compatibility with 286 emulation
- 14 MHz, 16-bit C&T F8680 CPU
- +5V only; 1.6W at 14.3 MHz, 1W at 7.2 MHz
- Intelligent sleep modes, O.lW in Suspend
- ROM-DOS and RTD enhanced BIOS
- Compatible with MS-DOS & real-time operating systems
- 1 M bootable solid state disk & free software
- 4K-bit configuration EEPROM (2K for user)
- 2M on-board DRAM
- IDE & floppy interfaces
- CGA CRT/LCD controller
- Two RS-232 ports, one RS-485 port
- Parallel, XT keyboard & speaker ports
- Optional X-Y keypad scanning/PCMCIA interface
- Watchdog timer & real-time clock

Expand This Or Any PC/l 04 System with the

CM106 Super VGA Controller utilityModule™

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

$395

Speed Product Development with the

SK-CM106-X Starter Kit

Your kit includes the CMF8680 cpuModule, CM106 SVGA controller, CM102 keypad scanning/PCMCIA utilityModule, CMF8680 cable kit & VGA monitor cable for just $1295.

Additional PC/104 compliant modules from RTD:

- CM104 1.8" hard drive carrier utilityModule
- 12- & 14-bit analog I/O modules
- 12-bit, 4-20 mA analog output modules
- opto-22 & digital I/O modules

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

Real Time Devices USA

200 Innovation Blvd. • P.O. Box 906
State College, PA 16804 USA
(814) 234-8087 / Fax: (814) 234-5218

RTD Europa • RTD Scandinavia

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

16

FEATURE ARTICLE

Ron Stence

Motorola's 68322 Processor: Redefining the Low-end Laser Market

Chip integration has consistently reduced the size and price of devices while improving their speed, reliability, and features. The laser printer is no exception. This new chip could revolutionize the low-end market.

Technological improvements in the microprocessor industry are changing the embedded system environment of the '90s. Key microprocessor manufacturers are introducing with an ever-increasing intensity integrated products with heightened capabilities. These new products will offer throughout the decade a multitude of choices for system designers in all areas of the embedded control marketplace.

In the past, general-purpose microprocessors were integrated into embedded designs with little insight from microprocessor manufacturers about how to improve the overall system. Today, microprocessor designers are looking for the balance between a set of features, cost, and design cycle time while system designers seek more performance at a lower system cost.

The 68322 is an example of Motorola's response to the needs of the embedded control market. The chip is the first of Motorola's 68000 family microprocessors specifically designed to redefine the low-end laser printer market.

The market for laser printers is composed of three segments: high-end, mid-range, and low-end printers. High-end printers use high-performance CPUs for enhanced networking capability, higher resolution, and, most importantly, faster printing. As the price of high-end printers drops,

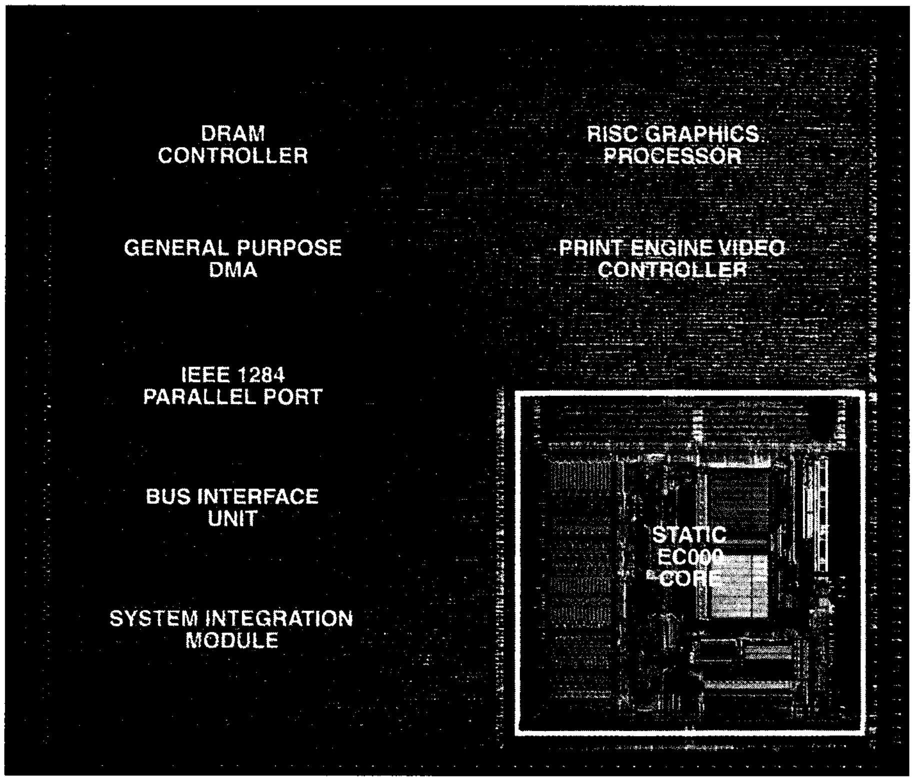
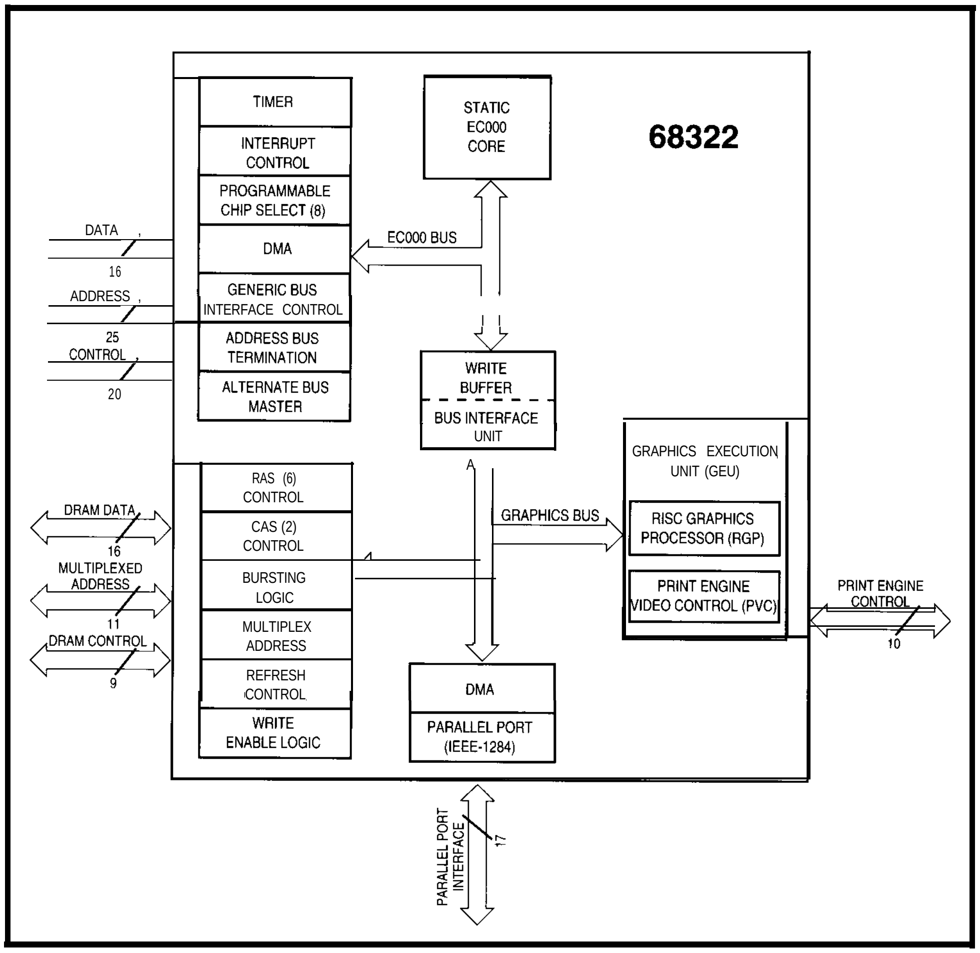


Photo 1—*By squeezing more onto a single piece of silicon, Motorola hopes to further drive down the price of low-end laser printers.*

Figure 1—*The* 68322 uses a *dual-bus, dual-processor architecture. It also contains IEEE-1284 parallel port, printer engine, DRAM interfaces to virtually eliminate the need for external glue logic.*



and low-end printers gain more power and additional features, manufacturers of mid-range laser printers are challenged to compete with lower prices and additional features.

A fourth market segment—Windows printing or graphics device interface (GDI)-will emerge in 1994. Manufacturers in the GDI market will integrate a less powerful processor in a laser printer which will rely on a host computer for much of the CPU-intensive rasterization. The host computer will generate a bitmap image or an image composed of low-level graphic commands that will be down-loaded to the GDI laser printer for printing.

68322 TECHNICAL INFORMATION

The integrated laser printer microprocessor, 68322, is designed specifically for the GDI and low-end laser printer markets. Several key features make the 68322 well-suited for personal printers.

Figure 1 shows the general layout of the 68322. With a true dual-bus, dual-processor architecture, the 68322 fetches instructions for the 68ECOO0 core from the 68322 bus while the graphics bus performs operations on text and graphics images in the DRAM from the second bus. This feature provides a significant performance improvement while operating from the dual 16-bit buses. A specialized graphics accelerator is on the chip to interpret graphics commands known as ***display lists.*** The graphics unit is divided into a RISC graphics processor (RGP), which executes or interprets graphics commands from DRAM, and a print engine video controller, which handles the print engine interface and video transfers from DRAM to the print engine.

The RGP is a powerful accelerator containing several registers. After being initialized, the RGP interprets the display list and creates a page, one band at a time, with no intervention from the 68EC000 processor. The print engine video controller acts as an intelligent DMA unit transferring the video from DRAM to an internal FIFO and out the video port without interaction from the 68EC000 processor. The 68EC000 processor compresses the video image using a run-length encoded (RLE) compression technique. The RGP automatically decompresses the video, and the print video controller sends the data to the print engine. These dedicated features remove the burden from the processor and use dedicated hardware to dramatically improve performance and reduce system cost.

The bursting DRAM interface supports CPU data structures, graphics rendering, bit-block transfers, DMA transfers, and display list interpretation while transferring video to the print engine. The DRAM bus is shared with the 68EC000 core, RGP, print video controller, and two DMA units. A single DMA unit is dedicated to the support of a bidirectional, IEEE- 1284-compatible, parallel communication port. The second DMA channel transfers data between the processor bus and the DRAM bus.

Modifications to the 68000 processor bus result in a glueless interface to virtually all peripherals. For example, separate read-and-write strobes are added to the system integration module to support non-Motorola peripherals. The chip selects are programmable to provide individual setup-and-hold and recovery times for each of the eight banks.

The 68322 video interface requires no external glue logic to interface with most laser printer engines. The 68322 is so versatile that it can be used in inkjets, facsimile machines, and even in nongraphics applications that require DMA, DRAMs, and a general-purpose 68000 processor.

In addition, the 68322 has new environmental green features designed into the device. The 68322 uses a new static 68000 core processor. With the ability to stop the clock, the device will lend itself to hand-held and low-power applications. The SmarToner option has been designed into the 68322 to reduce the amount of toner applied to the page when printing in a draft mode.

THEORY OF OPERATION

The primary advantage of the 68322 processor is its capability to reduce the amount of memory (one of the most costly devices in a laser printer) required to print a page. The 68322 features banding and video image compression to ultimately reduce manufacturing and consumer costs--two major considerations when designing for the low-end laser printer market.

Banding allows the raster image processor (RIP) board to store the page in data strips. When banding, a full page does not reside in memory at one time. Instead, the video image consumes only two bands of memory. These bands range in size from less than 10 KB to a full page. Typically, a band is 32-64 KB. The 68322's RGP interprets the display list or graphics orders into a band of video image. By using a banding technique, the printer dramatically reduces the memory required to print a page. The 68322 prints a banded 600-dpi page at full speed, 6-8 ppm, using less memory than conventional printers. Roughly 1 MB of memory is required to hold the rasterized data of a single page at 300 dpi. When the resolution increases to 600 dpi, approximately 4 MB are required if there is no memory compression or banding.

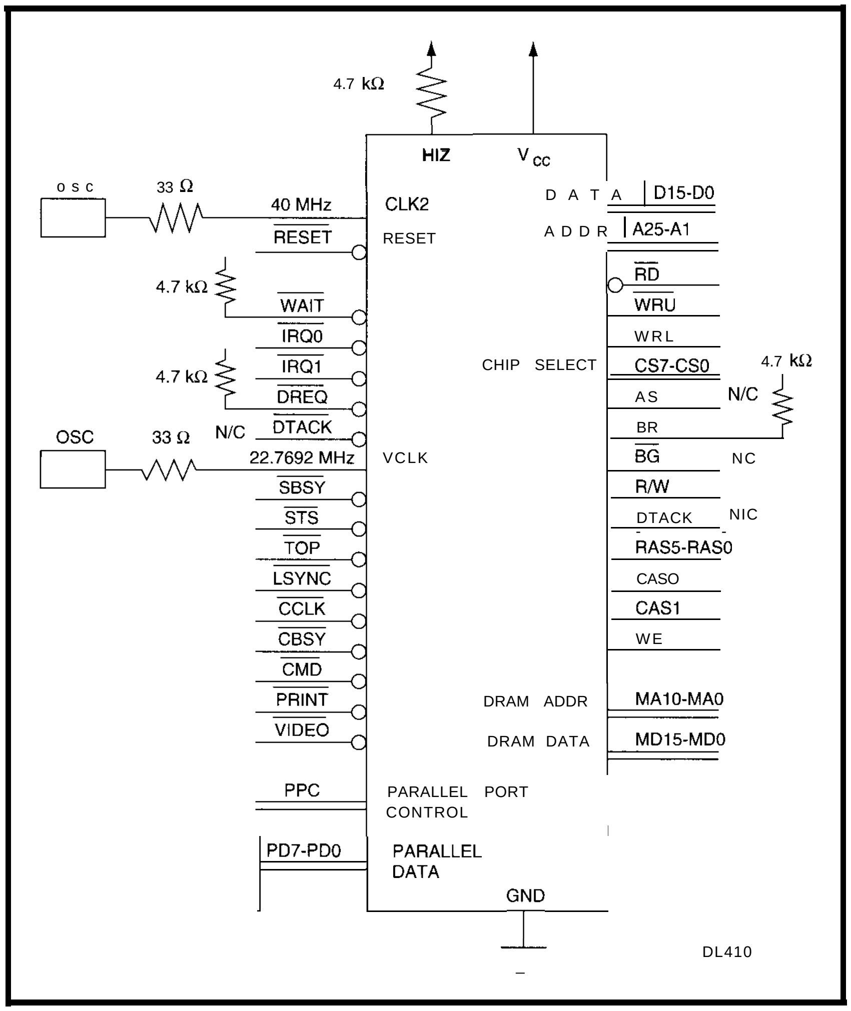


Figure 2-By including *all* the most-needed *electronics* on *the* chip, the 68322 requires *very little* external *support circuitry*

The 68322 uses an additional memory compression technique known as scan-line tables or run-length encoding, which compresses repetitive rasterized data. The scan-line tables are not possible without the 68EC000 and the versatile RGP. The 68EC000 creates the compressed data, and the RGP decompresses the scan-line tables on the fly.

For the 68322 to provide the maximum performance of 8 ppm at 600 dpi, a significant amount of data must be manipulated in real time. A process known as *race the laser* requires the printer controller board to translate display-list graphics orders into a banded, rasterized bitmap image as fast as the printer can place the pixels on the page.

To successfully print from banded memory, three conditions must exist. First, the system must read the complete display list, place the banded, rasterized data into memory, and then retrieve the rasterized data in real time to send to the print engine. The list of instructions and data to be manipulated for printing a page is very long, resulting in a potential problem with bus bandwidth. Because the 68322 has a dual bus architecture, the 68EC000 core can simultaneously fetch instructions for rasterization and video data manipulation.

The DRAM bus bandwidth is 16 MBps at 20 MHz. The data manipulation required to band a page at 600 dpi is approximately 24-40 MB of data. At 600 dpi and 8 ppm, a page prints every 7.5 s. The resulting bus utilization is approximately 2033 %, providing ample margin for time-critical functions.

The second condition required for printing from banded memory is that the RGP must be capable of reading the display list and interpreting graphics commands fast enough to keep up with the video-pixel data being sent to the print engine. To print a page, the RGP must interpret the display list in real time and generate the rasterized data for a complete band. This action must transpire in the amount of time it takes to transfer the preceding band of video data to the print engine.

The third condition is the reduction of required memory by the system. If memory and the resulting cost cannot be significantly reduced, banding is not economical and not worth the investment. When the page cannot be printed in the banding mode, the 68322 has the ability to exit the banding mode and generate a single band that happens to be a full page in length. Additional memory can be installed or a lower-resolution mode can be used to complete a lengthy or graphically intensive print job.

LASER PRINTER ON A CHIP

Designing for a laser printer is similar to designing for other embedded applications-designers must include a processor, memory, peripherals, input stimulus, and output data. What distinguishes a laser printer from other applications is the amount of code that must be executed and the amount of required memory to perform the task.

Currently, the PROMs inside a laser printer range from 1 MB to more than 2 MB, depending on the number of available fonts and supported

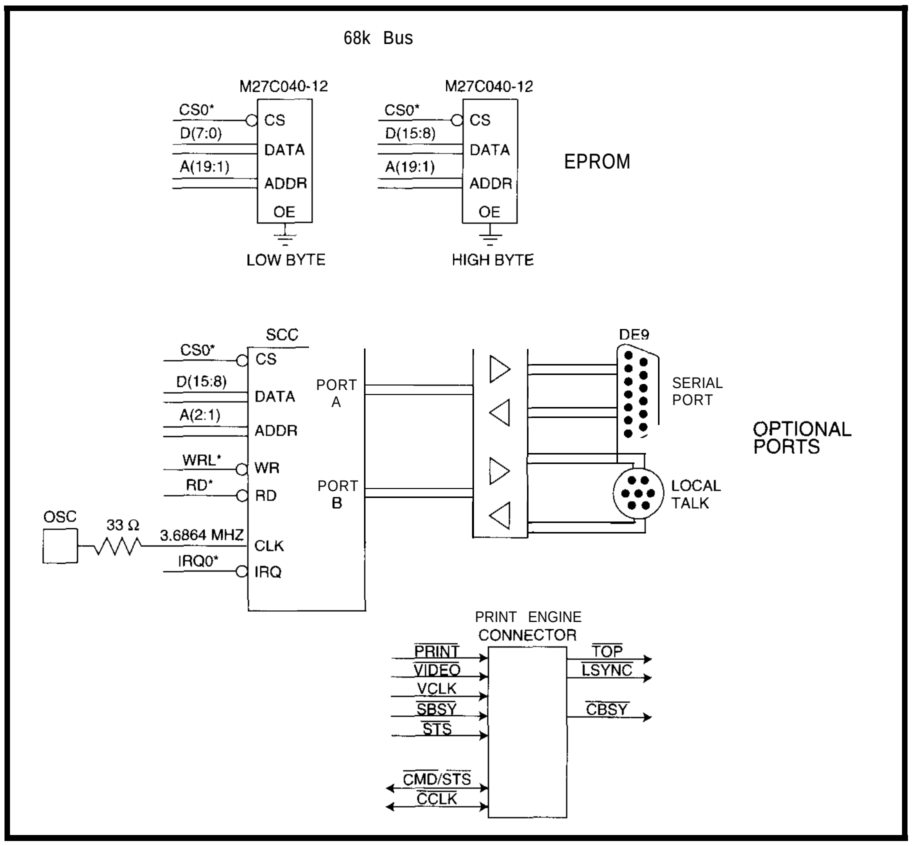


Figure 3—*The* 68322 *directly supports* a *standard* parallel *port, and also supports a serial port and a LocalTalk port with the* addition *of a* serial communications *chip (SCC).*

PLIX™ HARDWARE **X-10**™ TRANSCEIVER CHIP

Micromint introduces its new Power Line Interface for X-10. PLIX is an 18-pin ASIC chip that automatically handles all the specialized X-10 timing and bit-shuffling between a computer and a TW523 power line module.

- ✖ Complete interface between parallel port and TW523 or PL513 modules
- ✖ Performs all X-10 transmit and receive functions
- ✖ Detects AC power loss
- ✖ Simple interface and timing — operates even in interpreter BASIC
- ✖ Low power — only 1.8 mA @ 5 V

PLIX chip and data sheet **$20**
100 qty. OEM **$12**
PLUS SHIPPING

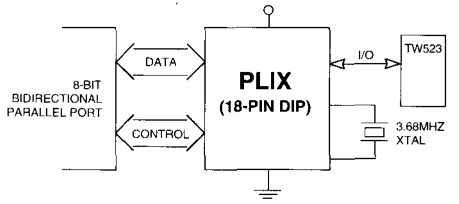
Call **1-800-635-3355**
or Write for a PLIX data sheet

PLIX EVALUATION BOARD

To speed your evaluation, we have produced a low-cost PLIX interface that plugs into a parallel printer port. The PLIX-EKit includes a PLIX chip and data sheet, PCB with all components, application note, and PC-compatible software on diskette.

Add $30 for TW523 X-10 transceiver module

$39.00
PLUS SHIPPING

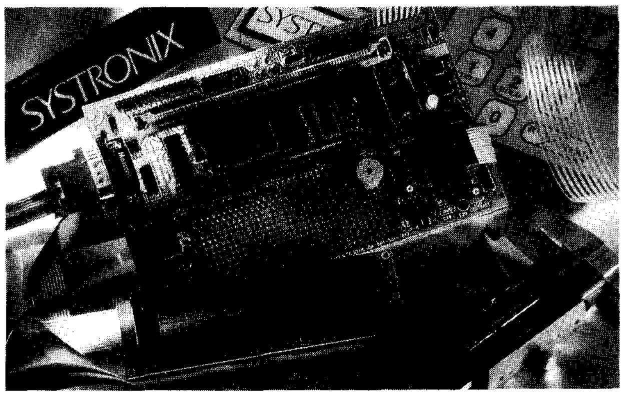


MICROMINT, INC.
4 PARK ST., VERNON, CT 06066
(203) 871-6170 FAX: (203) 872-2204

X-10 is a trademark of X-10(USA), Inc.

#119

NEW! UNIVERSAL DALLAS DEVELOPMENT SYSTEM from **$199!**



- It's a complete single board computer!
- One board accommodates any 40 DIP DS5000, 40 SIMM DS2250, 40 SIMM DS2252, or 72 SIMM DS2251, 8051 superset processor! Snap one out, snap another in.
- Programs via PC serial port. Program lock & encrypt.
- LCD interface, keypad decoder, RS232 serial port, 8-bit ADC, four 300 mA 12V relay driver outputs.
- Power with 5VDC regulated or 6-13 VDC unregulated
- Large prototyping area, processor pins routed to headers
- Optional enclosures, keypads, LCDs, everything you need
- BCI51 Pro BASIC Compiler w/50+ Dallas keywords $399

SYSTRONIX® TEL: 801.534.1017 FAX: 801.534.1019
555 South 300 East. Salt Lake City, UT, USA 84111

#117

languages. Printers shipped today include 512 KB to 16 MB of DRAM. Communications options such as a parallel port, LocalTalk, Ethernet, or RS-232 are common. All print engine controllers have a video port that is typically serial.

The 68322 has integrated a parallel port that complies with the IEEE- 1284 specification and can be directly connected to a host computer. The DRAM interface is glueless with RAS, upper and lower CAS strobes, write enable, multiplexed addresses, automatic incrementing of the low-order address bits, memory decoding, and automatic refresh.

The system integration module (SIM) has a timer, prioritized processor chip selects, byte-write enables, and a separate read strobe to provide direct connections to virtually any peripheral or SRAM found in printers on the market today. The SIM chip selects can be individually programmed for hold and recovery times and can be set up with automatic termination, providing a tremendous amount of flexibility to the processor chip-select logic.

Figures 2, 3, and 4 describe the logic required for a typical 68322-based laser printer. Printer manufacturers provide various configurations of base memory and communications. The 68322 provides the chip select with a versatile DRAM interface to meet the many demands of a constantly changing printer market. A 68322-based design redefines the entry-level laser printer available on the market today. The 68322 innovations will reduce laser printer manufacturing costs while improving the performance and resolution.

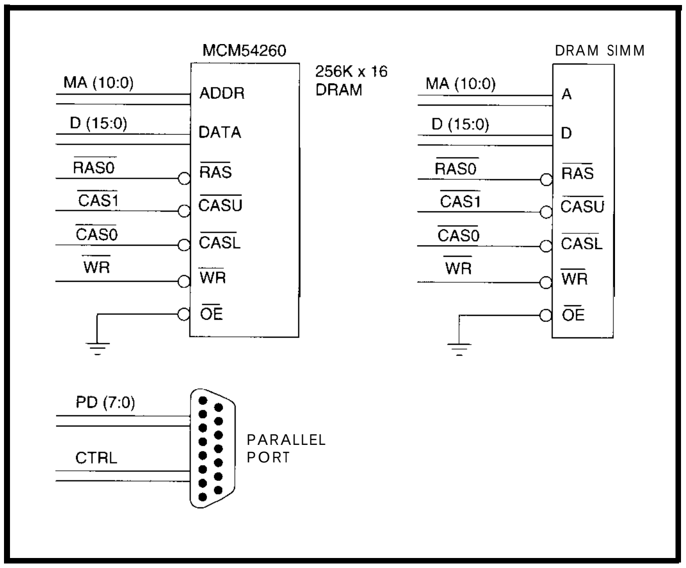


Figure 4—*The* 68322 also *directly supports both DRAM chips and SIMMs in addition to printer engines.*

AN EMBEDDED FUTURE

The microprocessor industry is moving to meet the demands of the evolving printer marketplace. Microprocessor manufacturers are under increasing pressure to be the first to develop new and innovative solutions for the embedded market. Increasing integration and performance will become the norm from the major microprocessor design houses. The 68322 provides large-scale integration, a significant increase in performance, and reduces system cost. It provides a strong option for laser printer designs. ❑

Ron Stence is a senior systems application designer at Motorola. In addition to providing applications support for 68000 products, Ron has been currently focusing his efforts on the design of a 68000-based laser printer.

CONTACT

Motorola Literature Distribution
P.O. Box 20912
Phoenix, AZ 85036

IRS

407 Very Useful
408 Moderately Useful
409 Not Useful

Real- Time Multitasking with DOS

for Microsoft C, Borland C, Borland/Turbo Pascal

Develop Real-Time Multitasking Applications under MS-DOS with ***RTKernel!***

RTKernel is a professional, high-performance real-time multitasking kernel. It runs under MS-DOS or in ROM and supports Microsoft C, Borland C++, Borland/Turbo Pascal, and Stony Brook Pascal+. RTKernel is a library you can link to your application. It lets you run several C functions or Pascal procedures as parallel tasks. RTKernel offers the following advanced features:

- pre-emptive, event/interrupt-driven scheduling
- number of tasks only limited by available RAM
- task-switch time of approx. 6 µs (33-MHz 466)
- performance is independent of the number of tasks
- use up to 64 priorities to control your tasks
- priorities changeable at run-time
- time-slicing can be activated
- programmable timer interrupt rate (0.1 to 55 ms)
- high-resolution timer for time measurement (1 µs)
- activate or suspend tasks out of interrupt handlers
- programmable interrupt priorities
- semaphores, mailboxes, and message-passing
- keyboard, hard disk, and floppy disk idle times usable by other tasks
- interrupt handlers for keyboard, COM ports, and network interrupts included with source code
- supports up to 36 COM ports (DigiBoard, Hostess boards)
- supports protocols XOn/XOff, DTR/DSR, RTS/CTS
- full support of NS16550 UART chip
- supports math coprocessor and emulator
- fast, inter-network communication using Novell's IPX
- runs under MS-DOS 3.0 to 6 x, DR-DOS, LANs, or without operating system
- DOS calls from several tasks without re-entrance problems
- supports resident multi-tasking applications (TSRs)
- runs Windows or DOS Extenders as a task
- supports CodeView and Turbo Debugger
- Kernel Tracer for easy debugging
- ROMable
- full source code available
- no run-time royalties
- free technical support by phone or fax

RTKernel-C 4.0 $495
C Source Code: add $445

RTKernel-Pascal 4.0 $445
Pascal Source Code: add $375

International orders: add $30 shipping and handling.
Mastercard, Visa, check, bank transfer accepted

On Time MARKETING
Professional Programming Tools

In North America, please contact:
On Time Marketing
88 Christian Avenue • Setauket, NY 11733 • USA
Phone (516) 689-6654 • Fax (516) 689-1172
CompuServe 73313.3177

Outside North America, please contact:
On Time Marketing
Karolinenstrasse 32 • 20357 Hamburg • GERMANY
Phone +49 40 43 74 72 • Fax +49 40 43 51 96
CompuServe 100140,633

#118

FEATURE ARTICLE

**Mike Collier &
Fred Gweme**

Preventing the Ultimate Blow: A Portable Checking Unit for 8751s

When you're doing engineering in a country where micro-controllers aren't particularly plentiful, you take great care not to blow up the ones you have. When one appears dead, plug it into this tester for instant results.

Most people think of Zimbabwe as elephants, waterfalls, rhino, and droughts. But behind these features which catch the eye of the media, there are other things going on. Among these is a growing electronics industry. At present it's fairly small stuff, but the opportunities in Southern Africa are quite inviting. One attractive area is the development of products built around microcontrollers. For example, we have developed a small device for measuring water flow in rivers, which uses a fascinating algorithm for salt dilution and conductivity. Parallel processing with 8-bit microcontrollers has been used, and LAN controllers based on the same chips have been investigated. On the heavy current side, locally produced UPSs use the same technology, and work is in progress on the control of electric motors. All of these are very much "appropriate technology" solutions to real problems that we meet in this part of the world.

At the University of Zimbabwe, the electrical engineering students study the 803 1/805 1 series of microcontrollers, and a number of the undergraduate and graduate research projects use this family of devices.

The big difference between hardware development in a land-locked African country and that in the First World is accessibility of components. A development laboratory in

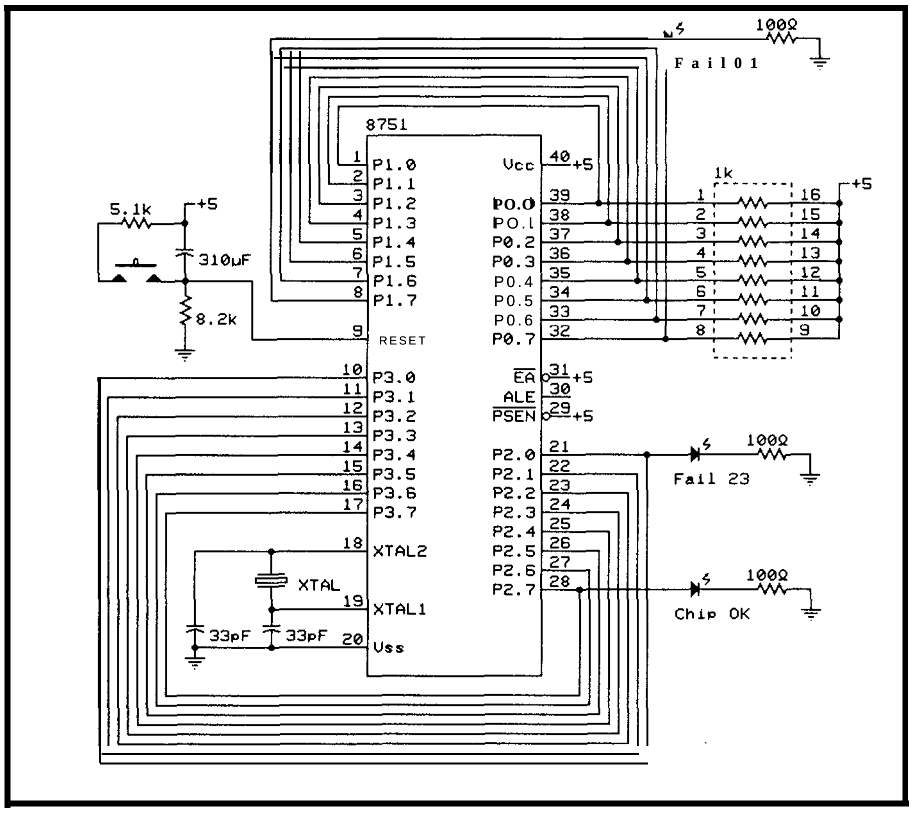


Figure 1--The *chip tester ties the 8751's I/O ports together for the diagnostics and reports its findings on three LEDs.*

the U.S. or Europe would have an unending supply of the crucial chips, but here we often do development with only one or two specimens in hand. Inevitably, we meet the situation in which a prototype doesn't work, and the cry goes up, "The chip's blown." At such a moment, one is loathe to put the last chip in Zimbabwe into the circuit to see if it also self-destructs!

To prevent such catastrophes, a year ago we decided we needed a free-standing tester that could quickly check through all the functions of a microcontroller. It had to be rugged, portable, and "studentproof." The portability requirement did not mean that it would be carried into the bush on a safari; but, it did imply that the unit should comprise a small box that would function as soon as it was turned on.

SPECIFYING THE MACHINE

Our experience with hard-luck stories from students suggested that most cases of damage to microcontrollers resulted from incorrect voltages on the pins. These errant voltages could arise either from the malfunctioning of circuits in which the device was embedded or from static electricity due to incorrect handling of the chips.

Since we were using the 875 1 (EPROM version of the 803 1), we decided to build a tester specifically for this family of chips, which now comes in 4-, 8-, 16-, and 32-KB EPROM sizes. We felt the most likely failure areas in the 875 1 chip to be the EPROM and the four I/O ports of 8 lines each. Since the interrupts, timers, and serial port all make use of particular lines of the four ports, these would automatically be covered by a general test of the I/O.

The tester needed to:

•provide a circuit which would allow a test program in the EPROM to assess the major functions of the device
•offer connections to pairs of I/O pins (preferably in different ports) which would be tested sequentially by using one pair for transmission of a pulse train and the other for reception

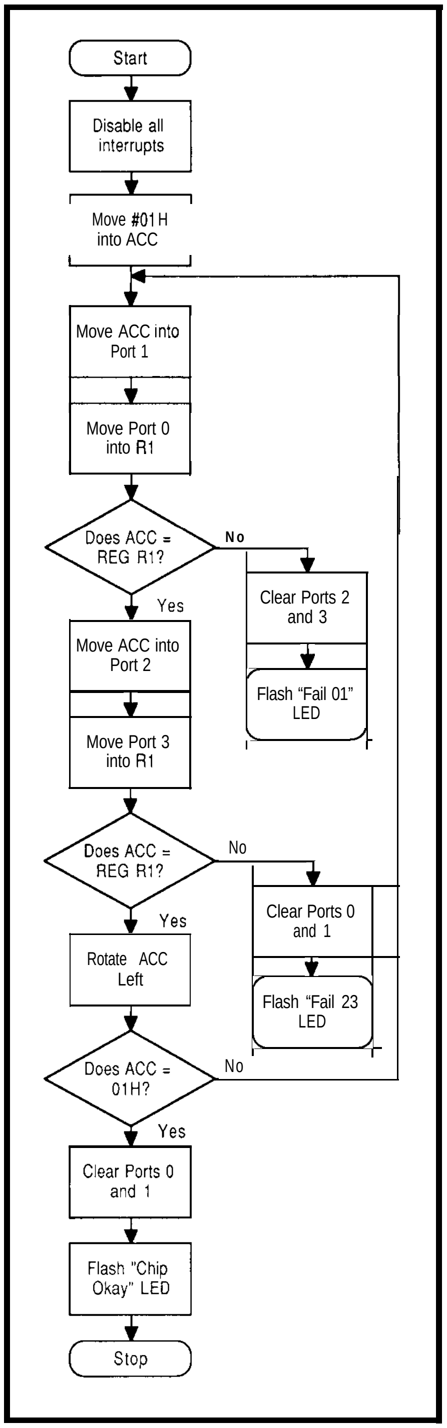


Figure 2—*The basic testing algorithm is quite simple, but is effective at weeding out bad parts.*

*output to LEDs, which would either indicate failure of a port pair or a successful test (the indicators must signal failure of the pins driving the LEDs when the test has failed)

To execute the general test procedure, an individual should:

*download the standard test program into the EPROM of the microcontroller (this could be achieved either by use of a computer connected to an EPROM programmer or by copying the contents of an existing EPROM into the microcontroller)
•use the programmer diagnostics to verify the EPROM contents, thus confirming the nonvolatile memory to be intact
•put the chip into the functional tester and power up
*observe the state of the LED indicators to determine whether the ports are functioning correctly

THE HARDWARE

Figure 1 shows the circuit of the tester as it finally evolved. Pairs of ports are connected for the I/O test while other pins also provide the driver circuitry for the LEDs.

The three indicators are labeled as "Fail 01," "Fail 23," and "Chip OK," to signal the possible outcomes of the test.

The tester includes a power supply to provide the 5 V for the operation of the 875 1.

THE TESTER SOFTWARE

The basic algorithm for the test procedure is shown in the form of a flowchart in Figure 2.

The test program was written in ASM5 1 assembly language and is given in Listing 1.

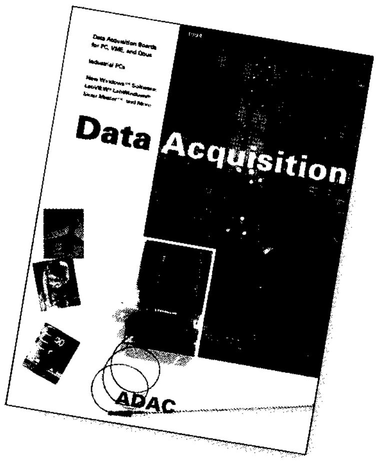
OUR EXPERIENCE WITH THE SYSTEM

We have used the tester over a period of a year to enable students and research workers to verify their microcontrollers. In many cases in which the students were vociferously blaming the chip for having failed, they subsequently found that the problems lay in their circuitry or software. Thus, the major benefit of the tester lay in confirming that the microcontroller was intact so that troubleshooting could proceed in other areas.

The stand-alone nature of the tester has given it a high degree of portability, which would not be the case if we have made a computer-based system. As a result, we have used a single machine for people working on

NEW Data Acquisition Catalog

Covers expanded low cost line.



FREE!

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

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

NEW Low Cost I/O Boards

NEW Industrial PCs

NEW Isolated Analog and Digital Industrial **I/O**

New from the inventors of plug-in data acquisition.

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

ADAC

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

#121

Listing 1—*This program is used to test 8751 microcontrollers and can only be used on the appropriate tester.*

```
; Main program

          org     0000h
ptest:    mov     ie,#00h         ;disable all interrupts
ports:    mov     0070h,#80h      ;initial test pin value
          mov     a,0070h
tpin:     rl      a               ;rotate one place left
          mov     0070h,a
          acall   test23
          acall   test01
          mov     a,0070h
          cjne    a,#80h,tpin     ;test for last port pin
finish:   mov     p2,#80h         ;send Chip OK signal
          acall   delay
          mov     p2,#00h
          acall   delay
          sjmp    finish          ;keep on sending

;the subroutine to test port 2 and 3 pins.

test23:   mov     p2,0070h
          mov     p3,#0ffh
          mov     a,p3
          cjne    a,0070h,fail23
          mov     p2,#00h
          mov     p3,#0ffh
          mov     a,p3
          cjne    a,#00h,fail23
          sjmp    ret23
fail23:   mov     p2,#01h         ;send chip fail signal
          mov     p1,#00h
          acall   delay
          mov     p2,#00h
          acall   delay
          sjmp    fail23          ;send continuously
ret23:    ret

;the subroutine to test port 0 and 1 pins.

test01:   mov     p1,0070h
          mov     p0,#0ffh
          mov     a,p0
          cjne    a,0070h,fail01
          mov     p1,#00h
          mov     p0,#0ffh
          mov     a,p0
          cjne    a,#00h,fail01
          sjmp    ret01
fail01    mov     p1,#80h         ;send chip fail signal
          acall   delay
          mov     p1,#00h
          acall   delay
          sjmp    fail01          ;send continuously
ret01:    ret

delay:    mov     dptr,#0000h     ;LED flashing delay
del1:     mov     a,#0ffh
del2      dec     a
          cjne    a,#00h,del2
          inc     dptr
          mov     r0,dpl
          cjne    r0,#0d1h,del1
          mov     r0,dph
          cjne    r0,#02h,del1
          ret

          end
```

a variety of projects. The tester can be operated by people with limited experience, mainly because of the simplicity of the testing procedure and the binary nature of the output indicators.

Although the machine that we have developed is intended for the **8751** chips only, the addition of a separate EPROM containing the test program would enable it to test the 803 1 and 805 1 microcontrollers since they are not user-programmable.

With the radical changes taking place in Southern Africa, we predict that the market for electronic products with a local flavor is going to increase. Thus, devices of the type described here will be needed to support electronic development initiatives and to stimulate an industry which can be independent and reliable. ❑

*Dr. Mike Collier and Mr. Fred Gweme are both members **of staff** in the Electrical Engineering Department at the University of Zimbabwe. Mike teaches software engineering, microprocessor applications, and computer engineering, and previously lectured in Hong Kong and the U.K. Fred is a Research Fellow in microcontroller systems design. Both authors are committed to the development of engineering within Zimbabwe. They may be reached at collier@zimbix.uz.zw.*

REFERENCES

Jump S., *Microcontroller Applications and Development,* Proc. 4th Intl. Conf. on I.T., Gaberone, Botswana, May 1993.

Collier M. & Gweme F., *Developing an Intermediate Computer,* Proc. 4th Intl. Conf. on I.T., Gaberone, Botswana, May 1993.

Collier M., *A Low-Cost Software-Intensive Local Area Network,* Proc. Africon '92 Conference, Swaziland, September 1992.

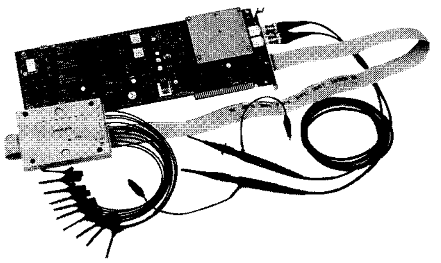
8-bit Embedded Controllers, Intel, 1990, pp. 8-1 to 8-45.

I R S

410 Very Useful
411 Moderately Useful
412 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



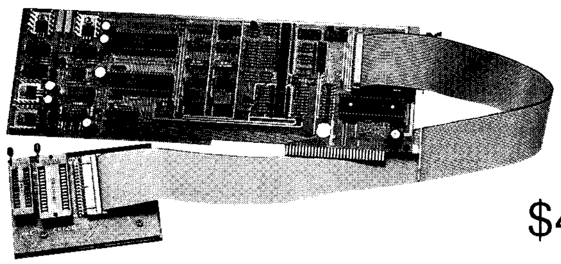
$1799 - DSO-28204 (4K)
$2285 - DSO-28264 (64K)

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
SAL
EPROM
EEPROM
FLASH
MICRO
PIC
etc..

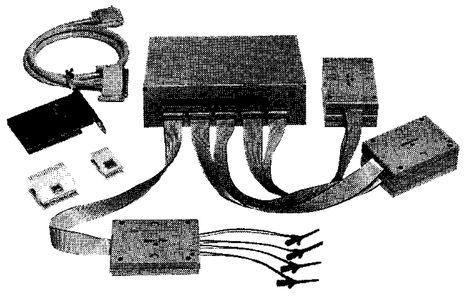


$475

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

Call (201) 808-8990

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

#122

FEATURE ARTICLE

Stephen Bigelow

Understanding PC Buses

While most people are familiar with buses like ISA, EISA, and MCA, what about newer players like VL and PCI? Why do we need more and more buses when the old ones just won't die? Stephen surveys the field to find out.

IBM's first PC marked a radical departure for personal computer designs. Previous computers incorporated key systems (video, storage, communications, and so on) permanently into the motherboard, but IBM chose to build a motherboard containing only core circuitry (the CPU, math coprocessor, RAM, ROM, and glue logic). Functions such as video adapters, serial and parallel ports, modems, floppy and hard drive storage controllers were treated as "add-on" devices that could be purchased and upgraded later. Such add-on devices were integrated with the PC through an expansion bus—a series of connectors that give add-on cards access to data, memory, interrupts, power, timing signals, and the like.

Just *why* the early IBM engineers made this fateful design choice is a question that will probably never be answered, yet the impact of that decision sparked a personal computer revolution that continues today. In the intervening 15 years, competition between peripherals manufacturers has given consumers high-performance, photorealistic color video; storage systems exceeding 2 GB; modems that can transfer data well above 14.4 kbps; and a host of other peripherals which take advantage of a PC's bus.

With continuous improvements in CPUs and other core logic, the PC expansion bus has become a key factor in the overall performance of a system, New bus architectures are now appearing that promise to keep pace with future generations of PCs. This article is intended present the architecture and layout of today's most popular PC expansion buses.

ISA

The venerable Industry Standard Architecture (ISA) shown in Figure 1 was the first open system bus architecture used for personal computers. Since there were no restrictions placed on the use of ISA buses (also referred

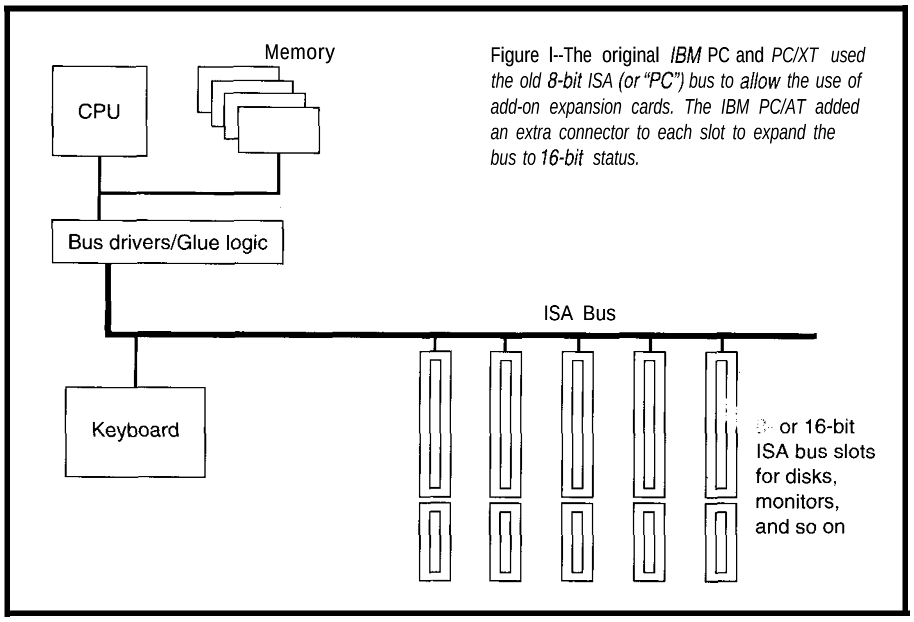


Figure I--The original *IBM* PC and *PC/XT used the old 8-bit ISA (or "PC") bus to allow the use of add-on expansion cards. The IBM PC/AT added an extra connector to each slot to expand the bus to 16-bit status.*

to as "PC buses"), they were duplicated in every IBM-compatible clone that followed. Not only did the use of a standard bus pave the way for thousands of manufacturers to produce compatible PCs and expansion devices, but it also encouraged the use of standardized operating systems and applications.

8-BIT ISA

Use of the S-bit XT bus started in 1982. The S-bit ISA bus consisted of a card-edge connector with 62 contacts. The bus provided eight data lines and twenty address lines which enabled the board to reside within the XT's 1 MB of conventional memory. The bus also supported connections for six interrupts (IRQ2–IRQ7) and three DMA channels and ran at a system speed of 4.77 MHz. Although the bus itself was relatively simple, IBM failed to publish specific timing relationships for data, address, and control signals. This ambiguity left early manufacturers groping to find proper timing relationships by trial and error.

Although each connector on the bus was supposed to work the same, early PCs designed with eight expansion slots required any card inserted in the eighth slot (the slot closest to the power supply) to provide a special "card-selected" signal on pin B8. Timing requirements for the eighth slot were also tighter. Contrary to popular belief, the eighth slot had nothing to do with IBM's expansion chassis. The demands of slot 8 were for support of a keyboard / timer adapter board for IBM's special configuration called the 3270PC. Most XT clones did not adhere to this "eighth slot" peculiarity.

Table 1 shows the pinout for an XT-bus configuration. The oscillator pin provides the 14.3-MHz system oscillator signal to the expansion bus, while the clock pin supplies the 4.77-MHz system clock signal. When the PC needs to be reset, the RESET DRV pin sets the whole system into a reset state. The twenty address pins (O-19) connect an expansion board to the system's address bus. The eight data lines (O-7) connect the board to the system's data bus. When address signals are valid, the address-latch-enable signal indicates that the address may now be decoded.

Signal	Pin	Pin	Signal
Ground	BI	AI	–I/O CHCK
Reset	B2	A2	Data Bit 7
+5 VDC	B3	A3	Data Bit 6
IRQ 2	B4	A4	Data Bit 5
-5 VDC	B5	A5	Data Bit 4
DRQ 2	B6	A6	Data Bit 3
-12 VDC	B7	A7	Data Bit 2
-Card Selected	B8	A8	Data Bit 1
+12 VDC	B9	A9	Data Bit 0
Ground	B10	AI0	–I/O CHRDY
-SMEMW	B11	AII	AEN
–SMEMR	B12	AI2	Address Bit 19
-I/O w	B13	AI3	Address Bit 18
–I/O R	B14	AI4	Address Bit 17
–DACK 3	B15	AI5	Address Bit 16
DRQ 3	B16	AI6	Address Bit 15
–DACK 1	B17	AI7	Address Bit 14
DRQ 1	B18	AI6	Address Bit 13
-REFRESH	B19	AI9	Address Bit 12
Clock (4.77 MHz)	B20	A20	Address Bit 11
IRQ 7	B21	A21	Address Bit 10
IRQ 6	B22	A22	Address Bit 9
IRQ 5	B23	A23	Address Bit 8
IRQ 4	B24	A24	Address Bit 7
IRQ 3	B25	A25	Address Bit 6
–DACK 2	B26	A26	Address Bit 5
T/C	B27	A27	Address Bit 4
BALE	B28	A26	Address Bit 3
+5 VDC	B29	A29	Address Bit 2
Osc (14.3 MHz)	B30	A30	Address Bit 1
Ground	B31	A31	Address Bit 0

Table **1**—*The 8-bit ISA bus was used on the original IBM PC and PC/XT.*

The I/O Channel Check (-I/OCHCK) flags the motherboard when errors occur on the expansion board (the minus sign [–] indicates an active-low signal). The I/O Channel Ready (-I/O CHRDY) is active when an addressed expansion board is ready. If this pin is logic 0, the CPU will extend the bus cycle by inserting wait states. The six hardware interrupts (IRQ2–IRQ7) are used by the expansion board to demand the CPU's attention. Interrupts 0 and 1 are not available to the bus since they handle the highest priorities of the timer chip and keyboard. The I/O Read (-I/O R) and I/O Write (-I/O W) indicate that the CPU or DMA controller want to transfer data to or from the data bus. The Memory Read (-MEMR) and Memory Write (-MEMW) signals tell the expansion board that the CPU or DMA controller is going to read or write data to main memory.

The XT bus supplies three DMA Requests (DRQ1–DRQ3) thereby enabling an expansion board to transfer data to or from memory. If the Address Enable (AEN) signal is true, the DMA controller is controlling the bus for a data transfer. Finally, the Terminal Count (T/C) signal provides a pulse when DMA transfer is complete.

Signal	n	Pin	Signal
Ground	B1	AI	–I/O CHCK
Reset	B2	A2	Data Bit 7
(identical to 8-bit bus in Table 1)			
Osc (14.3 MHz)	B30	A30	Address Bit 1
Ground	B31	A31	Address Bit 0
Key	Key	Key	Key
-MEM CS16	D1	CI	–SBHE
–I/O CS16	D2	c 2	Address Bit 23
IRQ 10	D3	c 3	Address Bit 22
IRQ 11	D4	c 4	Address Bit 21
IRQ 12	D5	c 5	Address Bit 20
IRQ 15	D6	C6	Address Bit 19
IRQ 14	D7	c 7	Address Bit 18
–DACK 0	D8	C8	Address Bit 17
DRQ 0	D9	C9	-MEM R
–DACK 5	D10	CI0	-MEM W
DRQ 5	DII	CI1	Data Bit 8
–DACK 6	D12	CI2	Data Bit 9
DRQ 6	D13	CI3	Data Bit 10
–DACK 7	D14	CI4	Data Bit 11
DRQ 7	D15	C15	Data Bit 12
+5 VDC	D16	CI6	Data Bit 13
-MASTER	D17	C17	Data Bit 14
Ground	D18	C18	Data Bit 15

Table **2**—*The 16-bit ISA bus is identical to the B-bit ISA bus, but adds a second connector with more signals.*

16-BIT ISA

The limitations of the 8-bit ISA bus were soon obvious. With the floppy and hard drives taking up two of the six available interrupts, COM3 and COM4 taking another two (IRQ 4 and IRQ 3), and an LPT port taking IRQ 7, competition for the remaining interrupt was fierce. Of the three DMA channels available, the floppy and hard drives take two-only one DMA channel remains available. Since only 1 MB of address space is addressable, 8 data bits form a serious bottleneck for data transfers. Although it would have been a simple matter to start from scratch and design an entirely new bus, that would have made the entire installed base of XT owners obsolete.

The next logical step in bus evolution came in 1984 and '85 with the introduction of the 80286 in IBM's PC/AT. System resources were added to the bus which still allowed XT boards to function in the expanded bus. The result became what we know today as the 16-bit AT bus. Instead of using a different bus connector, the original 62-pin connector was left intact, and an extra 36-pin connector was added (see Table 2) and designated C and D. As well, an extra eight data bits were added to bring the total data bus to 16 bits, and five interrupts and four DMA channels were included.

Signal	Pin	Pin	Signal
ESYNC	BV10	AV10	VSYNC
Ground	BV9	AV9	HSYNC
P5	BV8	AV8	BLANK
P4	BV7	AV7	Ground
P3	BV6	AV6	P6
Ground	BV5	AV5	EDCLK
P2	BV4	AV4	DCLK
PI	BV3	AV3	Ground
PO	BV2	AV2	P7
Ground	BV1	AV1	EVIDEO
Key	Key	Key	Key
AUDIO Ground	BI	AI	-CD Setup
AUDIO	B2	A2	MADE24
Ground	B3	A3	Ground
Osc (14.3 MHz)	B4	A4	Address Bit 11
Ground	B5	A5	Address Bit 10
Address Bit 23	B6	A6	Address Bit 9
Address Bit 22	B7	A7	+5 VDC
Address Bit 21	B8	A8	Address Bit 8
Ground	B9	A9	Address Bit 7
Address Bit 20	B10	Al0	Address Bit 6
Address Bit 19	Bll	All	+5 VDC
Address Bit 18	B12	Al2	Address Bit 5
Ground	B13	Al3	Address Bit 4
Address Bit 17	B14	Al4	Address Bit 3
Address Bit 16	B15	A15	+5 VDC
Address Bit 15	B16	Al6	Address Bit 2
Ground	B17	Al7	Address Bit 1
Address Bit 14	B18	Al8	Address Bit 0
Address Bit 13	B19	A19	+12 VDC
Address Bit 12	B20	A20	-ADL
Ground	B21	A21	-PREEMPT
–IRQ 9	B22	A22	-BURST
–IRQ 3	B23	A23	-12VDC
–IRQ 4	B24	A24	ARBOO
Ground	B25	A25	ARB 01
–IRQ 5	B26	A26	ARB 02
–IRQ 6	B27	A27	-12VDC
–IRQ 7	B28	A28	ARB03
Ground	B29	A29	ARB/ -GNT
Reserved	B30	A30	-TC
Reserved	B31	A31	+5 VDC
–CHCK	B32	A32	-SO
Ground	B33	A33	-SI
-CMD	B34	A34	M/-I/O
CHRDYRTN	B35	A35	+12 VDC
-CD SFDBK	B36	A36	CD CHRDY
Ground	B37	A37	Data Bit 0
Data Bit 1	B38	A38	Data Bit 2
Data Bit 3	B39	A39	+5 VDC
Data Bit 4	B40	A40	Data Bit 5
Ground	B41	A41	Data Bit 6
CHRESET	B42	A42	Data Bit 7
Reserved	B43	A43	Ground
Reserved	B44	A44	-DS 16 RTN
Ground	B45	A45	-REFRESH
Key	Key	Key	Key
Key	Key	Key	Key
Data Bit 8	B48	A48	+5 VDC
Data Bit 9	B49	A49	Data Bit 10
Ground	B50	A50	Data Bit 11
Data Bit 12	B51	A51	Data Bit 13
Data Bit 14	B52	A52	+12 VDC
Data Bit 15	B53	A53	Reserved
Ground	B54	A54	–SBHE
–IRQ 10	B55	A55	-CD DS 16
–IRQ 11	B56	A56	+5 VDC
–IRQ 12	B57	A57	–IRQ 14
Ground	B58	A58	–IRQ 15
Reserved	B59	A59	Reserved
Reserved	B60	A60	Reserved

Table 3—*The16-bit Micro Channel* Architecture *(MCA) bus* includes *audio and video signals in addition* to *typical* bus *signals.*

Four more address lines were also provided in addition to several more control signals. Clock speed was increased on the AT bus to 8.33 MHz.

The System Bus High Enable (–SBHE) is active when the upper eight data bits are being used. If the upper eight bits are not being used (i.e., an XT board in the AT slot), –SBHE will be inactive. If the expansion board requires 16-bit access to memory locations, it must return an active -MEM CS16 signal. If the expansion board requires 16-bit access to an I/O location, it must make the -I/O CS16 signal active. The –MEMR and –MEMW signals provided by an expansion board tell the CPU or DMA controller that memory access is needed up to 16 MB. The –SMEMR and -SMEMW signals only indicate memory access for the first 1 MB. The -MASTER signal can be used by expansion boards that are able to take control of the bus through use of a DMA channel. It is interesting to note that small, highly integrated AT systems are available for embedded systems and dedicated applications.

MCA

With the introduction and widespread use of 32-bit microprocessors such as the 80386 and 80486, the 16-bit ISA bus again faced a data throughput bottleneck. Passing a 32-bit word across the expansion bus in two 16-bit halves presented a serious waste of valuable processing time. Not only were data and CPU speed an issue, but video and audio systems in PCs had also improved. By early 1987, IBM concluded that it was time to lay the ISA bus to rest, and to unleash an entirely new bus structure which it dubbed the Micro Channel Architecture (MCA). IBM incorporated the MCA bus into their PS/2 series of personal computers and their System/6000 workstations.

MCA offers plug-in audio and video capability, bus arbitration for sophisticated peripherals, 10-MHz operation for high data throughput, and automatic setup configuration. Although MCA offers many enhancements over the ISA bus, computer users are refusing to abandon their hardware and software investment to scramble for limited MCA-compatible peripherals to fill their needs. As a result, the MCA bus has not become the standard IBM hoped it would be.

There are two major types of MCA slots: 16-bit with video extensions and 32-bit. The 16-bit MCA slot is shown in Table 3. It is a primary type of MCA connector which combines video and audio signals in the expansion bus. The connection itself can be divided into three sections: the video section (xV10–xV1), the 8-bit section (1–45), and the 16-bit section (48-58). Power, ground, and interrupt lines are easy to spot, but most other signals are new. The 32-bit MCA slot is shown in Table 4. The 32-bit bus replaces the video section with a smaller matched memory control section (xM4–xM1), but 8- and 16-bit sections remain the same. The 32-bit MCA slot also includes a 32-bit section (59-89).

Enable Synchronization (ESYNC) controls VGA signals on the motherboard. When ESYNC is true, the Vertical Synchronization (VSYNC), Horizontal Synchronization (HSYNC), and Blanking (BLANK) signals control the display. An independent 8-bit video data bus (PO-P7) supports 256 colors on the VGA display. VGA timing signals are controlled by the Enable Data Clock (EDCLK) and Data Clock (DCLK) signals. The Enable Video

Signal	Pin	Pin	Signal
Ground	BM4	AM4	Reserved
Reserved	BM3	AM3	-MMC CMD
-MMCR	BM2	AM2	Ground
Reserved	BM1	AM1	-MMC
AUDIO Ground	B1	AI	-CD Setup
AUDIO	B2	A2	MADE24
Ground	B3	A3	Ground
(Identical to B4/A4–B57/A57 in Table 3)			
Ground	B58	A58	–IRQ 15
Reserved	B59	A59	Reserved
Reserved	B60	A60	Resewed
Reserved	B61	A61	Ground
Reserved	B62	A62	Reserved
Ground	B63	A63	Reserved
Data Bit 16	B64	A64	Reserved
Data Bit 17	B65	A65	+12 VDC
Data Bit 18	B66	A66	Data Bit 19
Ground	B67	A67	Data Bit 20
Data Bit 22	B68	A68	Data Bit 21
Data Bit 23	B69	A69	+5 VDC
Reserved	B70	A70	Data Bit 24
Ground	B71	A71	Data Bit 25
Data Bit 27	B72	A72	Data Bit 26
Data Bit 28	B73	A73	+5 VDC
Data Bit 29	B74	A74	Data Bit 30
Ground	B75	A75	Data Bit 31
–BE 0	B76	A76	Reserved
-BE 1	B77	A77	+12 VDC
-BE2	B78	A78	-BE3
Ground	B79	A79	-DC 32 RTN
TR32	B80	A80	-CD DS 32
Address Bit 24	B81	A81	+5 VDC
Address Bit 25	B82	A82	Address Bit 26
Ground	B83	A83	Address Bit 27
Address Bit 29	B84	A84	Address Bit 28
Address Bit 30	B85	A85	+5 VDC
Address Bit 31	B86	A86	Reserved
Ground	B87	A87	Reserved
Reserved	B88	A88	Reserved
Reserved	B89	A89	Ground

Table *4-The 32-bit MCA bus is nearly identical to the 16-bit version, but rep/aces the video section with a smaller matched memory control section and includes additional address and data lines.*

BUS SPECIFICATION COMPARISON

Specification	ISA(8)	ISA(16)	EISA(16/32)	MCA(16/32)	VL(32/64)	PCI(32/64)
Data Bus	8	16	16/32	16/32		64 *
Address Bus	20	27	27/32	24/32	32/64	64 *
Bus Pins	62	98	98/100	140/182	116	188 (w/keys)
Clock Speed	4.77 MHz	8.33 MHz	8.33 MHz	10 MHz	**	33 MHz
Interrupts	6	11	11	11/11	1	4
DMA Channels	3	7	7	none***	none***	none***

* address and data lines are multiplexed on the same conductors
** depends on the speed of the host computer
*** bus mastering used

(EVIDEO) signal switches control of the palette bus enabling an external video adapter to provide signals on P0–P7. Audio [AUDIO) and Audio Signal Ground (AUDIO Ground) enable the expansion board to send tone signals to the motherboard speaker.

There are 32 address bits (Address Bit O-Address Bit 3 1), 11 interrupts, and 32 data bits (Data Bit 0–Data Bit 3 1). The Address Latch (–ADL) signal is true when a valid address exists on the address lines. A Channel Check (-CHCK) signal flags the motherboard when an error is detected on the expansion board. When data on the data bus is valid, the Command (-CMD) is true. The Channel Ready Return (CHRDYRTN) signal is sent to the motherboard when the addressed expansion board I/O channel is ready. A Channel Reset (CHRESET) signal can be used to reset all expansion boards. The Card Setup (-CDSETUP) instructs an addressed board to perform a setup. The Memory Address Enable 24 (MADE24) line activates address line 24. After completing its access, the Channel Ready (CHRDY) line indicates that the addressed board is idle. When Burst (-BURST) is true, the system bus executes a burst cycle.

The Data Size 16 Return (-DS 16RTN) and Data Size 32 Return (–DS32RTN) tell the motherboard whether the board is running at a 16- or 32-bit bus width. The System Byte High Enable (SBHE) signal is true when the upper 16 data bits are being used, but the Card Data Size 16 (CDDS16) signal is true when only 16 data bits are being used. If all 32 bits of data are being transferred, the Card Data Size 32 (-CDDS32) signal is true. When main memory is refreshed, the Refresh (-REF) line is true. This allows any dynamic memory on expansion boards to be refreshed as well. The Memory or I/O (M/-I/O) signal defines whether the expansion board is accessing a memory or I/O location. Signals -SO and -S 1 carry the status of an MCA bus.

The Preempt (-PREEMPT) signal is true when a bus arbitration cycle begins. The Arbitration signals (ARB00–ARB03) indicate which of the 16 possible bus masters has won arbitration. The Arbitration or Grant (ARB/-GNT) is high when the bus is in arbitration, and low when bus control has been granted. When a DMA transfer is done, Terminal Count (TC) is true. Byte Enable signals 0 to 3 (–BE0 to –BE3) indicate which four bytes of a 32-bit data bus are transferring data. When an external master is a 32-bit device, the Translate 32 (TR32) line is true. The -MMCR, -MMCCMD, and -MMC lines are matched memory control signals.

EISA

In 1988–'89, the Extended ISA (EISA) bus, a 32-bit bus, was developed to meet the demand for greater speed and performance from expansion peripherals incited by the speed of 80386 and 80486 CPUs. It also did not make sense to leave the entire 32-bit bus market to IBM's MCA bus. Even though the EISA bus works at 8.33 MHz, the 32-bit data path doubles data throughput between the motherboard and expansion boards.

Unlike the MCA bus, EISA ensures backward compatibility with existing ISA peripherals and PC software. The EISA bus is designed to be fully compatible with ISA boards as shown in the pinout of Table 5.

32 bit	16 bit	Pin	Pin	16 bit	3 2
Ground	Ground	B1	AI	-I/O CHCK	-CMD
+5 VDC	Reset	B2	A2	Data 7	-START
+5VDC	+5 VDC	B3	A3	Data 6	EXRDY
Reserved	IRQ 9	B4	A4	Data 5	–EX32
Reserved	-5VDC	B5	A5	Data 4	Ground
Key	DRQ2	B6	A6	Data 3	Key
Reserved	-12VDC	B7	A7	Data 2	–EX16
Reserved	–0 WAIT	B8	A8	Data 1	–SLBURST
+12 VDC	+12 VDC	B9	A9	Data 0	-MSBUFiST
M -I/O	Ground	B10	A10	–I/O CHRDY	W - R
-LOCK	-SMEMW	B11	A I I	AEN	Ground
Reserved	–SMEMR	B12	AI2	Addr. 19	Reserved
Ground	-I/O W	B13	AI3	Addr. 18	Reserved
Reserved	-I/O R	B14	AI4	Addr. 17	Reserved
-BE3	–DACK 3	B15	AI5	Addr. 16	Ground
Key	DRQ3	B16	AI6	Addr. 15	Key
–BE2	–DACK 1	B17	AI7	Addr. 14	–BE1
–BE0	DRQ 1	B18	AI8	Addr. 13	–Addr. 31
Ground	-REFRESH	B19	A19	Addr. 12	Ground
+5 VDC	Clock (8.33 MHz)	B20	A20	Addr. 11	–Addr. 30
–Addr. 29	IRQ 7	B21	A21	Addr. 10	–Addr. 28
Ground	IRQ 6	B22	A22	Addr. 9	–Addr. 27
–Addr. 26	IRQ 5	B23	A23	Addr. 8	–Addr. 25
–Addr. 24	IRQ 4	B24	A24	Addr. 7	Ground
Key	IRQ 3	B25	A25	Addr. 6	Key
Addr. 16	–DACK 2	B26	A26	Addr. 5	Addr. 15
Addr. 14	T/C	B27	A27	Addr. 4	Addr. 13
+5 VDC	BALE	B28	A28	Addr. 3	Addr. 12
+5 VDC	+5 VDC	B29	A29	Addr.2	Addr. 11
Ground	Osc. (14.3 MHz)	B30	A30	Addr. 1	Ground
Addr. 10	Ground	B31	A31	Addr. 0	Addr. 9
Key	Key	Key	Key	Key	Key
Addr. 8	-MEM CS16	D1	C I	–SBHE	Addr. 7
Addr. 6	-I/O CS16	D2	C2	Addr. 23	Ground
Addr. 5	IRQ 10	D3	C3	Addr. 22	Addr. 4
+5 VDC	IRQ 11	D4	C4	Addr. 21	Addr. 3
Addr. 2	IRQ 12	D5	C5	Addr 20	Ground
Key	IRQ 15	D6	C6	Addr 19	Key
Data 16	IRQ 14	D7	C7	Addr. 18	Data 17
Data 16	-DACKO	D8	C8	Addr. 17	Data 19
Ground	DRQO	D9	C9	–MEM R	Data 20
Data 21	–DACK 5	D10	CI0	-MEM W	Data 22
Data 23	DRQ 5	D11	CI1	Data 8	Ground
Data 24	–DACK 6	D12	CI2	Data 9	Data 25
Ground	DRQ 6	D13	CI3	Data10	Data 26
Data 27	–DACK 7	D14	CI4	Data11	Data 28
Key	DRQ 7	D15	CI5	Data12	Key
Data 29	+5 VDC	D16	CI6	Data13	Ground
+5 VDC	-MASTER	D17	CI7	Data14	Data 30
+5 VDC	Ground	D18	CI6	Data15	Data 31
–MAKx	———	D19	C I 9	———	–MREQx

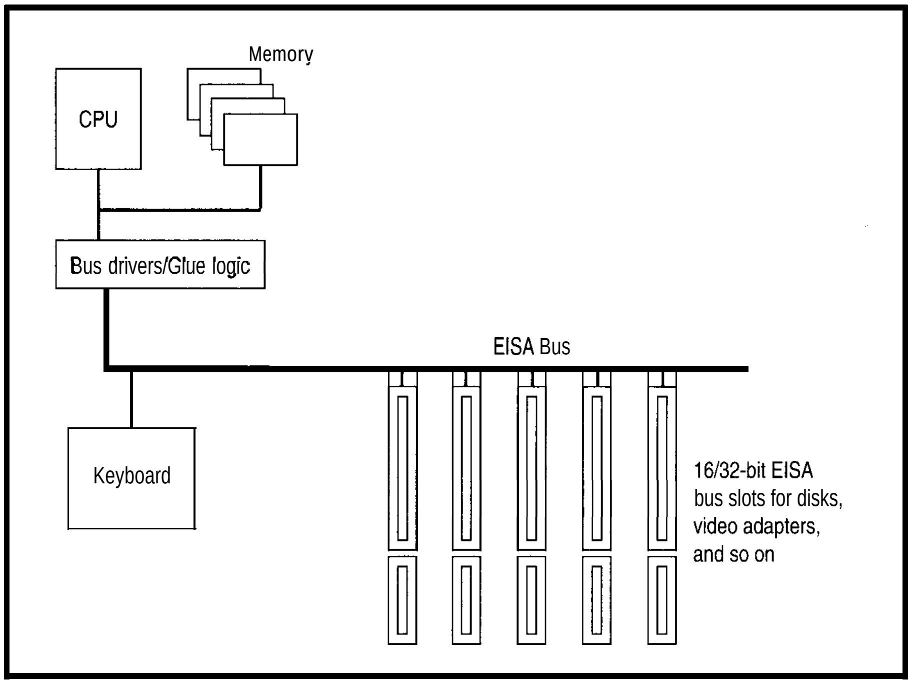
Table 5—*EISA* **(Enhanced** *ISA) includes all the signals from* **the original** *16-***bit** *ISA bus, but extends the bus to* 32 *bits* **using** *a second row of pins.*

EISA switches automatically between 16-bit ISA and 32-bit EISA operation using a second row of edge connectors and the -EX32 and -EX16 lines. Thus, EISA boards have access to all of the signals available to ISA boards and the second row of EISA signals.

As with the MCA bus, EISA supports arbitration for bus mastering and automatic board configuration which simplifies the installation of new boards. The EISA bus can access fifteen interrupt levels and seven DMA channels. To maintain backward compatibility with ISA expansion boards, however, there is no direct bus support for video or audio as there is with the MCA bus. Since the EISA bus clock runs at the same 8.33-MHz rate as ISA, the potential data throughput of an EISA board is roughly twice that of ISA boards. EISA systems are used as network servers, workstations, and high-end PCs. Although EISA systems have proliferated more than MCA systems, EISA remains a high-end standard—never really filtering down to low-cost consumer systems.

The EISA bus uses 30 address lines (Addr. 2-Addr. 31). The lower two address lines (AO, Al) are decoded by the Byte Enable lines (-BE0 to -BE3). Data bits O-15 are taken from the ISA portion of the bus, but the upper 16 data lines are provided by (Data 16–Data 31). Like the MCA, the M/-I/O signal determines whether a memory or I/O bus cycle is executed, while the Write or Read (W/-R) line defines whether the access is for reading or writing. When an EISA device completes a bus cycle, the EISA Ready (EXRDY) line is used to insert wait states. When the motherboard is providing exclusive access to the EISA board, the Locked Cycle (-LOCK) signal is true. If an EISA board can run in 32-bit mode, the EISA 32-bit Device (-EX32) signal is true. But if the board can only run in 16-bit mode, the EISA 16-bit Device (-EX16) signal is true.

The Master Burst (-MSBURST) signal is activated by the EISA bus master which informs the EISA bus controller that a burst transfer cycle will commence, thereby doubling the bus transfer rate. When an external device must send a data burst, it activates the Slave Burst (-SLBURST) line. An external device requests control of the EISA bus using the Master Request (-MREQ) line. If the bus arbitrator decides that the requester can control the bus, a Master Acknowledge (-MACK) signal is sent to the requesting device. A Command (-CMD) signal is sent to synchronize the EISA bus cycle with the system clock, and the Start (-START) signal coordinates the system clock with the beginning of the EISA bus cycle. Finally, the Bus Clock (BCLK) is provided at 8.33 MHz.


Memory
CPU
Bus drivers/Glue logic
EISA Bus
Keyboard
16/32-bit EISA bus slots for disks, video adapters, and so on
Figure **2—***The VL* (Video Local) bus designed by *VESA* is local to *the microprocessor and system memory and is* designed to *allow the processor* high-speed access to graphics *and* video electronics.

VL

The demands of data transfer across the expansion bus have continued to evolve faster than the throughput of classical ISA/EISA bus architectures. The volumes of data required by graphic user interfaces, such as Microsoft Windows, presented serious challenges to video adapter and memory design. Early in 1992, the Video Electronics Standards Association (VESA) proposed a new local bus standard called the Video Local (VL) bus which was intended to improve the performance of graphics and video systems. In general terms, a VL bus is a pathway that gives peripherals access to the system's main memory quickly. For the VL bus (Figure 2), such improved access means higher data throughput and performance for video information at the speed of the CPU itself. By using a stand-alone bus for video, ISA or EISA buses can be implemented for backward system compatibility. Users can upgrade to a new motherboard and graphics card, but all other peripherals and software remain compatible.

The VL bus uses a 116-pin, card-edge connector with small contacts (similar in appearance to Micro Channel contacts) as shown in Table 6. The current VL bus release (V2.0) offers a full 64-bit data path (Data O-Data 63) with a maximum rated data throughput of about 260 MBps. The Data or Command (D/-C) signal tells whether information on the bus is data or a command. Clock signals from the CPU are provided through the Local Bus Clock (LCLK) line. Memory or I/O (M/-I/O) distinguishes between memory and I/O accesses, while the Write or Read (W/-R) signal differentiates between read and write operations. Since the VL bus is 64 bits wide, -BE0 to -BE7 indicate which 8-bit portions of the 64-bit bus are being transferred. A Reset signal (-RESET) initializes the VL device, and the Ready Return (-RDYRTN) line indicates that the VL bus is accessible.

64 bit	32 bit	Pin	Pin	32 bit	64 bit
—	Data00	A01	B01	Data 01	—
—	Data 02	A02	B02	Data03	—
—	Data 04	A03	B03	Ground	—
—	Data06	A04	B04	Data05	—
—	Data08	A05	B05	Data 07	—
—	Ground	A06	B06	Data09	—
—	Data 10	A07	B07	Data 11	—
—	Data 12	A08	B08	Data 13	—
—	+VCC	A09	B09	Data 15	—
—	Data 14	Al0	B10	Ground	—
—	Data 16	All	Bll	Data 17	—
—	Data18	Al2	B12	+VCC	—
—	Data20	Al3	B13	Data19	—
—	Ground	Al4	B14	Data21	—
—	Data 22	Al5	B15	Data 23	—
—	Data24	Al6	B16	Data 25	—
—	Data26	Al7	B17	Ground	—
—	Data 28	Al8	B18	Data27	—
—	Data30	Al9	B19	Data29	—
—	+VCC	A20	B20	Data31	—
Data 63	Addr. 31	A21	B21	Addr. 30	Data62
—	Ground	A22	B22	Addr. 28	Data 60
Data 61	Addr. 29	A23	B23	Addr 26	Data 58
Data 59	Addr. 27	A24	B24	Ground	—
Data 57	Addr. 25	A25	B25	Addr. 24	Data 56
Data 55	Addr. 23	A26	B26	Addr. 22	Data54
Data 53	Addr. 21	A27	B27	+VCC	—
Data 51	Addr. 19	A28	B28	Addr. 20	Data52
—	Ground	A29	B29	Addr. 18	Data 50
Data 49	Addr. 17	A30	B30	Addr. 16	Data48
Data 47	Addr. 15	A31	B31	Addr. 14	Data46
—	+VCC	A32	B32	Addr. 12	Data 44
Data 45	Addr. 13	A33	B33	Addr. 10	Data42
Data 43	Addr. 11	A34	B34	Addr. 8	Data 40
Data 41	Addr. 9	A35	B35	Ground	—
Data 39	Addr. 7	A36	B36	Addr. 6	Data 38
Data 37	Addr. 5	A37	B37	Addr. 4	Data 36
—	Ground	A38	B38	–WBAK	—
Data 35	Addr. 3	A39	B39	–BE 0	-BE 4
Data 34	Addr. 2	A40	B40	+VCC	—
–LBS64	n/c	A41	B41	-BE 1	-BE 5
—	-RESET	A42	B42	-BE2	-BE 6
—	D/ –C	A43	B43	Ground	—
—	M/ –I/O	A44	B44	-BE 3	-BE 7
—	WI-R	A45	B45	-ADS	—
Key	Key	Key	Key	Key	Key
Key	Key	Key	Key	Key	Key
—	–RDYRTN	A48	B48	–LRDY	—
—	Ground	A49	B49	-LDEV	—
—	IRQ 9	A50	B50	–LREQ	—
—	–BRDY	A51	B51	Ground	—
—	-BLAST	A52	B52	–LGNT	—
Data 32	ID 0	A53	B53	+VCC	—
Data 33	ID 1	A54	B54	ID2	—
—	Ground	A55	B55	ID3	—
—	LCLK	A56	B56	!D 4	–ACK64
—	+VCC	A57	B57	n/c	—
—	–LBS16	A58	B58	-LEADS	—

Table **6**—*The VL bus (V2.0) was* designed *specifically for* graphics and video use.

Data bus width is determined by the Local Bus Size 16 (-LBS16) or Local Bus Size 64 (-LBS64) signals. If a 64-bit bus width is used, the Acknowledge 64-bit (-ACK64) signal is true.

Accessing the VL bus is a process of arbitration-much like the arbitration that takes place on an MCA or EISA bus. Each VL device is defined by its own ID number (ID0–ID4). The Local Bus Ready (-LRDY), Local Bus Device (-LDEV), Local Bus Request (-LREQ), and Local Bus Grant (-LGNT) lines negotiate control of the VL bus. In most cases, there is only one VL device on the bus, but arbitration must be performed to ensure proper access to memory.

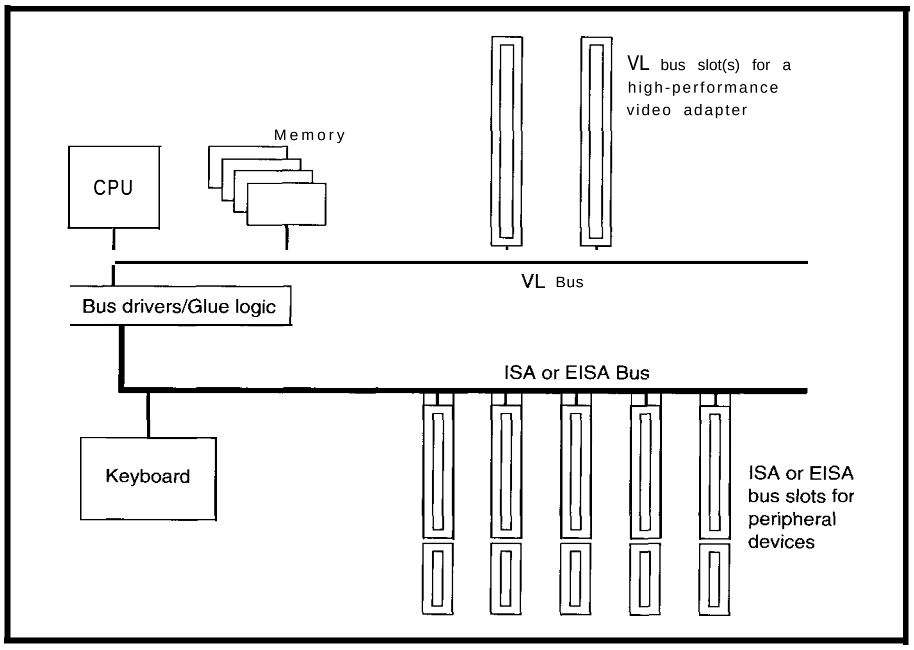

VL bus slot(s) for a high-performance video adapter
Memory
CPU
VL Bus
Bus drivers/Glue logic
ISA or EISA Bus
Keyboard
ISA or EISA bus slots for peripheral devices

Figure **3**—*The PCI (Peripheral Component Interconnect) bus was designed by Intel and a consortium of computer manufacturers as a* complete *system bus of the future.*

PCI

Despite advantages offered by the VL bus, there are some serious limitations that must be overcome. Perhaps most important is the VL's dependence on CPU speed (fast computers must use wait states with the VL bus) and support of only 1 or 2 slots. As well, the VL standard is voluntary; not all manufacturers adhere to VESA specifications completely.

In mid-1992, Intel Corporation and a comprehensive consortium of manufacturers introduced the Peripheral Component Interconnect (PCI) bus. The VL bus was designed specifically to enhance PC video systems, but the 188-pin PCI bus (Figure 3) looks to the future of CPUs [and PCs in general) by providing a bus architecture that also supports peripherals such as hard drives, networks, and so on.

Unlike the VL bus, PCI uses an independent, internal clock frequency of 33 MHz. This decouples the PCI expansion device from CPU speed, making it possible to upgrade future boards regardless of PC performance. With a 64-bit data path, the PCI bus (shown in Table 7) offers about the same data throughput as VL. PCI also supports up to 16 slots (or expansion devices) on the motherboard using bus-mastering techniques, although only PCI video adapters will be available in the immediate future. While systems are now being developed with PCI compatibility, ISA or EISA buses are also included to ensure backward compatibility. Given the tenacious history of ISA buses, it is likely that PCI devices will continue to share real estate with ISA devices into the next decade.

To reduce the number of pins needed in the PCI bus, data and address lines are multiplexed together (Adr./Dat 0-Adr./Dat 63). From Table 7, it is also interesting to note that PCI is the first bus standard designed to support low-voltage (+3.3 VDC) logic implementation. On inspection, you will see that +5 VDC and +3.3 VDC implementations of the PCI bus place their physical key slots in different locations so that the two implementations are not interchangeable. The Clock (CLOCK) signal provides timing for the PCI bus only, and can be adjusted from DC to 33 MHz. Asserting the Reset (-RST) signal resets all PCI devices. Since the 64-bit data path uses eight bytes, the Command or Byte Enable signals (C/-BEO-C/-BE7) define which bytes are transferred. Parity across the Adr/Dat and BE lines is represented with a Parity (PAR) or 64-bit Parity (PAR64) signal. Bus mastering is initiated by the -REQ line and granted after approval using the Grant (–GNT) line.

5 Volt	B.3 Volt	n	Bin3	Volt	5 Volt
-12 VDC	–12 VDC	B I	A I	–TRST	–TRST
TCK	TCK	B2	A 2	+12 VDC	+12 VDC
Ground	Ground	B3	A3	TMS	TMS
TDO	TDO	B4	A4	TDI	TDI
+5 VDC	+5 VDC	B5	A 5	+5 VDC	+5 VDC
+5 VDC	+5 VDC	B6	A 6	–INTA	–INTA
–INTB	–INTB	B 7	A 7	–INTC	–INTC
–INTD	–INTD	B8	A 6	+5 VDC	+5 VDC
-PRSNTI	–PRSNT1	B9	A9	Reserved	Reserved
Reserved	Reserved	B10	AI0	+3.3 VDC (I/O)	+5 VDC
–PRSNT2	–PRSNT2	B I I	AI 1	Reserved	Reserved
Ground	Key	B12	AI2	Key	Ground
Ground	Key	B13	AI3	Key	Ground
Reserved	Reserved	B14	AI 4	Reserved	Reserved
Ground	Ground	B15	AI5	–RST	–RST
Clock	Clock	B16	AI6	+3.3 VDC	+5 VDC
Ground	Ground	B17	AI7	-GNT	-GNT
–REQ	–REQ	B16	AI8	Ground	Ground
+5 VDC	+3.3 VDC	B19	AI9	Reserved	Reserved
Adr/Dat 31	Adr/Dat 31	B20	A20	Adr/Dat 30	Adr/Dat 30
Adr/Dat 29	Adr/Dat 29	B21	A21	+3.3 VDC	+5 VDC
Ground	Ground	B22	A22	Adr/Dat 28	Adr/Dat 28
Adr/Dat 27	Adr/Dat 27	B23	A23	Adr/Dat 26	Adr/Dat 26
Adr/Dat 25	Adr/Dat 25	B24	A24	Ground	Ground
+5 VDC	+3.3 VDC	B25	A25	Adr/Dat 24	Adr/Dat 24
C/-BE3	C/-BE3	B26	A26	IDSEL	IDSEL
AdriDat 23	Adr/Dat 23	B27	A27	+3.3 VDC	+5 VDC
Ground	Ground	B28	A28	Adr/Dat 22	Adr/Dat 22
Adr/Dat 21	Adr/Dat 21	B29	A29	Adr/Dat 20	Adr/Dat 20
Adr/Dat 19	Adr/Dat 19	B30	A30	Ground	Ground
+5 VDC	+3.3 VDC	B31	A31	Adr/Dat 18	Adr/Dat 18
Adr/Dat 17	Adr/Dat 17	B32	A32	Adr/Dat 16	Adr/Dat 16
C/-BE2	C/-BE2	B33	A33	+3.3 VDC	+5 VDC
Ground	Ground	B34	A34	-FRAME	-FRAME
–IRDY	–IRDY	B35	A35	Ground	Ground
+5 VDC	+3.3 VDC	B36	A 3 6	–TRDY	–TRDY
-DEVSEL	-DEVSEL	B37	A37	Ground	Ground
Ground	Ground	B38	A38	-STOP	-STOP
-LOCK	-LOCK	B39	A39	+3.3 VDC	+5 VDC
–PERR	–PERR	B40	A40	SDONE	SDONE
+5 VDC	+3.3 VDC	B41	A41	-SBO	-SBO
–SERR	–SERR	B42	A42	Ground	Ground
+5 VDC	+3.3 VDC	B43	A43	PAR	PAR
C/–BE1	C/–BE1	B44	A44	Adr/Dat 15	Adr/Dat 15
Adr/Dat 14	Adr/Dat 14	B45	A45	+3.3 VDC	+5 VDC
Ground	Ground	B46	A46	Adr/Dat 13	Adr/Dat 13
Adr/Dat 12	Adr/Dat 12	B47	A47	Adr/Dat 11	Adr/Dat 11
Adr/Dat 10	Adr/Dat 10	B48	A46	Ground	Ground
Ground	Ground	B49	A49	Adr/Dat 9	Adr/Dat 9
Key	Ground	B50	A50	Ground	Key
Key	Ground	B51	A51	Ground	Key
Adr/Dat 8	Adr/Dat 8	B52	A52	C/–BE0	C/–BE0
Adr/Dat 7	Adr/Dat 7	B53	A53	+3.3 VDC	+5 VDC
+5 VDC	+3.3 VDC	B54	A 5 4	Adr/Dat 6	Adr/Dat 6
AdriDat 5	Adr/Dat 5	B55	A55	Adr/Dat 4	Adr/Dat 4
Adr/Dat 3	Adr/Dat 3	B56	A56	Ground	Ground
Ground	Ground	B57	A57	Adr/Dat 2	Adr/Dat 2
Adr/Dat 1	Adr/Dat 1	B58	A58	AdriDat 0	Adr/Dat 0
+5 VDC	+3 3 VDC	B59	A59	+3.3 VDC	+5 VDC
–ACK64	–ACK64	B60	A60	–REQ64	–REQ64
+5 VDC	+5 VDC	B61	A61	+5 VDC	+5 VDC
+5 VDC	+5 VDC	B62	A62	+5 VDC	+5 VDC
Key	Key	Key	Key	Key	Key
Key	Key	Key	Key	Key	Key
Reserved	Reserved	B63	A63	Ground	Ground
Ground	Ground	B64	A64	C/-BE7	C/–BE7
C/-BE6	C/-BE6	B65	A65	C/-BE5	C/-BE5
C/-BE4	C/-BE4	B66	A66	+3.3 VDC	+5 VDC
Ground	Ground	B67	A67	PAR64	PAR64
Adr/Dat 63	Adr/Dat 63	B68	A 6 8	Adr/Dat 62	Adr/Dat 62
Adr/Dat 61	Adr/Dat 61	B69	A69	Ground	Ground
+5 VDC	+3.3 VDC	B70	A70	Adr/Dat 60	Adr/Dat 60
AdriDat 59	Adr/Dat 59	B71	A71	Adr/Dat 58	AdriDat 56
AdriDat 57	Adr/Dat 57	B72	A72	Ground	Ground
Ground	Ground	B73	A73	Adr/Dat 56	Adr/Dat 56
Adr/Dat 55	Adr/Dat 55	B74	A74	Adr/Dat 54	Adr/Dat 54
AdriDat 53	Adr/Dat 53	B75	A75	+3.3 VDC	+5 VDC
Ground	Ground	B76	A76	Adr/Dat 52	Adr/Dat 52
Adr/Dat 51	Adr/Dat 51	B77	A77	Adr/Dat 50	Adr/Dat 50
Adr/Dat 49	Adr/Dat 49	B78	A76	Ground	Ground
+5 VDC	+3.3 VDC	B79	A79	Adr/Dat 48	Adr/Dat 48
Adr/Dat 47	Adr/Dat 47	B80	A80	Adr/Dat 46	Adr/Dat 46
Adr/Dat 45	Adr/Dat 45	B81	A81	Ground	Ground
Ground	Ground	B82	A82	Adr/Dat 44	Adr/Dat 44
Adr/Dat 43	Adr/Dat 43	B83	A83	Adr/Dat 42	Adr/Dat 42
Adr/Dat 41	Adr/Dat 41	B84	A84	+3.3 VDC	+5 VDC
Ground	Ground	B85	A85	Adr/Dat 40	Adr/Dat 40
Adr/Dat 39	Adr/Dat 39	B86	A86	Adr/Dat 38	Adr/Dat 38
Adr/Dat 37	Adr/Dat 37	B87	A87	Ground	Ground
+5 VDC	+3.3 VDC	B88	A88	Adr/Dat 36	Adr/Dat 36
AdriDat 35	Adr/Dat 35	B89	A89	Adr/Dat 34	Adr/Dat 34
AdriDat 33	Adr/Dat 33	B90	A90	Ground	Ground
Ground	Ground	B91	A91	Adr/Dat 32	Adr/Dat 32
Reserved	Reserved	B92	A92	Reserved	Reserved
Reserved	Reserved	B93	A93	Ground	Ground
Ground	Ground	B94	A94	Reserved	Reserved

Table 7-The *188-pin PC/ bus* provides ***a*** *bus* architecture *that supports* hard drives, *networks,* and other *typical* peripherals. *It* ***also*** *handles both 5-V and 3.3-V system* ***power.***

When a valid PCI bus cycle is in progress, the Frame (-FRAME] signal is true. If the PCI bus cycle is in its final phase, Frame will be released. The Target Ready (-TRDY) line is true when an addressed device is able to complete the data phase of its bus cycle. An Initiator Ready (-IRDY) signal indicates valid data is present on the bus or the bus is ready to accept data. -FRAME, –TRDY, and –IRDY are used together. A Stop (-STOP) signal is asserted by a target asking a master to halt the current data transfer. ID Select (IDSEL) is used as a chip select signal during board configuration. Device Select (-DEVSEL) is both an input and an output. As an input, it indicates if a device has assumed control of the current bus transfer. As an output, it shows that a device has identified itself as the target for current bus transfer.

There are four interrupt lines (-INTA to –INTD}. When the full 64-bit data mode is used, an expansion device initiates a 64-bit Bus Request (-REQ 64) and awaits a 64-bit Bus Acknowledge (-ACK64) signal from the bus controller. The Bus Lock (-LOCK) signal is an interface control used to ensure use of the bus by a selected expansion device. Error reporting is performed by Primary Error (-PERR) and Secondary Error (-SERR) lines. Cache memory and JTAG support are also provided on the PCI bus.

CONCLUSION

Through standardized bus interfaces, computers can work with peripheral products made by any number of manufacturers. Such open architectures have contributed to the inexpensive, readily available computers that we have today. In the last decade, bus architectures have come a long way. As they continue to change to keep pace with the advances in computer technology, an understanding of past and present bus structures helps you make the most of your current systems as well as preparing you for future changes.

Stephen Bigelow is an electrical engineer working as a technical editor and writer at Dynamic Learning Systems. He may be reached at (508) 366-9487 or at 73652.320% *compuserve.com.*

CONTACT

BCPR Services (EISA Specifications)
1400 L Street, NW
Washington, DC 20005
(202) 371-5921

Electronic Industries Association
(EIA)
Engineering Department
2001 Eye Street NW
Washington, DC 20006
(202) 457-4500

IEEE Computer Society Press
Worldway Postal Center
Los Angeles, CA 90080
(714) 8218380

PCI Special Interest Group
(PCI SIG)
P.O. Box 14070
Portland, OR 97214
(503) 696-2000
Fax: (503) 693-0920

Video Electronic Standards
Association (VESA)
2150 North First St., Suite 440
San Jose, CA 95 13 l-2029
(408) 435-0333
Fax: (408) 435-8225

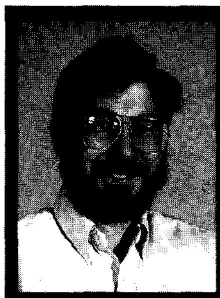
I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

FIRMWARE FURNACE

Ed Nisley

Journey to the Protected Land: Smashing Bugs in Gates



Now that you have a vast, wide-open space in which to put your code, how do you deal with calls and jumps? Also, how does the processor protect itself against rogue code? Ed checks out the scenery.

In real-mode PC programming, it's taken for granted that wild pointers or branches will corrupt data, code, stack, or heap space somewhere in that first megabyte of RAM. Protected mode doesn't solve those problems, but it's a lot easier to find the problems when the CPU says "Ahem, this pointer looks bogus" or "Umm, you don't ***really*** want to execute this data, do you!"

Now that our code is running in 32-bit protected mode, it's time to check out the scenery. This month, I'll cover branches and calls, describe how call and interrupt gates are applied in the IDT, then check out some interrupt-handler response times.

We are now venturing into the ***complex*** part of this Complex-Instruction-Set CPU. In real mode, the CPU does fairly simple things that don't take too long. In protected mode, there's a lot going on under the covers. Stay alert!

'JMPING' AND CALLING

Real-mode 386 programs branch hither and yon using JMP, CALL, and INT instructions with hardware interrupts sharing much of the INT machinery. All this remains true in protected mode, but, as you'll see, we have several other choices that not only simplify branching, but help keep bugs under control. First, the basics..

The distinction between N EAR and FAR jump instructions is familiar to anyone with 80x86 programming experience. Simply put, N EAR J M Ps don't change the CS register. FAR J M Ps transfer control to a different code segment, loading CS with the new segment address in the process. Both types load the IP (Instruction Pointer) register with the offset of the target instruction.

CA L L instructions save the address of the next sequential instruction on the stack before branching to the target instruction. A N EAR CA L L pushes only the offset of the next instruction, while a FAR CALL first pushes the CS register.

The whole purpose of a CA L L is to invoke a routine that will eventually return to the instruction after the CALL. RET instructions come in NEAR and FAR flavors that must match the CALL, although there is no way to tell how you got to a particular routine. Thus, it's essentially impossible to write a routine that's N EAR to some callers and FAR to others.

Pairing a FAR CALL with a NEAR RET leaves the caller's CS on the stack. Conversely, mismatching a NEAR CALL with a FAR RET restores something to CS. In either case, the CPU returns to the wrong address. Finding this problem is a favorite pastime of real-mode programmers.

The 8086 required FAR branches because real-mode code segments were limited to 64 KB. That restriction simply Goes Away in 32-bit PM; you can define a 4-GB code segment if you like. Most application programs won't need FAR branches to reach their own routines.

Mismatched CAL L/RET pairs generate a protection exception when CS is reloaded with junk from the stack or when the CPU executes an invalid instruction after returning to a bad address. The error handler can show you the exact location of the error, which is certainly better than having your system quietly lock up or hose the screen with pinball panic.

Another irritant vanishes in 32-bit PM; conditional J M P instructions can now reach a target anywhere within ±2 GB. The 8086 limitation of t128 bytes spawned an arms race resulting in *n*-pass optimizing assemblers (!) that shuffle your code around to find the smallest, fastest combination of J M Ps to reach the target instruction. No more of that!

Admittedly, the new conditional JMPs are six-byte behemoths instead of the svelte two-byte instructions you're used to. Those optimizers will figure out what you really wanted to code if you let them, although, as you'll see, there are some situations where such optimization is precisely what we *don't* want.

Even if all your application's code is in one segment, you still need a way to communicate with the operating system. Direct branch-into-the-OS interfaces are ancient history, so it should be no surprise that the '386 has a cleaner method. In grand CISC style, there are actually several ways to accomplish the goal.

Quiz: Which major IBM main-frame program branched directly into the operating system to improve performance?

Extra credit: What effect did that have on future operating system enhancements?

GATES IN THE WALL

Once the restrictions of real-mode addressing go away, you can do lots of interesting things. Current 32-bit flat-model operating systems (OS/2, Windows NT, and suchlike) use the vast expanse of a 4-GB address space to good advantage. The entire application fits neatly into the first 2 GB, while the operating system is tucked into the upper 2 GB.

Does anyone yet remember when 64 KB was enough for the operating system and the application? It wasn't ***that*** long ago when 1 MB (well, 640 KB) was enough. Surely 2 GB ought to last for a while.. .

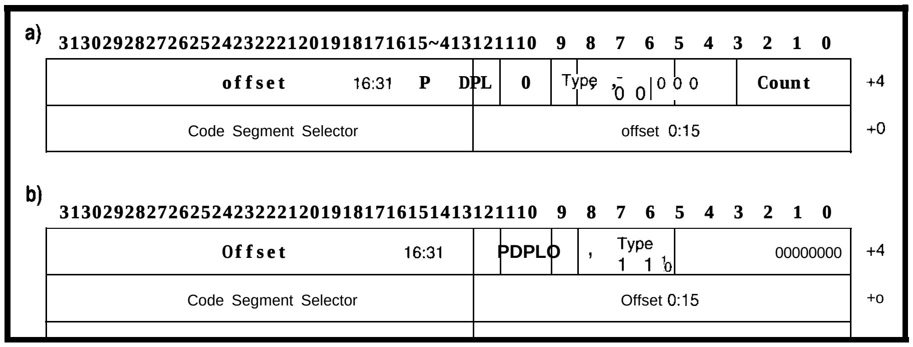


Figure 1--The 80386 architecture encourages control transfers through gates, which are special system segment descriptors. The address fields in a gate descriptor contain a code selector and an offset address within that code segment; the layout is different from the usual memory and system segment descriptors. This diagram shows gate field definitions; refer to Figure 1 in Issue 48 for the bit definitions. For 80286 compatibility, there are a/so 16- and 32-bit gates: subtract 8 from fhe 32-bit Type field to gef the 16-bit Type and make sure the high word of the address offset field is zero. a) Call gate, Type = C. The Count field is not used when fhe caller has the same privilege as the called routine because the stack entries are nof copied. b) Interrupt (Type = E) and trap (Type = F) gates. The CPU clears the Interrupt Flag to prevenf further hardware interrupts after passing through an Interrupt gate. The Interrupt Nag is not changed by trap gates to allow hardware interrupts in trap handlers

Listing 1--This structure contains the fields for 32-bit call, interrupt, and trap-gate descriptors. Unlike ordinary memory and system descriptors, gates include the code-segment selector and 32-bit offset of the routine entry point. The CopyCount field applies on/y to call gafes fhaf transfer control between different privilege /eve/s, which We aren't concerned about for the time being.

```
struct DESC_GATE{
   WORD OffsetLow;      // routine offset address 0:15
   WORD CSSelector;     // routine CS selector
   BYTE CopyCount;      // stack entries to copy, 0..31
   sys_access Access;   // access bits
   WORD OffsetHigh;     // routine offset address 16:31};

typedef struct DESC_GATE desc_gate;
```

Just like IBM's ATF (which, as I recall, mutated into TPF with those branch targets agonizingly intact) it seems you can once again J M P right into the operating system!

The '386 (and '486 and Pentium) protection hardware prevents that by running the application at a low privilege level and forbidding access to the more-privileged OS code. In any event, it's a Bad Idea to hardwire somebody else's addresses into your instructions.. that's obvious by now, isn't it?

The 80x86 architecture encourages a different method: passing control through a gate. Just as a barnyard gate restricts the flow of cattle, a programming gate restricts a program's flow of control. An application can invoke an operating system function through a gate without knowing the function's address, so the OS can change without affecting the application program at all.

Gates fall into four main categories: call, interrupt, trap, and task. Call gates correspond to ordinary function CAL Ls in application or system software, while interrupt and trap gates are used in the IDT to pass control to interrupt and error handlers. Task gates will become vitally important when we begin using the 80386 multitasking hardware.

Figure 1 shows the fields within call, interrupt, and trap gates. The address fields are slightly different from the memory and system segment descriptors I showed in *CAJ* 48, but the P and DPL bits serve the same functions. All of our P bits will be set to indicate that the segments are present, and we run with DPL=O to gain maximum privilege.

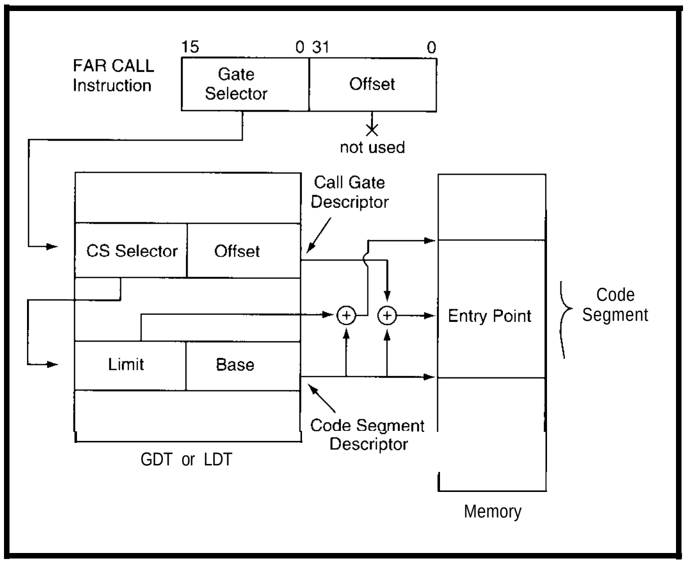


Figure 2-A *call gate transfers control to a function in a new code segment using two descriptors. The process starts with a FAR CA L L containing a selector corresponding to the call gate's descriptor; the four-byfe offset in the CA L L instruction is ignored. The call gate descriptor contains the new code segment's selector and the entry point offset. The CPU loads the new code segment descriptor, adds the call gate's offset field to the segment base address, and branches to the function. The process is much more complex if the programs use the CPU's privilege hardware,* but *we'll ignore that for now.*

Int ID	Definition	Type	Error Code
00	Divide By Zero Error	Fault	No
01	Step/Debug	Trap	No
02	NMI pin	Interrupt	No
03	Breakpoint	Trap	No
04	INTO (Overflow)	Trap	No
05	Bound Check	Fault	No
06	Invalid Opcode	Fault	No
07	x87 Not Available	Fault	No
08	Double Exception	Abort	Yes, alwavs 0
09	x87 Seg Overrun	Abort	No
OA	Invalid TSS	Fault	Yes
OB	Seament not present	Fault	Yes
OC	Stack fault	Fault	Yes
OD	General protection	Fault	Yes
0E	Page fault	Fault	Yes, special
0F	Reserved		
10	x87 Error	Fault	No
11	Alignment check (486 only)	Fault	Yes, always 0
12–1F	Reserved		

Figure 3—*Intel reserves the first 32 IDT entries for the CPU hardware. About ha/f remain unused, but we'// not make the same mistake as IBM did in the Original PC; as far as we're concerned, "reserved" means "hands off!" The distinction between Faults, Traps, and Aborts is simple: faults are detected before the instruction is complete, Traps occur after the instruction is complete,* **and** *Aborts are catastrophic failures. The NMI interrupt is a special case that is recognized at the next interruptible point between instructions. The CPU pushes an error code on the stack if the error can be associated with a particular segmenf or value; the error handler can use the code to track down and resolve the problem.*

Gates provide so much isolation that the function need not even be present in RAM when the application calls it. If the P bit is zero, the CPU invokes the "segment not present" handler when the application uses the gate. The error handler figures out which segment caused the error, determines which disk file contains the code corresponding to that gate, reads it into RAM (perhaps swapping another segment out in its place), marks the gate "present," aims it at the code, then reexecutes the CAL L. This time it will work normally, so the application doesn't know anything out of the ordinary happened!

Complex enough for you? The Firmware Furnace Task Switcher won't use this feature, but it's the sort of thing you can do if you feel the need. Talk about a simple matter of programming..

Listing 1 is the C structure that defines the fields within gates. The C opy C o unt field is used only in call gates between code of different privilege levels; it holds the number of stack entries to copy between the stacks at each level. Fortunately, because we are running at the highest privilege level, we don't have to worry about that nuance just yet, but we'll get into it eventually.

For compatibility with older 80286 code, there are also 16-bit versions of these gates formed by subtracting 8 from the Type field. These gates push the 16-bit IP rather than all 32 bits of EIP onto the stack. So, if the high-order bits of EIP were nonzero when the gate is invoked, a RET can't take you home again! The error handling code this month uses 16-bit gates, but will likely be the last time you see them in a 32-bit program.

Call gate descriptors must be located in the GDT or LDT while interrupt and trap gate descriptors must be in the IDT. Invoking a call gate is a simple matter of executing a FAR CALL or JMP with the gate's

ISO 9001 Certified

Data Acquisition & Control Solutions

Temperature Measurement & Control Solution Package – $895

Key Features:

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

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

FREE!

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

1-800-800-6889

Industrial & Lab Automation with PCs

ADVANTECH®

750 East Arques Ave., Sunnyvale, CA 94086
Tel: (408) 245-6678 FAX: (408) 245.8268

#124

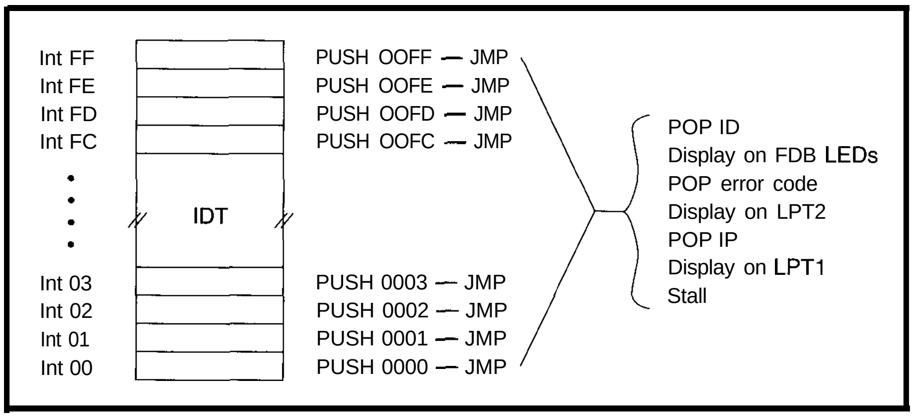


Figure 4—*Intel 80x86 CPUs invoke interrupt handlers through the Interrupt Descriptor Table. This diagram sketches the method I used to save and display the interrupt ID number without a lot of elaborate code. Each interrupt gate directs the CPU to a short routine that pushes a constant corresponding to the interrupt ID number and then branches to the common error routine. The listings in this column show the code required for each piece of this puzzle.*

selector as part of the branch address. Trap and interrupt gates are invoked by one of the INT instructions, all hardware interrupts, and any of the CPU error detectors.

Figure 2 shows how a call gate works. The CPU uses the FAR CALL's selector to locate the call gate in the GDT or LDT. That descriptor contains the code segment selector and offset for the function, along with several other fields. Fetching and loading all the descriptors is a fairly lengthy process, so passing through a call gate can take 50-100 cycles rather than the ten you're used to in real mode.

The return process is somewhat simpler. When the CPU encounters a RET instruction, it pops the selector and offset of the return address from the stack and transfers control back to that code segment. As long as you're not passing from one privilege level to another, RET isn't much more expensive than in real mode.

Because our code is running at the highest privilege level, the CPU does not invoke the checks needed to verify transfers between privilege levels. Eventually we must know about that machinery, but for now I'm keeping it simple.

Honest, this *is* the simple version!

Interrupt and trap gates are similar to FAR CALLs except that the gate is identified by an interrupt number,

Listing 2—*This function fills an IDT with 16-bit interrupt gates for the BIOS "Switch to Protected Mode" function. Each gate transfers control to one of the small routines that pushes the interrupt ID on the stack and branches to the common handler. Those routines are spaced eight bytes apart starting at PMBadIntVector. This listing doesn't include the error checking and display routines in the actual code.*

```
int BuildIDT(desc_norm ** ppIDT){

desc_gate * pDesc;
int Index;
LADDR HandlerAddr;

    *ppIDT = calloc(IDT_LENGTH,sizeof(desc_gate));
    HandlerAddr = FP_OFF(PMBadIntVector);
    pDesc = (desc_gate *)*ppIDT;
    for (Index=0; Index<IDT_LENGTH; ++Index){
        pDesc->OffsetLow = (WORD)HandlerAddr;
        pDesc->CSSelector= GDT_CS;
        pDesc->Access.Present= 1;
        pDesc->Access.Type= INT_GATE286;
        ++pDesc;
        HandlerAddr += 8;
    }
    return 0;
```

rather than a selector in an instruction (so the descriptor must be in the IDT), and the CPU pushes the Flags register on the stack before saving CS and IP. The interrupt may be caused by an INT instruction, an external hardware interrupt, or a CPU error condition. Once the gate is activated, the same actions shown in Figure 2 take place.

The interrupt handler must return using an IRET instruction to restore the Flags register. As in real mode, if you RET instead of IRET, things go awry in a hurry. Fortunately, the CPU will detect the problem and tell you what happened.

Probably the best way to see how all this works is to watch some code in action. At this point we don't have much of an operating system, so I'll exercise a pair of interrupt gates using hardware interrupts. Once again, a few blinking LEDs will either confirm that the code is running correctly or help diagnose the problem.

First, we will review the 16-bit interrupt handlers that are already in place to capture CPU errors. Then I'll aim two of the IDT entries at new 32-bit handlers and make a few response time measurements. If you're just getting up to speed with real-mode interrupts, refer back to *CAJ* 35 where I covered the grisly hardware details of your PC's interrupt system.

ERROR CENTRAL

The Interrupt Descriptor Table defines all of the interrupt handlers available to the CPU. Just as in real mode you may have up to 256 handlers, but now you can safely provide only the actual number of handlers you need because the CPU will trap any attempt to invoke a handler "off the end" of the IDT. You can also put the IDT at any convenient location and change IDTs by simply reloading the CPU's IDT Register.

Figure 3 shows the 32 interrupts reserved for CPU-detected errors. This figure is similar to one that appeared in *CAJ* 35, but we no longer have to worry about real-mode BIOS functions and hardware interrupts colliding with the Intel reserved interrupts. Those interrupts should invoke a "Can't Happen Here" handler, but they ought not be used for any other system functions.

Because an error may occur at any time, the IDT must be valid when the CPU enters PM. The BIOS "Switch to Protected Mode" requires a read/write data descriptor for GDT selector 0010 covering the IDT so it can load the IDTR correctly. That data segment descriptor gives the starting address and length of the segment, which is what the LIDT instruction requires.

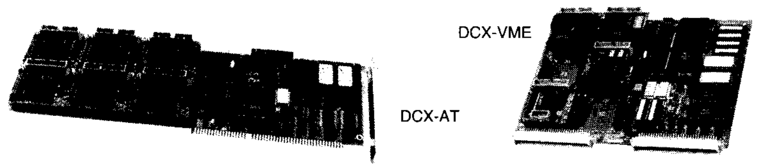
The BIOS disables external interrupts before switching modes, so I

Listing 3—This code creates 256 routines to push the interrupt ID onto the stack and branch to the common error handler. Borland's TASM extends the PUSH opcode to handle a variety of special cases, but as a result it has trouble with this simple situation. I resorted to encoding the PUSH instruction as a data byte to avoid the extended functions; there may be a better way. I also turned off the "Inefficient Code Generation" warning to avoid spurious messages resulting from the code needed to maintain an eight-byte interval between entry points. Be sure you don't let the assembler optimize the NOP instructions out of existence!

```
        PROC    PMBadIntVector
        PUBLIC  PMBadIntVector
        NOWARN  ICG             ; suppress inefficient code warning
@@ID    =       0
        REPT    256
        DB      06Ah            ; PUSH immediate byte, MSB = 00h
        DB      @ID
        JMP     SMALL PMBadIntHandler   ; 16-bit offset
        NOP
        NOP
        NOP
@@ID    =       @@ID + 1
        ENDM
        WARN    ICG             ; resume warnings
        ENDP    PMBadIntVector
```

Multi-Axis

Motion Control



1 to 8 Axis Motor/Motion Controllers

- DCXcontrollers incorporate state-of-the-art DSP, RISC; and FPGA technologies to insure maximum performance at minimum cost
- DCXcontrollers provide many sophisticated ON-BOARD control features such a MULTI-TASKING (the ability to perform independent tasks simultaneously), COMPLEX-CONTOURING, and the ability to programmed directly in USER-UNITS such as inches, meters, etc.
- DCXcontrollers can operate totally STAND-ALONE, using their own on-board "computing + memory + I/O" capabilities, or EMBEDDED in a machine. They can also be installed in an ISA-bus (PC/XT/AT), or in a VME-bus (6U)
- DCXcontrollers have a MODULAR ARCHITECTURE for creating a limitless variety of Application Specific MULTI-AXIS controllers. They are created, in minutes. by installing one or more intelligent *DCX function modules* on an intelligent ***DCX** motherboard*
- DCXcontrollers are supplied with (free) EASY-TO-USE, POWERFUL SOFTWARE for set-up, programming and operating. These include all major HLL interface programs, with examples and source code, a CNC control user-interface with G-code and HPGL capability, and an interactive Servo-Tuning utility

Precision MicroControl
CORPORATION
8122 Engineer Rd. • San Diego, CA 92111

TEL (619) 565-1500
FAX (619) 565-1511

#125

filled all 256 IDT entries with 16-bit gates leading to a single-error handler that would take control if any interrupt occurred, no matter what the cause. The references describe the differences between fault, trap, and abort interrupts, but, for our present purpose, they're all the same. If any interrupt occurs, it means the code encountered a problem that we'll have to fix manually.

However, using a single interrupt handler discards a key piece of information-the interrupt ID that invoked the handler. It's a characteristic of the Intel method of generating and handling interrupts that an interrupt selects its handler, but a multi-interrupt handler can't identify who invoked it.

Figure 4 sketches how I got around this problem. Each interrupt gate is aimed at a short chunk of revectoring code that pushes a precomputed constant corresponding to the interrupt ID onto the stack, then branches to the common handler. Popping the constant from the stack provides a quick and easy way to identify the interrupt's source, at the cost of an intermediate layer of code that occupies quite a bit of space.

Listing 2 shows the C code that generates the gates in the IDT using the `struct` layout shown in Listing 1. This code executes in real mode so the IDT is just another chunk of RAM within the current data segment. I allocated it from the near heap using `calloc()` to zero-fill all the unused fields, then filled in the required information.

The macro in Listing 3 generates the revectoring routines. The 256 entry points occur every eight bytes startingat `PMBadIntVector`, which meant I needed tight control over the generated code. After several attempts to keep the assembler from helping me, I finally defined PUSH as a data byte and specified the branch target as a 16-bit SMALL value.

The interrupt code is located in a 16-bit code segment and must be entered through a 16-bit interrupt gate. Even if the error causing the interrupt occurs in a 32-bit code segment the CPU will switch to 16-bit mode before pushing anything on the stack. Thus, the default data size is 16 bits and the stack will contain 16-bit entries, including the return address.

The PUSH instruction extends its immediate byte operand with a high-order zero byte and pushes a complete word onto the stack. In a 32-bit code segment this instruction would extend the byte with three zero bytes and push a 32-bit value onto the stack. If you mix code segment sizes, it can be difficult to tell what's on the stack.

Listing 4 presents the error handler, which is an exercise in brutally simple, user-hostile code. It recovers several vital pieces of information from the stack and displays them in raw binary on the various LEDs attached to the system. Later on we'll make it more attractive, but for now it gets the job done.

The 16-bit value atop the stack is the Interrupt ID pushed by the revectoring routine. The code pops this into CX for safekeeping while recovering and displaying the remaining information. The handler eventually enters a spin loop that displays the ID on the FDB's LED digits and blinks one of the decimal points to indicate that it's stalled.

The next stack entry may be an error code if the CPU generates one for

Listing 4-An *error handler should be short and simple to ensure that it works correct/y. This code recovers the interrupt ID number, error code, and return address from the stack and displays them on various LEDs. Obviously there is room for future improvement, not the least of which will be to handle the case where there is no error code on the stack! The code blinks one of the decimal points on the FDB LED digits to indicate that fhis routine is in control.*

```
        PROC    PMBadIntHandler
        PUBLIC  PMBadIntHandler

        POP     CX                  recover the interrupt ID
        NOT     CX                  flip it to make it readable

        POP     AX                  recover error code
        MOV     DX,SYNC_ADDR2     ; show on LPT2
        OUT     DX,AL                 low byte only

        POP     AX                  recover return address (IP)
        MOV     DX,SYNC_ADDR      ; show on LPT1
        OUT     DX,AL             .   low byte only

        MOV     DX,LED_ADDR       ; set up for the display
        MOV     AX,CX               recover interrupt ID
        XOR     CX,CX               set up initial count
@Stall:
        OUT     DX,AX               display new data
        XOR     AX,08000h           flip the high decimal point
        MOV     BL,5
@@Punt: LOOP    @Punt               delay for a decent interval
        DEC     BL
        JNZ     @Punt
        JMP     @Stall
        ENDP    PMBadIntHandler
```

Listing B-*Setting up a profecfed mode interrupt handler requires essentially the same steps as in real mode. This code loads a 32-bit interrupt gate into the appropriate IDT descriptor, resets the 8259 mask bit, and puts the FDB's Timer 0 into Mode 3 to produce a periodic 1 ms inferrupt. The PMUnMaskIRQ and PMLoad8254 routines are similar to the code shown in Issue 35 for real mode interrupts.*

```
LEA    EAX,[IOHandler]            ; set up interrupt vector
CALL    PMSetVector,I8259A_VECTOR_PM+5,  \
        (MASK Present) + INT_GATE,       \
        GDT_CODE32,EAX
CALL   PMUnMaskIRQ,5               ; set up interrupt controller
CALL   PMLoad8254,I8254_BASE,0,36h,I8254_TOCLK/1000
```

Listing 6—*This routine inserts an interrupt gate's attributes, code segment selector, and 32-M entry point offset address into a descriptor in the IDT. The GDT_IDTALIAS selector maps the IDT as a writeable data segment to allow this update; otherwise, the IDT is not directly addressable.*

```
PMProc   PMSetVector

ARG      IntNum:DWORD,Access:DWORD,SegDesc:DWORD,Offs:DWORD
USES     EAX,EDX,EDI,ES

MOV      EAX,GDT_IDTALIAS        ; get data access to IDT
MOV      ES,AX

MOV      EDI,[IntNum]            ; get pointer to descriptor
SHL      EDI,3                   ;... eight bytes each

XOR      EAX,EAX                 ; zero the entire descriptor
MOV      [ES:EDI],EAX
MOV      [ES:EDI+4],EAX

MOV      EAX,[SegDesc]           ; set code segment descriptor
MOV      [(DESC_GATE PTR ES:EDI).CSSelector],AX
MOV      EAX,[Offs]              ; set up 32-bit offset
MOV      [(DESC_GATE PTR ES:EDI).OffsetLow],AX
SHR      EAX,16                  ; align high word
MOV      [(DESC_GATE PTR ES:EDI).OffsetHigh],AX
MOV      EAX,[Access]            ; set segment access bits
MOV      [(DESC_GATE PTR ES:EDI).Access],AL

RET

PMEndP   PMSetVector
```

this interrupt, as indicated in Figure 3. Obviously, a more friendly handler would decode the Interrupt ID and either extract or skip the error code. This handler simply pops a 16-bit value and displays the low byte on the LEDs connected to port 0278.

If the CPU didn't push an error code, these LEDs will display the low-order byte of the Instruction Pointer rather than the error code. Given the size of the code segment, this ought to be enough to identify the failing instruction, but some sleuthing may be in order.

The low byte of the next stack entry appears on the LEDs at port 0378. If the CPU pushed an error code this will be the Instruction Pointer; otherwise, it's the CS selector value.

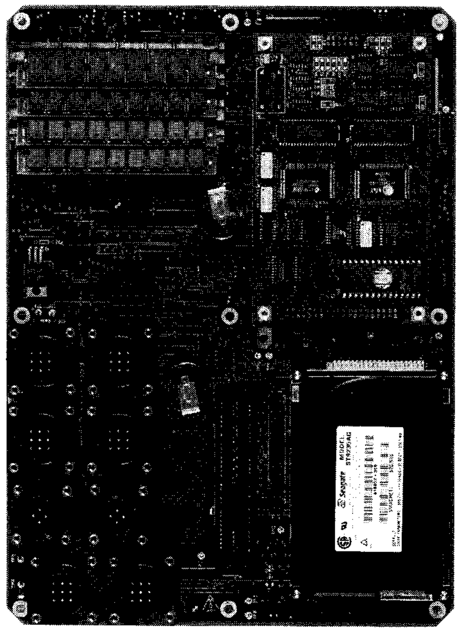
This code will give you an idea of what failed. In upcoming columns, we'll add more features to the error handler and improve the displays so they're really useful. One of the fundamental rules of building new code is to not complexicate everything at once!

Tuple, Inc.

18732 SE 43rd Street
Issaquah, WA 98027
206-746-3935 Voice
206-78 I-963 1 FAX

WANTED:
BETA SITE DEVELOPERS!

Are you tired of fighting with non-native development systems? Are you tired of circumventing non-real time system software? Are you tired of trying to make your applications work on a PC? Are you tired of trying to package your PC for factory floor applications? **The Tuple Computer is just what you're looking for!**



- **Native development/ Target 68K system (68341)**
- **ANSI 'C' Compiler, Linker, Loader, Debugger**
- **200 MB Shock Tolerant Disk On Board**
- **Graphics Co-Processor On Board (1024x768)**
- **Up to 64 MB Ram**
- **Four Serial Ports plus ARCNET ™**
- **Industrial Temp and Packaging**
- **Real Time Pre-Emptive Scheduling**
- **On Board Battery Backed RAM**
- **8 X 11 inch Board Size**
- **Expansion via Native 68K Bus**
- **768Kb Flash - On Board Programmable**

DOS

126 See us at Embedded Systems West booth #1005

Now that we have the error handler under control we're ready to try something new. With any luck you'll never see that blinking decimal point....

STAND BY FOR INTERRUPTION!

The BIOS "Switch to protected Mode" function requires a valid IDT and the new Interrupt ID numbers corresponding to the two 8259 Programmable Interrupt Controllers on the system board. Recall that the first 8259 produces Int 08 through Int OF in real mode, compare that to Figure 3, and you'll see why it must be changed.

The BIOS reprograms the 8259 Interrupt Vector Byte registers using the new Interrupt ID bytes. I picked 50 and 70 (hex) because those are the two more-or-less standard values. Refer back to the column in *CAJ* 35 for more details on how 8259s handle interrupt inputs and interface to the CPU.

The BIOS also disables all external interrupts by loading FF into the 8259 Interrupt Mask Registers. This is obviously a last-ditch attempt to prevent unexpected interrupts in PM, but it seems to me that if you've gone to the trouble of setting up an IDT, you'll have valid interrupt handlers in place.

Listing 7—*This protected mode interrupt handler simply blips a printer port bit and returns. The scope trace in Photo 1 shows that about 7 ms elapse from the rising edge of the IRQ line to the OUT instruction that turns on the porf bit. TASM 3.1 requires an IRETD to force a 32-bit operation; Version 4.0 works correctly with IRET in a 32-bit code segment.*

```
PMProc    TOHandler
PUSH      EAX
PUSH      EDX

MOV       EDX,SYNC_ADDR      ; send a trace output
IN        AL,DX
OR        AL,08h
OUT       DX,AL
MOV       AH,AL

MOV       AL,NS_EOI          ; send EOI to controller
OUT       I8259A,AL

MOV       AL,AH              ; clear trace output
AND       AL,NOT 08h
OUT       DX,AL

POP       EDX
POP       EAX
IRETD
PMEndP    TOHandler
```

Incidentally, this isn't documented anywhere I could find; I had to disassemble the BIOS code to make sure it worked this way. Sometimes it seems that all the references simply repeat each other, and you've got to actually try it out to get the straight dope. But that's what this column is all about, right?

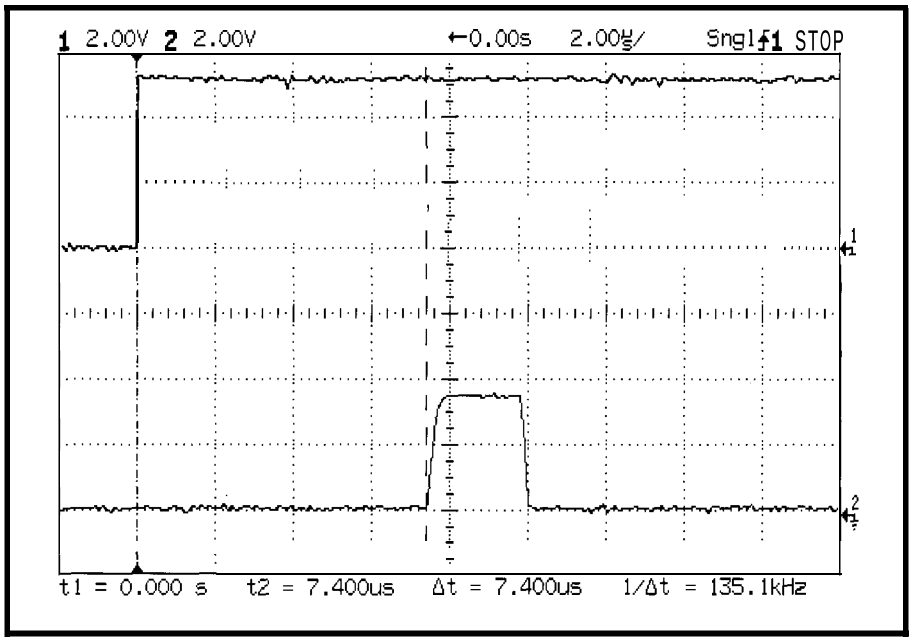


Photo I-Responding *to an interrupt fakes more time in protected mode, but the difference may not be as greaf as you think. In this case, a simple inferrupthandferdisplays a 7-ms response fime: the upper trace is the IRQ line, the lower trace is the handler's output on the parallel port. The interrupt gate in the IDT ensures that fhe CPU is in 32-bit mode during the interrupt, so all register and stack operations default to 32 bits rather than the usual 16 bits found in real mode.*

The Firmware Development Board has an 8254 that can produce an interrupt on IRQ 5. I set Timer 0 up for a 1-ms tick rate. If you haven't built the Graphic LCD interface you can also experiment with Timers 1 and 2 on IRQs 10 and 15, respectively. For those of you who haven't yet built any hardware at all, I set up Timer 0 on the system board for 5-ms interrupts on IRQ 0.

Preparing a PM interrupt handler should be familiar to anyone who's done real-mode programming. Listing 5 shows the sequence of events for the FDB timer: capture the interrupt vector, set up the hardware, then clear the 8259's IMR bit. That's all there is to it!

Listing 6 has the key difference between real and protected mode—filling in an IDT entry, rather than capturing a raw vector address. ES holds the data selector alias for the IDT that allows access to that chunk ofRAM.IusedthePUSHFDandPOPFD instructions to maintain 32-bit stack alignment.

After all that setup, the interrupt handler in Listing 7 is almost anticlimactic. It raises a bit in port 0378, sends an EOI command to the interrupt controller, and clears the bit. The result is shown in photo l-the upper trace is the rising edge of IRQ 5 on the ISA bus, the lower trace is the printer port trace bit. The overall delay is about 7 us, which includes the time required to save two 32-bit registers and twiddle the port.

The fastest real-mode handler I presented in *CAJ* 35 required 5 µs to get to about the same point. Although the PM code is slower, it's not a *lot* slower.. .and that may come as a surprise. Yes, you can write PM code that runs down on the bare metal in the microsecond range.

Maybe this 32-bit protected mode stuff isn't such a bad idea after all?

Admittedly, this is about as simple as an interrupt handler can get. When we start switching tasks and privilege levels, the response time will drag out as the CPU does more work on our behalf. We'll keep those scope probes ready to track the changes and understand what's going on.

RELEASE NOTES

The code this month checks out two interrupt sources on your system. If all goes well, you should see a pair of dimly lit LEDs on the LPT1 port monitor box and a rapid count on the Firmware Development Board's LED display. If there's a problem, the FDB will show you the trap ID number in raw binary, and LPT1 will display the error code.

Borland's TASM 3.1 had a bug that clobbered 32-bit constant calculations: you could define a 32-bit constant, but the arithmetic operators used only the low-order 16 bits. Version 4.0 fixed that, but it optimizes branches a little more aggressively and is more strongly typed, which broke some code. Although it pains me to say this, you'll need the latest and greatest Borland assembler to compile the code starting with this month's column. I will continue using C++ 3.1 for the real-mode code, though.

Shortly after last month's column went read-only, I got a flyer from Intel saying that their databooks and component manuals are now available only from McGraw-Hill. Naturally, McGraw-Hill changed all the order numbers, so you'll have to call (800) 822-8 158 for more information.

Thanks to Marshall, Lou, and Pete in Hewlett-Packard's Raleigh branch office for showing off an HP54602B oscilloscope. They knew I couldn't live without one.. .so the scope "photos" will be digital from now on.

Next month, we return to real mode and the mysteries of the DOS FAT file system to build a loader that fetches a program from disk, stuffs it into RAM, and fires it up in 32-bit protected mode. ❑

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

I R S

416 Very Useful
417 Moderately Useful
418 Not Useful

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

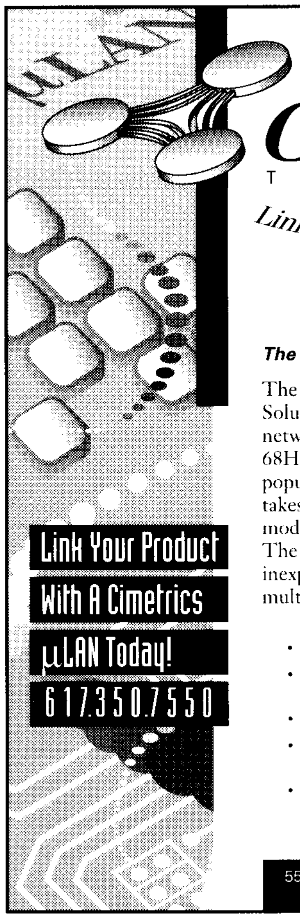
- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 805 1/803 1 or 8052/8032 CPUs*
- *Compatible with any RAM or ROM memory mapping*
- *Runs up to 50 times faster than the MCS BASIC-52 interpreter.*
- *Includes Binary Technology's SXA51 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51*** $295.

508-369-9556
FAX 508-369-9549

❑

Binary Technology, Inc.
P.O. Box 541 • Carlisle, MA 01741

#127



Cimetrics
TECHNOLOGY
Linking Microcontrollers. Breaking Boundaries.

The 9-Bit Solution

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

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

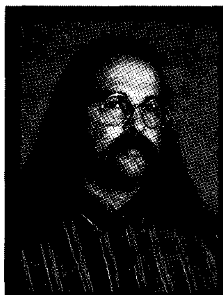
55 Temple Place • Boston, MA 02111-1300
Ph 617.350.7550 • Fx 617.350.7552

#128

Probing the Dark Side: the Motorist's Aid to Hindsight

FROM THE BENCH

Jeff Bachiochi



Motorcyclists have enough going against them when they take to the road without having to worry about missing someone in their blind spot. Jeff turns to technology to help make his next ride just a little bit safer.

Check the mirrors, the coast is clear, turn signal on, accelerate, pull out, and pass. We don't think much about this driving skill until someone cuts us off or worse yet we cut someone off. They (you) didn't mean it; you [they) were in their (your) blind spot.

An auto's blind spots are located in the lanes beside your car and run from just behind your vehicle to the position immediately adjacent to the driver. The rear view mirror doesn't cover these areas, and the side mirrors cover only a small section of them. Convex mirrors, which either clip onto the rear view or stick onto the side mirrors, give the driver a wider field of view, but they also distort distance. And, if you ride a motorcycle you're stuck with just side mirrors.

One solution to this problem might be to never drive without a copilot. The copilot's job would be to ride backwards, thereby gaining an unobstructed view of oncoming traffic. This, of course, would cause many communication problems: "Car fast approaching on right. Swerve left.. .No! My left-not yours." And, although permissible in an auto, most patrol officers would frown on seeing motorcyclists riding back-to-back.

I find myself riding my VT700c Shadow to work more often these days. Not because my oldest son Dan is home from college and needs to borrow the car every day, it's just that cruisin' helps break down that built-up stress, granting short recesses from reality. It can also be quite an olfactory sensation, especially living in rural farm country.

This month's project requires a moving test bed. My trusty iron horse will be the recipient of not just a pair, but five eyes-eyes strategically positioned toward the rear, rear corners, and sides of my bike, covering all the blind spots. (Please note that this is an aid and not a replacement for mirrors or taking a good look and see.)

EYES HAVE IT

To perform the part of the eyes, I call on the Polaroid sonar ranging module (SRM) with multiple transducers. Multiplexing the transducers reduces the number of transceiver modules necessary to one. Otherwise, costs could easily become unreasonable as the number of zones increases.

A PIC processor is responsible for selecting a transducer, initiating the ranger, measuring any reflections, and reporting these to the external display unit. Transducer voltages are quite high [a few hundred volts) and, even though the currents are not a problem, I want to ensure that the relays won't have to switch live HV, but merely direct it. I used a '238 decoder to select transducer channels and to provide a master gate control over the actual output selection devices (see Figure 1).

The information collected by the PIC could easily be too confusing for the driver to interpret on the fly. Although I want direction and distance, I don't want to have to read messages or even distances to use this system. Output must be kept simple.

So, I designed a display which is broken down into five regions in the pattern of the transducers. Each region has three LEDs that represent zones of interference. The green LED represents the outermost zone, the first zone entered by a target. The yellow LED signifies an intermediate zone, and the red LED depicts the closest zone to the transducers. Although the SRM can give very accurate distance measurements, I select only three distances and indicate when a target moves within these respective zones. Figure 2 illustrates the display.

SUPPORTING CIRCUITRY

In addition to the SRM, a small amount of support circuitry is needed

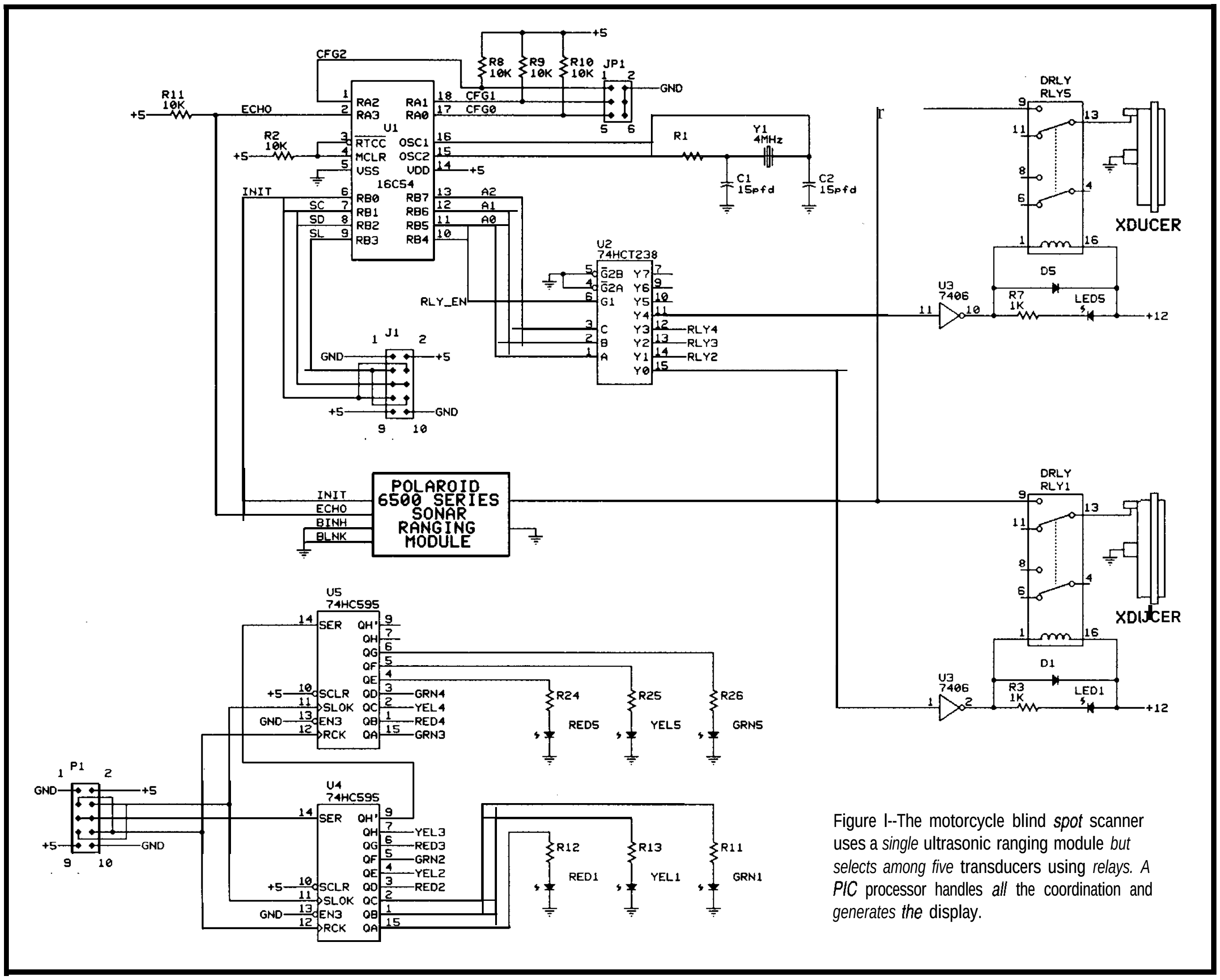


Figure l--The motorcycle blind *spot* scanner uses a *single* ultrasonic ranging module *but selects among five* transducers using *relays. A PIC* processor handles *all* the coordination and *generates the* display.

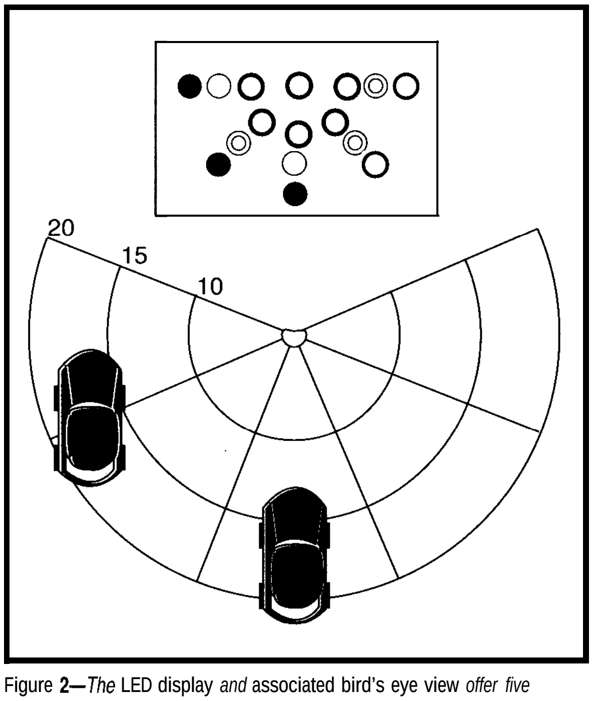


Figure **2—***The* LED display *and* associated bird's eye view *offer five zones and three* distance *ranges.*

including a microprocessor, a decoder, a driver, and two serial-to-parallel shift registers. While any small micro will do, I chose the smallest PIC available-the 16C54 (512 words of code space)-and still used only half of the available RAM and code space.

Mechanical relays direct the HV 50-kHz bursts from the single ranging module to one of the five transducers. Each transducer covers a 45" zone. The relays are fully debounced prior to releasing the burst of energy and remain energized while the ranger listens for an echo. Since the relays are going to be running continuously, mechanical life becomes a factor. Mechanical life is listed at about 10 million operations. My maximum duty cycle (limited by the SRM's maximum listening time) is 2.5 cycles/second. That's 9000 cycles/hour or about 1111 hours. On the motorbike, it would last over 60,000 miles at 60 mph.

While most of the electronics are within the enclosure that mounts on the rear of the bike, the display (LEDs) needs to be in front of the driver. A five-conductor shielded cable connects the two pieces of circuitry. Power, ground, and three shift register signals-data, clock, and latch-are used. Each shift register has eight latched-output bit positions and a serial output (used for chaining). Fifteen outputs are needed-one for each of the three distance zones within

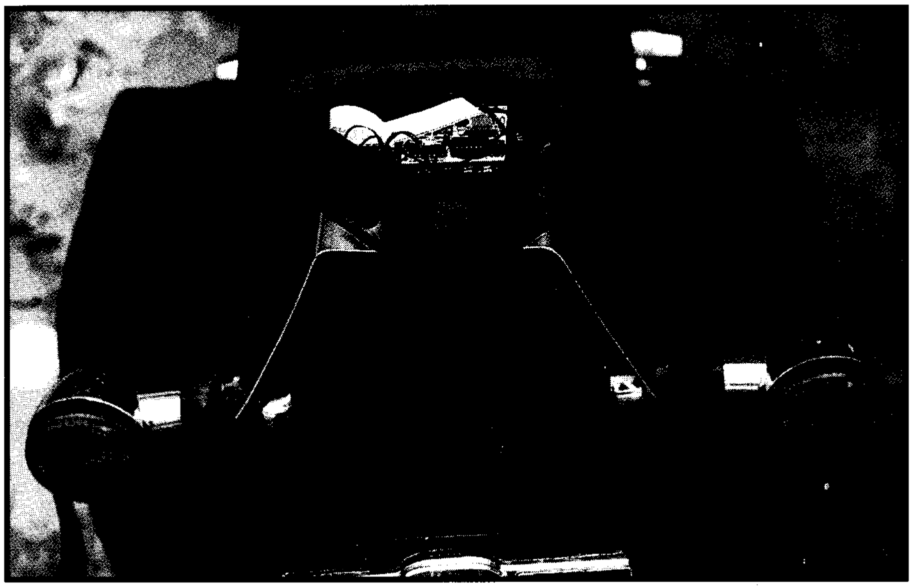


Photo 1—*The transducer package mounts on the* back of the motorcycle *using some spare luggage rack bolts.*

the five regions. At the last minute, I made use of a sixteenth bit to provide a heartbeat LED just to verify that the system is actually executing.

The electrical system of most vehicles is very noisy, so I used a hefty hash choke and capacitor on the input to cut the noise prior to the 5-V regulators. I used one regulator for the microprocessor (and display circuitry) and one for the SRM. The system requires about 300 mA operating current (depending on the number of LEDs on) plus about 2 A during transducer bursts. Using separate regulators prevents current spikes from affecting the micro. A good-sized, heat-sinking surface became available when I constructed a mounting bracket from aluminum scraps I had laying around the shop.

GO WITH THE FLOW

If you refer to the flowchart in Figure 3, you will see how simple this control program is. After initializing the PIC's ports, the configuration port is read. Eight possibilities can be selected, although at present I am using four. The configuration jumpers are used to choose the alert points (distance from the transducer) for each of the three colored zones (see Table 1).

Configuration 0 is used for testing purposes for those of you who don't have 30' rooms without obstacles. The maximum distance of the sonar ranging module is about 35' and is

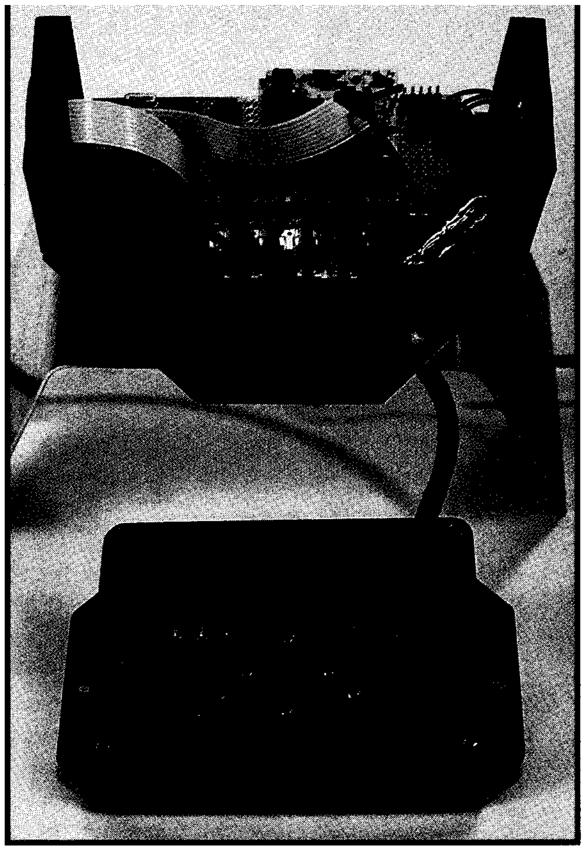


Photo 2—*The handlebar display unit shows three distances in five different zones plus blinks another LED to show it's alive.*

AMX™

The Real-Time Multitasking Kernel

680x0, 683xx
80x86/88 real mode
80386 protected mode
i960® family
R3000, LR330x0
Z80, HD64180

Features

NEW DOS Compatible File System TCP/IP

- Full-featured, compact ROMable kernel with fast interrupt response
- Preemptive, priority based task scheduler with optional time slicing
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility eases system construction
- InSight™ Debug Tool is available to view system internals and gather task execution statistics
- Supports inexpensive PC-hosted development tools
- Comprehensive, crystal clear documentation
- No-hiclden-charges site license
- Source code included
- Reliability field-proven since 1980

Count on KADAK.
Setting real-time *standards since* 1978.

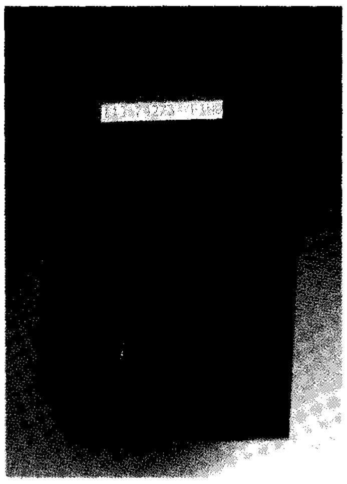
For a *free* Demo *Disk*
and your copy of our excellent AMX product description, contact us today.
Phone: (604) 734-2 796
Fax: (604) 734-8114

KADAK Products Ltd.
206 - 1847 West Broadway
Vancouver, BC, Canada V6J 1Y5

AMX is a trademark of KADAK Products Ltd.
All trademarked names are the property of their respective owners.

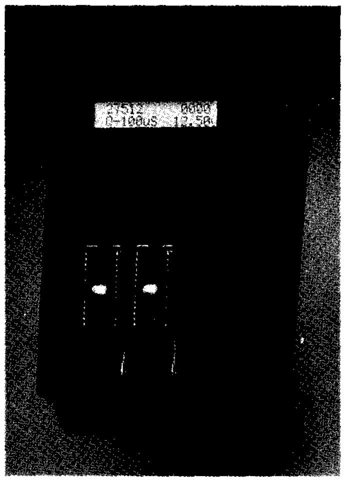
#130 See us at Embedded Systems West booths #940–942

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



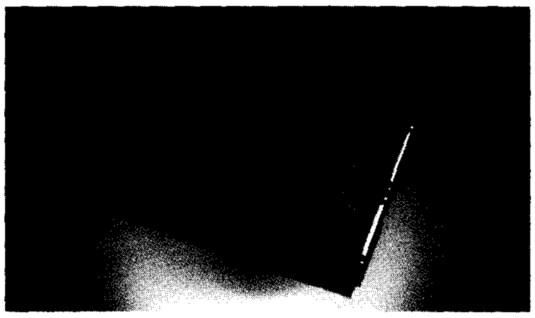
HT-2 8 *Express*

The HT-28 is a very convenient way of testing Logic IC's and DRAM's. Tests most TTL74, CMOS40/45 and DRAM's 4164-414000, 44164-441000. It can also identify unknown IC numbers on TTL 74 and CMOS 40/45 series with the 'Auto-Search' feature.
$189.95



HT- 14 *Express*

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



P-300

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

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

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

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

#131

limited by maximum amplifier gain, which is used to hear those distant reflected echoes.

Next a channel (transducer) is selected and a short, debouncing loop is entered which assures the relay's contacts are settled prior to enabling the HV transmitting pulses. Raising INIT on the SRM begins the measurement cycle. Every millisecond, the ECHO line is polled for a logic high which would indicate that a reflected echo has been detected. The elapsed time is saved, and the routine is ended.

Now the elapsed time register is compared to each of the zone points originally set by the configuration jumpers. The first three bits of the present channel's status register are updated with either 1s or 0s to indicate whether there is a target in or out of the three zones. These first three bits of each of the channel's status register are shifted out to two external shift registers located in the display. Green, yellow, and red LEDs are connected to the shift register's outputs. Each channel's status bits should end up shifted out to the corresponding LED.

Finally, the channel is incremented (or cleared) and a jump made back to the top of the loop. Loop timing is dynamic and based on the time needed to receive an echo (or timeout).

PROS AND CONS

There are some disadvantages to using a single SRM and multiple transducers. With five transducers, each zone can be updated 2.5 times/second. At 60 miles per hour, a target travels 88 feet per second. Since 88 divided by 2.5 is 35 feet, a target can go from out-of-range to crashing-into-you in a single sample time. Fortunately, we are concerned with relative speeds (i.e., the difference between your speed and the speed of the approaching target). This is usually under 10 mph which is about a car length per second.

Decreasing the number of transducers or increasing the number of SRMs improves the update timing. You must determine what rate is necessary for your particular application. In robotics, you might use eight transducers in a 360" pattern and still have better than I sample/second using only a single SRM. (If I were to use eight transducers to get a full 360" display of my driving arena, I would add a second module and get better than 3 samples/channel/second.)

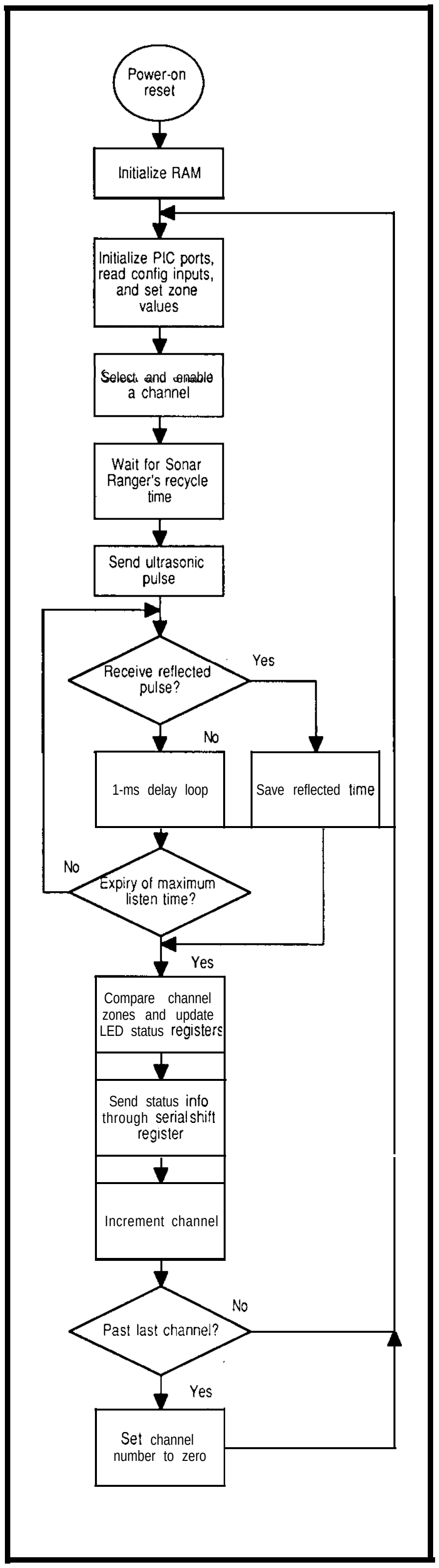

Power-on reset
Initialize RAM
Initialize PIC ports, read config inputs, and set zone values
Select and enable a channel
Wait for Sonar Ranger's recycle time
Send ultrasonic pulse
Receive reflected pulse?
Yes
No
1-ms delay loop
Save reflected time
Expiry of maximum listen time?
No
Yes
Compare channel zones and update LED status registers
Send status info through serial shift register
Increment channel
Past last channel?
No
Yes
Set channel number to zero

Figure 3-The *P/C's control program constantly polls each ultrasonic* transducer *and* displays fhe *results on the* operator *console.*

The display I am using reminds me of fuzzy logic. It's not so important for me to know the exact distance; however, I do want to know if an object is far, near, or close and whether it is approaching or falling back.

ROAD TEST

Securing the transducer assembly to the rear of the motorcycle is easy (see Photo 1). Bolts for mounting an optional luggage rack are presently not used. When a small group of "the guys" go camping in the spring and fall, we like to keep in touch with each other using CBs. So, I already had a handy 12-V connection available right underneath the passenger seat. I fished the display cable beneath the seat and tank and mounted the display to the handlebars. I donned my helmet and was ready to cruise.

Leaving the parking lot, I quickly noticed the LEDs extinguishing. The clutter of the lot was gone and all I was registering was the guardrail along my right side. As I made my way through a small neighborhood, parked cars blipped the display. While waiting for "the green" at the expressway's entrance, all the LEDs illuminated one by one as vehicles pulled in around me.

Now for the highway test. I proceeded up the ramp accelerating to meet a pack of roaring semis. I don't enjoy being boxed in between 18 wheelers so I pulled into the middle lane and let traffic zoom around me on both sides. The display activity coincided with the traffic movement. However, the activity in the two corner zones could be improved a bit.

The next exit brought me back into stop-and-go town traffic. The vehicles were much closer now and the display reflected this. I contemplated this while making my way to the Circuit Cellar world headquarters.

EPILOGUE

This project won't end here. There are three improvements I want to make. First, the colored LEDs I used are difficult to see in the daylight because they are tinted with color. Clear LEDs would be much easier to see; they actually change color.

Config	Alert Zone (distances in feet)		
Value	Red	Yellow	Green
0	2	4	6
1	5	10	15
2	5	10	20
3	10	20	30

Table I-The *unit can be* configured *for one of four* operating *modes, with each mode using a* different set of ranges *for the three alert zones.*

Second, I would like to supply each transducer with its own defined zone definitions. This way the corners can use larger zones than the sides or the back.

Last, I'd like to mount a switch on the display panel to control one of the configuration switches. This would let the driver change zone definitions from a city setting to a highway configuration on the fly.

This project uses Polaroid's standard transducer which comes with the SRM offered by Circuit Cellar Kits (TI01).

Have fun experimenting with this project. I hope to meet you somewhere out there on the road. Happy trails!

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

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.

SOURCE

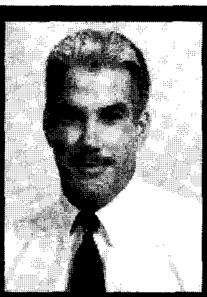
The TI01 ultrasonic ranger is available from Circuit Cellar Kits, 4 Park St., Vernon, CT 06066, (203) 875-2751, fax: (203) 872-2204. Price is $79.

419	Very Useful
420	Moderately Useful
421	Not Useful

SILICON UPDATE

Tom Cantrell

PID-Pong: Point, Set, Match: Using a Hitachi H8 for Real-time Control



Just when you thought it was safe to bring out your ping-pong balls again, Tom reintroduces his PID-Pong challenge. He now has a winner, and he describes just what victory required.

I'm pleased to report that the PID-Pong challenge has been met and ably overcome. The victor's name is PIDDLE (PID Design, Learning, and Experimentation), a combination of a high-integration, 8-bit micro running PID software and monitoring software on a PC.

For those of you who missed last January's article ("PID-Pong Challenge," *CAJ* 42), the original goal was to develop an interesting and educational platform to serve as a feedback-control-demonstration vehicle. The result was the PID-Pong machine, a lashup incorporating a ping-pong ball, plastic tube, 12-V fan with PWM (Pulse Width Modulator) motor controller, and a Polaroid TI01 Ultrasonic ranger (see Figure 1 and Photo 1).

PID-Pong is able to sense and control the ball position, so the challenge is to move the ball quickly and accurately between setpoints. I was delighted to discover that the machine isn't a pushover. Indeed, it's practically uncontrollable by mere humans. Due mainly to severe (i.e., a second or so) fan inertia ["lag" in PID-speak), the usual result is the ping-pong ball shoots out of the tube in a mortar-like fashion.

You'd be amazed at the amount of havoc a flying ping-pong ball can wreak in a cluttered office. It calls to mind the chaos theory parable of a butterfly flapping its wings causing a hurricane an ocean away. Well, the ping-pong ball taps the messy pile of magazines, that slides into the coffee cup, which sloshes on the disk..

Having a machine but needing a controller, I took Thomas Edison's words, "Genius is 1% inspiration and 99% perspiration," to heart. Thus, 99% of the credit for PIDDLE goes to Aleksander Hanslik, Andrzej Sitek, and Mirek Chojnacki of Hanslik Software Laboratory (Katowice, Poland). They are to be congratulated for writing a lot of great code while tactfully disabusing me of my more outlandish inspirations.

YAP, YAP, YAP

Before describing PIDDLE, I guess now's the time to launch into the obligatory Yet-Another-PID discourse. I can pursue one of two tacks.

First is the complex math treatise, complete with lots of differential equations, Z-transforms, Bode plots, Routh's Criterion, and so on. Fortunately for me, those who are capable and so-inclined can refer to any number of well-written tomes. In particular, I relied on the "Control Theory" chapter of the *Instrument Engineer's Handbook* (Liptak and Venczel, Chilton Book Company, ISBN# o-8019-7290-6).

Another approach is to reference a real-world situation such as braking a car for a stoplight. I like this idea, but this particular oversimplified analogy fails because it underestimates the skill (and foolishness) of all but first-time drivers. In fact, seeing commutoids making phone calls, drinking coffee, fixing their hair-indeed, all at once-indicates that driving is practically an open-loop process these days.

Back in the days when "muscle car" meant more than a taxi full of professional wrestlers, one popular "benchmark" was the 0-100-O test in which the goal was to go from a standing start to 100 mph and back to a stop in minimum time. Unlike the typical O-60 or %-mile spec, this reflected justifiable concern about the mismatch of a 500-HP motor against feeble drum brakes and iffy tires.

In these days of asthmatic wimpmobiles, a more realistic driving analogy is the Type-A executive driving a rent-a-wreck. You've surely seen such stoplight-to-stoplight Marios and probably wondered how they made it through childhood without reading *The Tortoise and the Hare.* That's what makes driving (and process control) interesting-real-world "constraints" ("If I don't get to that meeting on time, I'm history") and "disturbances" (death-wish pedestrians, flat tires, bad gas, etc.).

So, let me just leave the mumbo-jumbo behind and keep it as simple as I can (which is real simple). PID refers to the Proportional, Integral, and Derivative control technique. The control output is set depending on P, I, and D functions of the "error" which is simply the difference between the current position-where you are-and the desired-where you want to be (i.e., the setpoint)-state. Thus,

$$\text{OUTPUT} = (P*E) + (I*E) + (D*E)$$

Actually, various combinations of the factors are possible. P-only, PD, PI, and PID are most often used.

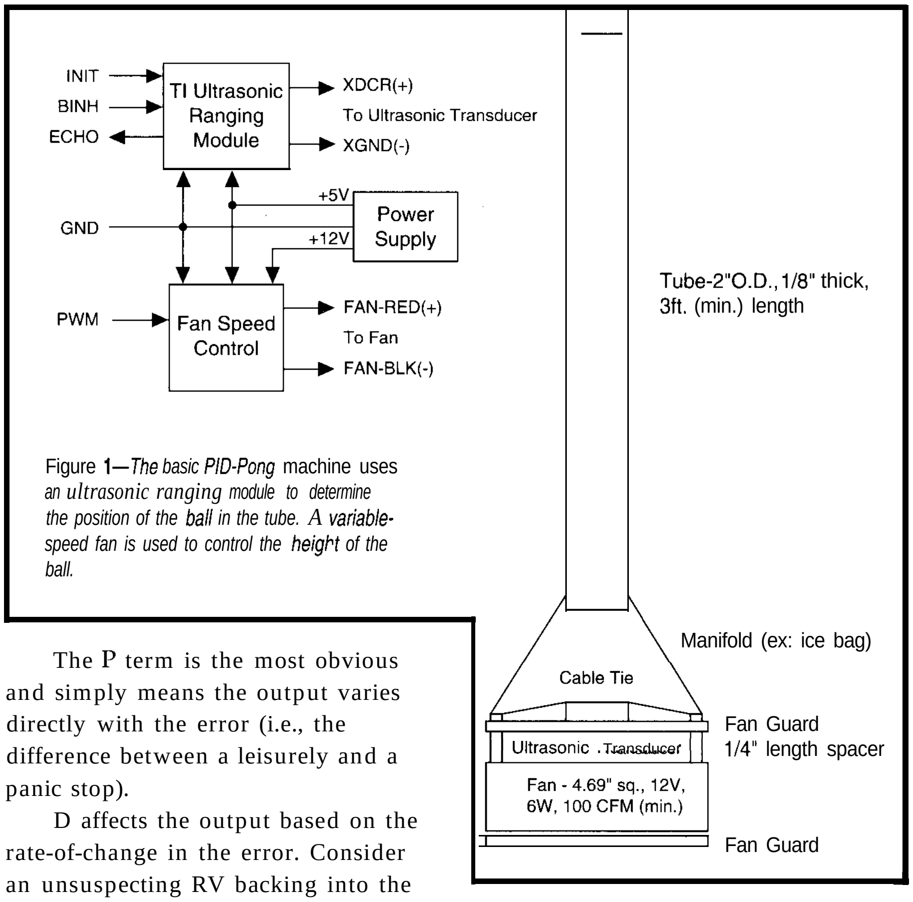


Figure 1—*The basic PID-Pong machine uses an ultrasonic ranging module to determine the position of the ball in the tube. A variable-speed fan is used to control the height of the ball.*

The P term is the most obvious and simply means the output varies directly with the error (i.e., the difference between a leisurely and a panic stop).

D affects the output based on the rate-of-change in the error. Consider an unsuspecting RV backing into the street 100 feet ahead. Without a D factor, the P-only braking response will be the same whether you're going 10 or 100 mph, the latter case likely leading to a particularly ugly case of "terminal overshoot."

I is a little more subtle and refers to an accumulation of previous errors. The best example is the Joe or Jane Procrastinator who insists on driving with worn-out brakes. Though they drown the brakes' dying screams by turning up the radio, eventually they notice they're stopping halfway into the intersection. The I term belatedly kicks in and they begin to adapt, first by braking earlier and when that gets old, perhaps resorting to abusive downshifts, flinging the doors open in a dive-brake-like manner, using their briefcase as an anchor, and so on.

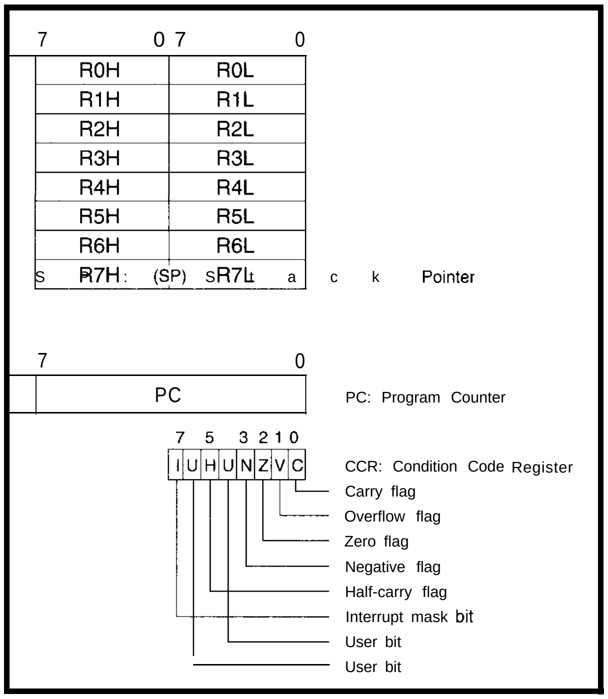


Figure 2—*The H8/325 register file consists of eight 16-bit registers that can be accessed in high or low B-bit chunks.*

Other PID embellishments have their corollary on the road. Output limiting used to be a simple byproduct of the fact that the pedal would go to but not through the metal. Today, onboard micros actively limit output with ABS (antilock braking system) and TCS (traction control system). The latter are usually switch activated—after all, you can't blame someone who spends enough to buy a heap that can get out of its own way for wanting to "light 'em up" once in a while.

"Antireset windup" is an arcane-sounding term describing exceptional start-up situations in which the state initially (i.e., at reset) resists change. In this situation, an integral term can accumulate outrageously (i.e., "wind up" resulting in overshoot while it "unwinds"). Consider the poor fool who stalls his car when the lights turn green. Invariably, even an otherwise smooth driver will launch frantically when they finally get it started. You see, they're simply trying to "unwind" the integral error of their ways, no doubt encouraged by the honking and hand gestures of the myfarcating Type As behind them.

Finally, PID calculations can be tweaked to deal with a degree of the nonlinearity that characterizes many real-world control problems. A car analogy is the way throttle linkage (or software in emerging "fly-by-wire" systems) damps initial throttle response in the interest of smooth starts, lest parking lot maneuvers turn into destruction derbies.

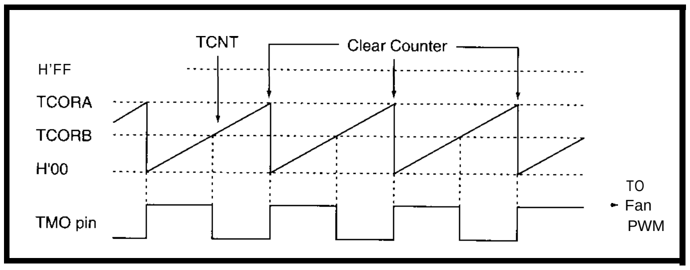


Figure 3—The PID-Pong machine's fan is controlled by a single pulse-widfh-modulated TTL bit.

SMALL BLOCK H8

Casting about for a suitable PIDDLE engine, we settled on the Hitachi H8/325, a relatively low-cost, 8-bit single chipper featuring a healthy complement of on-chip I/O, including 8-and16-bit timer/counters, UARTs, parallel I/O, and so on.

Deciding up front to use C, the ugly specter of bloated code raises its head. Any of you who have actually tried to cram a C program (especially with floating-point calculations] onto an 8-bit single-chip CPU know what I'm talking about. The H8/325 packs a relatively whopping 32 KB of ROM/ EPROM and 1 KB of RAM on chip, hopefully keeping "OUT OF MEMORY" messages at bay.

Architecturally, the H8/325 is blessedly simple. The register file (Figure 2) consists of eight 16-bit registers which can be accessed in high or low S-bit chunks as well. Thus, most of the 57 basic instructions are offered in byte and word versions.

The somewhat minimalist instruction set reflects the migration of RISC concepts off the desktop and into controllers. Notably; the H8/300 is a "LOAD/ STORE" machine in which all instructions reference only registers with the sole exception of loads and stores (MOV on the H8) that shuffle operands and results to and from memory. However, beyond this, the RISC concepts are tempered with various doses of reality.

For instance, instructions occupy either two or four bytes, which represents a tradeoff between code density and circuit size or speed, factors which are, respectively, enhanced and degraded by variable-length instructions.

Similarly, instruction execution time (@10 MHz) varies from 200 ns to 1.4 us (MLT/DIV) with a likely average of 300-400 ns or so for a typical mix, which is quite competitive with other 8-bit micros. Sure, the multiinstruction-per-clock RISC zealots would poo-poo such numbers, but they should keep in mind the H8 probably costs less than the socket for their 32- or 64-bit Superdupers.

The timer/counters (one 16-bit and two S-bit) deserve special examination since it turns out that they mate very well with the PID-Pong machine.

You'll remember that the output from the PID controller to the machine (i.e., the fan speed setting) is a single pulse-width-modulated TTL bit that sets the fan duty cycle (0/255 through 255/255). As shown in Figure 3, the S-bit timer easily handles the task-thanks to two timer-compare registers (TCORA, TCORB). The TMO timer-output pin is automatically inverted on each match with the FRC (Free Running Counter). Thus, having loaded TCORA with 255 and started the timer, setting the fan power is as simple as poking the duty-cycle factor into TCORB. The PWM output dutifully proceeds via TMO with absolutely no further software intervention required. Nice!

Talking to the PID-Pong machine's ultrasonic sensor is also easy, thanks to the smart 16-bit timer/ counter. Driving the TI01 calls for two outputs to toggle the INIT and BINH (Blank Inhibit] lines with a phase delay representing the "blanking time" (i.e., the delay from INIT to BINH in which the ECHO input should be ignored to avoid false detection). Well, check out Figure 4, and you'll see that the 16-bit units, two output-compare registers (OCRA, OCRB), and output pins (FTOA, FTOB) handily fill the bill, establishing both the sampling (INIT to INIT) and blanking (INIT to BINH) time.

But, it gets even better. The feedback from the PID-Pong machine (i.e., ball position) is determined by monitoring the TI01 ECHO output. The idea is to measure the elapsed time between the assertion of INIT and the receipt of ECHO, with position determined by the speed of sound. Well, what do you know-the 16-bit timer/counter also includes an input capture pin (FTI) that, when asserted, latches the value of the FRC into an input capture register (ICR).

Thus, once everything is rolling, determining the ball position is as simple as reading the ICR whenever you want-it will always contain the most recent ECHO time. The 16-bit unit doesn't have the feature to automatically toggle the output pins so OCRA and OCRB interrupt handlers are required. However, they are short (3 instructions) and sweet since they only need to toggle the pins (INIT and

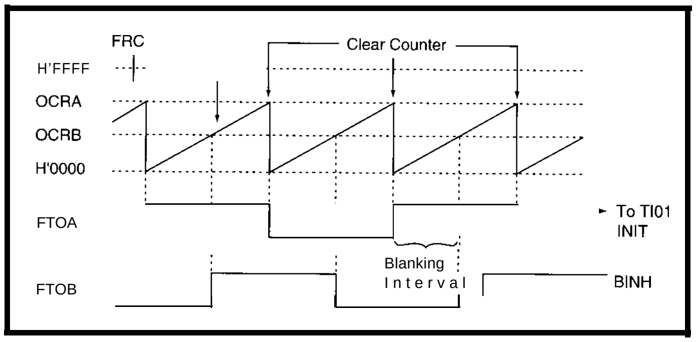


Figure 4-Jha H8's onboard timers can a/so automatically hand/e the TI01's INIT/BINH timing.

BINH pins, respectively), clear the interrupt, and return. Noting the sampling rate is a leisurely 20 Hz or so, a little calculation shows "trivial" (≈ 0.01%) is the right word to describe interrupt overhead.

Getting the TI01 working is one of those things that's easy-after you've done it. Ironically, the major stumbling block isn't the I/O timing, but the fact the darn ECHO pin is "almost" TTL compatible. I myself have spent more time than I care to admit head scratching over an "almost" working TI01 before I remembered to put a stupid pull-up on ECHO. Fortunately, all H8 inputs (including FTI) offer internal pull-ups-just remember to enable them in software.

I "C" A SOLUTION

Besides a little ASM to set up the timers, return the ball position, adjust the PWM, and so on, the rest of PIDDLE is written in C. It was a pleasant surprise to find the entire routine was only a couple of hundred lines, the guts of which (P I D LOO P) are shown in Listing 1. Let's step through it, and you'll see that it's actually quite straightforward.

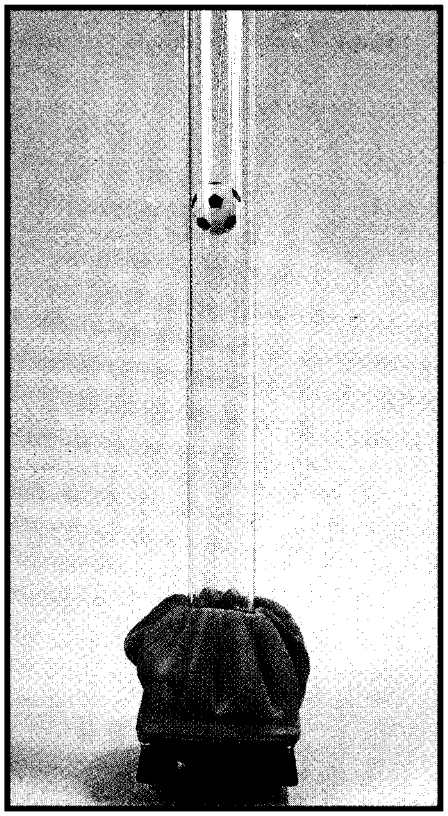


Photo 1—*The PID-Pong Challenge is to come up with a scheme that will automatically maintain the pin-pong ball in one position in a tube on a cushion of air.*

As expected, the PID loop starts by calculating the current error (e) which is simply the setpoint (x) minus the current ball position (y).

The next few statements are by far the trickiest and require a little more explanation. The goal is to derive a nonlinearity factor (FNL) that is applied to change the gain depending on the magnitude of the error. When the ball starts to move to a new setpoint, the error is large and high gain is called for. However, as the ball approaches the new setpoint, gain should be reduced for finer control. The nonlinear version eases what's otherwise a "choose your poison" juggling act between high (may overshoot or oscillate) and low (slow response] gain.

This adaptation relies on RFACT, a tuning factor which controls the gain multiplication. Roughly speaking, the gain is doubled for every RFACT percent increase in error. Thus, if

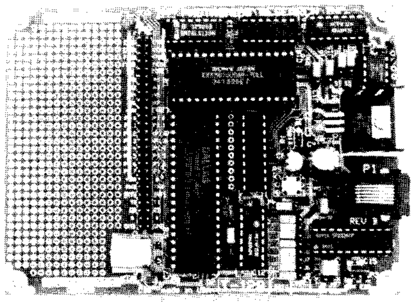
TURBO-128
THE *NEXT GENERATION* EMBEDDED CONTROLLER

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

Photronics Research Introduces the T-128 A True Single Board BASIC Development System The T-128 is based on Dallas Semiconductor's new 8051-compatible DS80C320. With its 2X clock speed (25MHz) and 3X cycle efficiency, an Instruction can execute in 160ns an 8051 equivalent speed of 62 5MHz!!! Equally impressive is the T-I 28's high-speed NVRAM Interface Any of the 128K RAM may be programmed directly from a PC file through the console, eliminating EPROMs and associated tools Program Development has never been faster or more convenient, even with the finest EPROM emulator The T-128 features PORT 0 bias and EA-select for DS87C520 upgrade

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

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



I/O
- Three S-bit Parallel Ports
- Two Full-Duplex RS232 Serial Ports
- Decoded Device I/O Strobes
- SO-Pin Bus Connector

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

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

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

UPGRADE [available soon]
- DS87C520 processor (33MHz)
- Instruction cycle: 121 ns
- 8.25 MIPS
- 8051 equivalent: 82.5 MHz
- Internal 16K ROM/I K SRAM
- Runs BASIC 800% *faster!*

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

#132

RFACT is small, the nonlinearity is high and vice versa as shown by cranking some sample numbers through the following calculation:

RFACT=	10	30	100
ERROR%	FNL	FNL	FNL
10	2.35	1.34	1.09
20	5.18	1.78	1.19
30	10.37	2.35	1.30

Next the proportional, derivative, and integral gains are calculated [note FNL's role) as necessary, depending on the controller type in effect (P, PI, PD, or PID). Similarly, a Switch statement builds the final output for a particular controller type from the component terms.

Some embellishments follow the raw calculation. The next few statements show integral "windup" (here called "saturation") prevention, but note that they have been commented out since antiwindup conflicts with the nonlinearity algorithm. The next two statements simply limit the output [fan speed) to empirically defined minimum and maximum values.

Finishing up, the current error (e) is made the previous error (ep) which will be used to calculate the derivative [change in error) on the next pass through the loop. Finally, the calculated output is returned to main() where the fan speed/PWM is set using the S-bit timer as previously described.

Listing I--The core of *the PIDDLE* control *code is the* PIDLOOP *routine.*

```
int pidloop(float *integ, int *ep, word y)      /* PID-loop calc */
                           /* integ integral part of controller */
                           /* ep preceding control error */
                           /* y process output */
/* this function uses global variables such as:
     ctyp type of controller (P, PI, PD, PID);
     Kp,Kd,Ki,U0 constants of controller;
     x  setpoint
     Umin,Umax   range of the output control signal */
{
 float output:          /* output control signal */
 int e:                 /* actual control error */
 float  err,            /* normalized 0 to 1 control error */
        FNL,            /* nonlinear gain factor */
        fract,          /* err/RFACT */
        prop,der;       /* prop and deriviative parts of regulator */

 e=x-y;                 /* calculation of actual error */

 /* RFACT sets the ctrl action nonlinearity by causing the factor
     FNL to double for every RFACT % incr in controller error */
 if ((err=(float)e/x)<0) err=-err;      /* normalized error calc */
 fract=err/RFACT;
 FNL=1.0+fract*(90.0+fract*(3045.0+fract*145675.0));

  /* output calculation */
 prop=Kp*e*FNL;               /* calculation of proportional part */
 if  ((ctyp==PD) || (ctyp==PID)) der=Kd/T*(e-*ep);
                              /* calculation of deriviative part */
 if  ((ctyp==PI)|| (ctyp==PID)) *integ=*integ+Ki*T*e/FNL;
                              /* calculation of integral part */
 switch  (ctyp){              /* calculation of output signal */
   case    P: output=prop+U0;            break;
   case  PD: output=prop+der+U0;         break:
   case  PI: output=prop+*integ+U0;      break:
   case  PID:  output=prop+der+*integ+U0;
              }

 /* when using nonlinear algorithm antiintegral saturation
    correction may cause improper action of controller when
    output>>Umax or output<<Umin */
 /*
if  ((ctyp==PI) || (ctyp==PID)){  antiintegral part sat correction
    if (output>Umax) *integ=*integ+Umax-output:
    else if (output<Umin) *integ=*integ+Umin-output; }  */

 if (output>Umax) output=Umax;        /* output limiter */
 else if (output<Umin) output=Umin;

 *ep=e;                 /* new error */
 return output:         /* return the object controlling magnitude */
}
```

TIME FOR A TUNE-UP

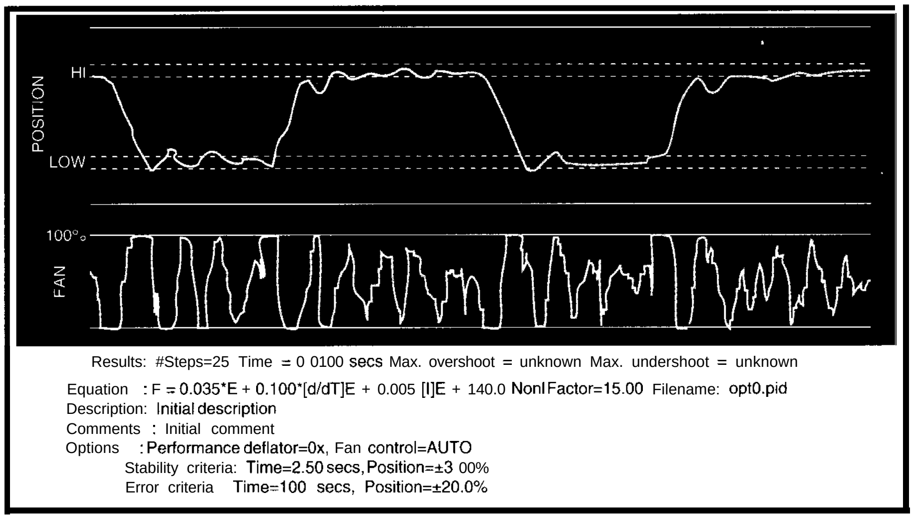
The "Dashboard," which runs on the PC (it's written in Turbo C) and communicates with the H8 via serial port, is the final piece of the PIDDLE puzzle and performs two key functions.

Via menu selection, the Dashboard allows all key parameters to be defined and downloaded to the H8 including the controller type (P, PD, PI, PID) and associated gains, setpoints, the previously described RFACT nonlinearity factor, maximum and minimum fan output, and stability criteria.

Once everything is set up, the game begins. Under control of the Dashboard, the H8, and PIDDLE software try to swing the ball back and forth between setpoints. On each pass through the PID loop, the H8 reports the state (i.e., ball position and fan speed) to the PC where it is displayed strip-chart style (Figure 5).

As you can see from the figure, PIDDLE does a great job. Notice how, underneath the smooth response of the ball, the fan is going through hoops. This highlights the PID-Pong challenge-imagine trying to achieve the same results twisting a knob.

Tuning the PID equations (i.e., choosing the controller type, gains, and RFACT) is a fairly ad hoc process of experimentation. A common strategy (the "Ultimate Method" described in Liptak and Venczel) is to choose a P-only control setup and increase the gain until the system



Results: #Steps=25 Time = 0 0100 secs Max. overshoot = unknown Max. undershoot = unknown
Equation : F = 0.035*E + 0.100*[d/dT]E + 0.005 [I]E + 140.0 NonlFactor=15.00 Filename: opt0.pid
Description: Initial description
Comments : Initial comment
Options : Performance deflator=0x, Fan control=AUTO
Stability criteria: Time=2.50 secs, Position=±3 00%
Error criteria Time=100 secs, Position=±20.0%

Figure 5—The Dashboard screen *printout* illustrates just how impossible if is for a mere human *to control the* fan *speed* quickly enough *to keep the* ball *position* stable.

oscillates (i.e., the ball bounces around, but never stabilizes on the setpoint). Then, back off the P gain and start fiddling with the other stuff. Derivative action and/or RFACT can be tweaked to reduce the step response over/undershoot at the expense of rise/fall time. Integral action deals with intrinsic variabilities such as the fact that the fan output is higher when it warms up, atmospheric conditions, or airflow restrictions (try placing your'finger over the top of the tube). In a sense, the P and D terms get the ball close to the setpoint quickly and the I term helps "nudge" it into place.

An intriguing possibility given the computing power at hand would be to make the system auto-tune itself. The PC could try various equations and measure the results. Tweaking could proceed algorithmically along the lines described in the previous paragraph or, if you're in no hurry, more or less randomly. Just let the thing crank for a few hours (days, weeks?) while you head for the beach and come home to a finely tuned setup.

Another idea would be to check out the fuzzy approach in which the PID calculations would be replaced with a series of "rules" along the lines of IF ERROR IS X AND BALL SPEED IS Y THEN SET FAN TO Z.

The good news is I'll put all this stuff on my list of things to do. The bad news-that list is real long and never seems to get any shorter so don't hold your breath. In the meantime, feel free to challenge the PID-Pong machine to a match of wits. ❑

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

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.

CONTACT

Hitachi America, Ltd.
Semiconductor & IC Division
2000 Sierra Point Pkwy.
Brisbane, CA 94005 18 19
(415) 589-8300
Fax: (415) 583-4207

Hanslik Software Laboratory
ul. Huculska 18
40-736 Katowice, Poland
011-48-21-524-261

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

TECHNICAL postcard DECK

If You Have The Product, We Have The Market

Circuit Cellar INK's

TECHNICAL POSTCARD DECK

•GREAT RATES & AD SIZE OPTION
•INTERESTED, RESPONSIVE READERS
•FREE READERSERVICE

FAX YOUR SPACE RESERVATION TODAY!
203-872-2204
Attn: Sue Hodge
Or call 203-875-2199
Deadfines: Space 9-20-94
Materials 1 O-l -94

EMBEDDED TECHNIQUES

John Dybowski

Fast Processors, Big Caps, and Ring Oscillators

Continuing his discussion of the ec.32 board, John finally spends some time with the DS80C320 itself, plus considers how to back up system components through a power outage and how to keep power use to a minimum.

Last month, I described some of the more conspicuous features of the DS80C320 microcontroller. This new offering from Dallas Semiconductor not only reaffirms the viability of 8-bit processors for new and demanding applications, but, by virtue of its sheer processing power, bridges the performance gap between 8- and 16-bit devices. In light of Dallas's past achievements, it should come as no surprise that the 80C320 is just the first of a new series of processors destined to give some of the bigger chips a run for their money.

Already, Dallas has announced its next generation silicon based on the 80C320 processing core. The DS87C520 contains 16 KB of EPROM, 1 KB of RAM, and operates all the way from DC to 33 MHz. Since high-speed operation implies a higher level of power consumption, this new and improved architecture provides some very interesting features designed specifically to reduce the electric bill. There's even a version that adds a built-in real-time clock to the 87C520: the DS87C530. Naturally, the inclusion of a built-in timepiece necessitates an external backup power source that is also used to back up the 1 KB of on-chip RAM.

Interestingly, the recommended backup power source is not the usual lithium cell that Dallas has become famous for. Instead, the preferred backup source is a battery-like, double-layer capacitor, known as a *supercap.*

But, I'm getting ahead of myself here. Having barely scratched the surface of the 80C320, I'm telling you about silicon that's just now becoming available. This can be dangerous stuff. You've got to be careful since working with even the second or third cut at a new processor can prove to be more than enough to satisfy your need for adventure. Being the first one in line

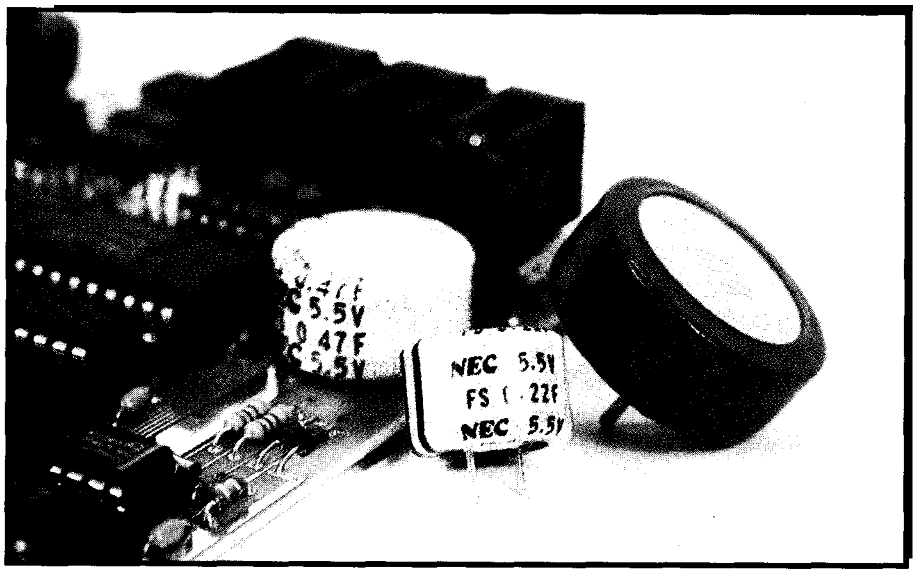


Photo 1-Supercaps are starting to appear in different shapes and sizes. While their appearance might be confused *with that of an ordinary battery, they are rechargeable and don't exhibit the same memory effects as shown by typical NiCds.*

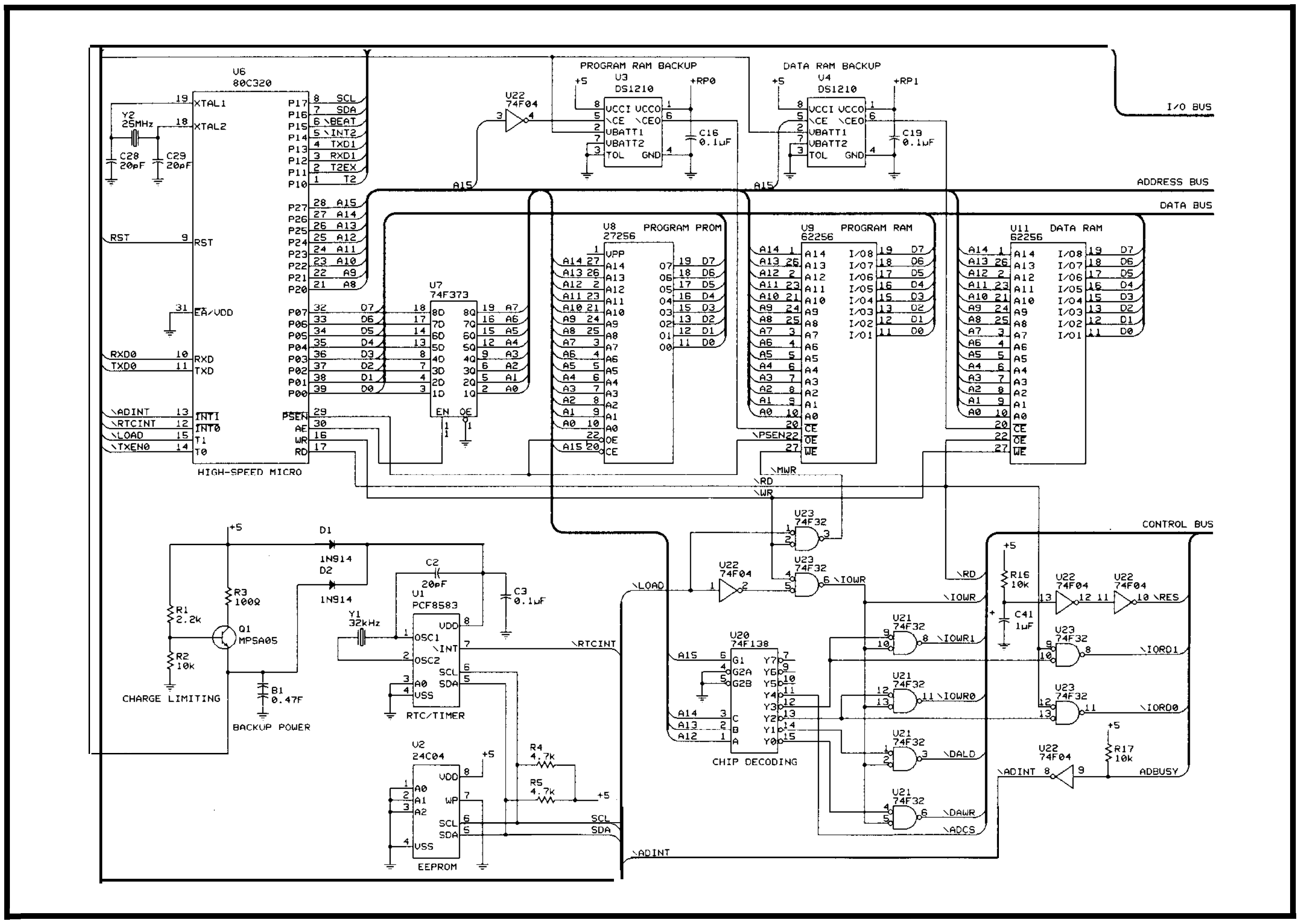


Figure I--The ec.32 *is based on the Dallas Semiconductor DS80C320 processor and includes everything needed to set up an embedded data collection system.*

can be downright reckless. Let me get back to the ec.32 embedded computer. Then, I'll tell you about supercaps and a little more about the 80C320.

THE ec.32

The 80C320, unlike the newer Dallas family members (with on-chip EPROM), requires external program memory to operate. Most applications also need external data memory since the 80C320 comes with only the usual 256 bytes of internal data RAM. I personally like an embedded computer that supports downloading executable programs directly into the target. EPROM emulators work okay, but having this capability built into the computer allows you to use this feature beyond the design phase and extend it into your end system.

Providing a built-in download capability does imply that special hardware considerations be addressed early in the design phase, but the overall system impact can be kept minimal. Coupling such hardware with a resident kernel and a PC-hosted debugger results in a potent, integrated development vehicle. The addition of a PC-based simulator along with a low-cost cross-assembler and C cross-compiler rounds out the system. This is the ec.32 embedded computer.

All things considered, the basic 803 1 architecture offers a fairly expansive addressing map for an 8-bit processor. This address map is also what tends to drive compiler developers nuts. With the program memory, internal data memory, external data memory, and bit addressable memory (not to mention the SFRs} that all overlap, it's a wonder these code generators don't choke.

The areas internal to the chip are, naturally, fixed. The memory regions that we can exercise control over are the external data and program areas. In an effort to keep things simple, the ec.32 partitions this program/data area into four 32-KB blocks, where program PROM and data RAM begin at 0 and program RAM is located at 8000h. The upper half of the data area at 8000h is allocated to the system's memory-mapped peripherals. This sparsely populated peripheral I/O area is, obviously, loosely decoded.

Although the program and data segments physically overlap, they are kept separate by independent read strobes: \PSEN for program memory and \RD for data memory. Through special gating of the \WR strobe, program RAM can be written to. During normal operation, program RAM is not writable, but this capability can be enabled by pulling \LOAD (P3.5) low via firmware. Asserting this pin has the effect of degating the \WR strobe to the peripheral section and instead routes it to the program RAM. This enables useful functions such as loading of executable programs and setting breakpoints.

Since the hardware, resident kernel, and PC debugger are closely coupled, I could have saved the \LOAD port pin and instead controlled this write-enable gating directly from the PC using a modem control line. I elected not to do so since I wanted to retain this powerful capability for application programs that didn't make use of the system's built-in debugging capabilities. Realize, however, that the resident kernel is designed for efficiency since it uses no RAM and consumes less than 2 KB of code space-it could ride along with just about any application program.

With 32 KB of program PROM you can burn a lot of functionality into the PROM and still be able to download sizable programs and functions into the 32-KB program RAM area. This downloadable program RAM can hold transient programs for performing diagnostic or analyses functions, replacements for major application routines, complete application programs, or big lookup tables.

The basic bus arrangement of the ec.32 is shown in Figure 1. The 80C320 connects, in the usual manner, via a transparent address latch to the PROMs, RAMs, and peripheral bus members. Chip-select and peripheral-support gating are implemented conventionally. The only unusual thing here is that FAST logic is used exclusively for the discrete logic. Picking up speed in the glue logic, as I explained last month, results in the ability to run with slower PROM, RAM, and peripheral chips.

The configuration shown requires a 90-ns program PROM and a 70-ns program RAM. The data RAM must be a 70-ns part if full-speed operation is required. With one stretch cycle, a common 150-ns part is adequate. Unless you really need the extra performance, it might make sense to leave the data access rate set to the default one stretch cycle. As I'll demonstrate next month, some of the system peripherals need that stretch cycle to remain within specifications.

Also shown in Figure 1 is the low-throughput-serial peripheral bus. This subbus supports a number of peripheral functions that don't need to operate at the full bus bandwidth. For this serial bus, the familiar two-wire I^2C is used. Local I^2C support is provided for 5 **12** bytes of E^2PROM using a 24C04. The PCF8583 RTC/timer handles a number of time-keeping tasks for the system and, additionally, provides 256 bytes of nonvolatile RAM. Contained within the PCF8583 is a real-time clock/calendar and a fully programmable interval timer that is wired as an interrupt source. The I^2C is carried through to a four-pin modular RJ- 11 connector and can be used to attach a companion LCD/keypad module or additional digital or analog peripherals.

THE TROUBLE WITH BATTERIES

With program and data RAM and an RTC, it's obvious that some form of backup power must be provided to make these functions nonvolatile. Often a lithium cell is selected since many of the more popular nonvolatile controllers are specifically designed to work with lithium characteristics. However, a nonrechargeable cell will

C COMPILER FOR PIC CONTROLLERS

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

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

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

PCB compiler	$99 (all 5x chips)
PCM compiler	$99 ('84, '71, '84 chips)
Pre-paid shipping	$5
COD shipping	$10

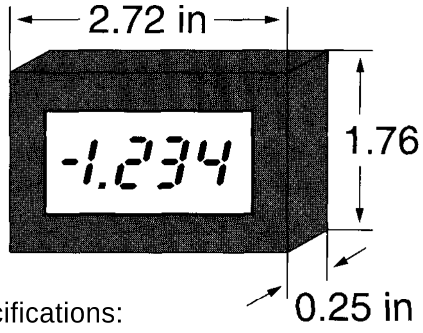
CCS, PO Box 11191, Milwaukee WI 53211
414-78 l-2794 x30

#198

3½-DIGIT LCD PANEL METER

-Available now at an unheard of price of **$15** plus s&h

New! Not surplus!



Specifications:

Maximum input: ±199.9mV
additional ranges provided through external resistor dividers
Display: 3½-digit LCD, 0.5 in. figure height, jumper-selectable decimal point
Conversion: Dual slope conversion, 2-3 readings per sec.
Input Impedance: >100M ohm
Power: 9-12 VDC @ 1 mA DC

Circuit Cellar, Inc.
4 Park Street, Suite 12, Vernon, CT 06066
Tel: (203) 875-2751 Fax: (203) 872-2204

eventually have to be replaced. Even with rechargeable types, there is a limited service life. This is a shame since you can bet your controller will end up in some totally inaccessible area and you've undoubtedly designed it to otherwise last forever. Given the available choices, maybe you'll find it surprising that I suggest not using a battery at all.

When supporting low-drain CMOS loads, good results can be achieved using a capacitor as a backup power source. However, not just any capacitor can be used for this purpose. Electric double-layer capacitors, more popularly known as ***supercapacitors,*** are specifically designed as a backup power source. This supercap technology offers low leakage and a volumetric efficiency 10–50 times that of the most compact, aluminum-electrolytic capacitor. This means you can squeeze 1 F of capacitance into less than a cubic inch.

Supercaps offer several advantages over conventional batteries. They don't need periodic replacement. The charge rate is not at all critical and can range from microamps to amps. They have no polarity and can't be damaged by reverse connection. They are inherently safe and will not leak or explode under temperature extremes since they contain very low electrolyte levels. Typically, they are rated to operate from -40 to +85°C. Charging and discharging a rechargeable battery causes a chemical reaction that typically cannot be repeated more than several hundred times. In contrast, charging or discharging supercaps is a physical, not chemical, phenomenon that does not cause damage.

The backup circuit shown in Figure 1 uses an NEC FS0H474Z 0.47-F supercap that feeds two DS1210 nonvolatile controllers that, in turn, supply power (and provide protection) to the program and data RAM. A tap is also taken off of the supercap through a 1N914 diode that is used to deliver backup power to the PCF8583 RTC/timer. A separate 1N914 isolates the main 5 V and feeds the RTC when the system is active.

Normally, in such a backup scheme, you'd want to use better diodes than the 1N914s that I selected. In this case, however, little advantage would be gained since the PCF8583 can operate all the way down to 1 V and the signal pins can exceed VDD by 1 V. Having made the rather cavalier statement that there's nothing to charging a supercap, you may well wonder why I've placed a handful of parts in my charging path. Let me explain.

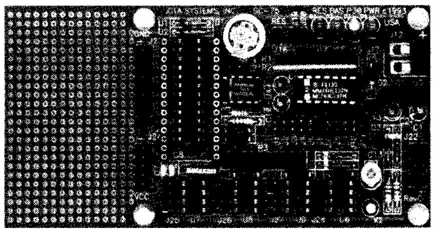
In principle, there's no reason why you couldn't hook your supercap directly to the system's 5-V rail. However, in such an arrangement, the inrush current would be limited primarily by the supercap's ESR. The specification for the 0.47-F part I am using shows this parameter to be about 13 Ω. This justifies the 100-R series-limiting resistor (R3) that limits the power supply burden during the initial current inrush.

This explanation still leaves several other components in the charging circuit unaccounted for. The

---

Speed Your Development Process

By Using Our Controllers!



We offer an array of controller boards and software tools for the 8051 and 87C751 families of microcontrollers. Complete packages are available to help you develop your projects. We also have a selection of add-on peripherals such as LCD and keypad interfaces.

Features:

- Breadboard area
- Flexible I/O arrangement
- Powerful controller BASIC for the 87C752
- Simulators

Ph: (702) ***831-6302*** • *Fax: (702) 831-4629*

Iota Systems, Inc.

POB 8987 • Incline Village, NV 89452-8987

#133

C COMPILERS
CROSS ASSEMBLERS
DEBUGGERS

68HC08	8051/52
6809	8080/8085
68HC11	8086/186
68HC16	8096/196

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

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

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

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

SPECIAL THIS ISSUE!! *Emily52 Simulator/Debugger* ***Reg. $49.95,*** only ***$25.00! with purchase of 8051/52 Developers Kit (above). Offer valid until October 31, 1994***

*Call or **write for** our free catalog **of development tools.***

DDS

Dunfield Development Systems
P.O. Box 31044 Nepean, Ont. K2B 8S8
CANADA
Tel/BBS: 613-256-5820 Fax: 613-256-5821

emitter-follower composed of RI, R2, and Q1 limits the maximum voltage to which the supercap can charge. Although limiting the charging voltage diminishes the ultimate backup time somewhat, it's necessary to ensure proper operation of the DS1210 nonvolatizers.

Briefly, the DS1210 protects its associated RAM by providing power from either the system 5 V or from the backup power source. Additionally, chip select is routed through the DS1210 and is held inactive when operating in backup mode, thereby protecting the RAM.

It seems the DS1210 doesn't actually monitor the VCC pin for the presence of 5 V but, instead, looks at the internal voltage that results from mixing the VCC voltage with the voltage from the backup battery. If this internal voltage is above the selected threshold level, the part won't go into backup mode when VCC goes away. Obviously, this renders the part useless and has dire consequences for the associated RAM. This threshold level is dependent on how you strap the DS1210's TOL pin. But, regardless of the setting, you won't have any problems if the backup power is kept somewhere under 4 V.

Since a supercap is, after all, still a capacitor, you might conclude that given enough charging current, a full charge could be attained in a matter of seconds or minutes. Although this may be true in a sense, when a supercap is charged, it actually goes through several distinct phases.

A closer examination reveals that there are actually several aspects to the charging current that flows into a supercap through these various phases:

*recoverable charging current that can be reclaimed when the capacitor delivers backup power
*absorption current that cannot be recovered on discharge
*leakage current

It is only during the last stage that the supercap exhibits minimum leakage current and therefore attains a low-level of self-discharge (l-2 µA).

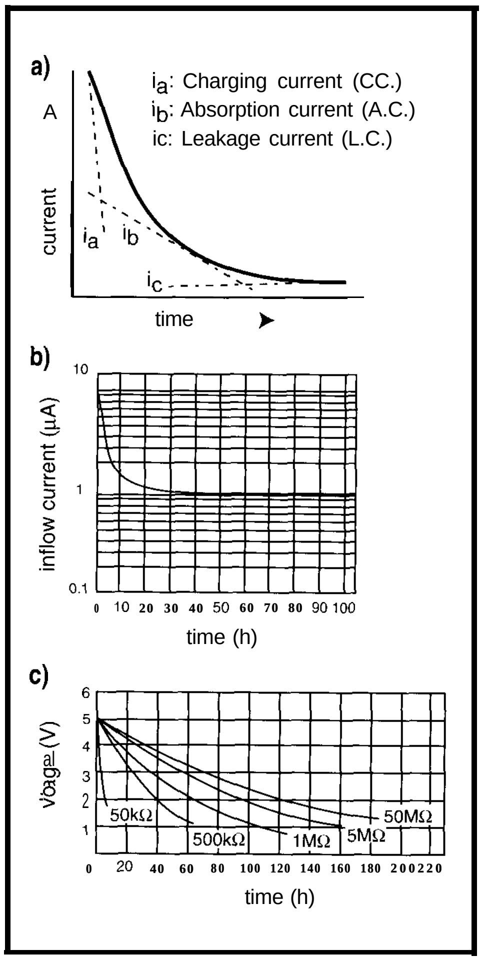


Figure 2-A *supercap can be characterized by a) various current components flowing info the supercap, b) the expected inflow current through a 100-ohm source impedance into 0.47 F plotted against time, and c) several representative discharge curves.*

Unfortunately, this only occurs after 10 or more hours of charging.

So you shouldn't knock yourself out trying to reduce your circuit's current consumption to inordinate levels since, after a point, the capacitor's own leakage current will predominate and may ultimately swamp any imagined power savings you can attain.

By now, I'm sure there's no doubt that a supercap is more than just a souped up capacitor. It should be evident it has electrical properties different from other capacitors. Because of these differences, projections of backup time based on familiar calculations may be misleading. First of all, the capacity or the amount of energy available will be less at high than at low discharge rates. And, for extremely light loads, the supercap's internal leakage current will dominate and be a greater loss than the circuit being backed up.

To further complicate matters, the capacitor's leakage current decreases drastically as the voltage decreases. Additionally, a typical CMOS load cannot be characterized as a constant current or a constant impedance and, as a result, also changes as the backup voltage decays. These factors complicate an accurate prediction of backup time for a given set of parameters. Best results can be attained by studying the manufacturer's specifications and discharge curves along with some breadboarding and empirical observations.

Figure 2 illustrates the various current components flowing into a supercap, the expected inflow current through a 100-n source impedance into 0.47 F plotted against time, and several representative discharge curves.

RINGS OSCILLATORS AND CRYSTALS

The 80C320, like all CMOS circuits, consumes more power at higher operating frequencies than when running slowly. In many system configurations, maximum performance is the overriding concern and power consumption may not even be an issue. However, there are circumstances under which we must stipulate both a high level of performance and low power consumption. These conflicting criteria represent a certain class of portable and battery-operated instrumentation.

The saving grace to this seeming paradox is that, although low power and high performance are mutually exclusive, you needn't deliver both simultaneously. A popular hack is to organize the system to remain idle for extended periods and to operate in high-speed bursts only in response to external events. This tactic drives the average power consumption way down yet leaves the door open to full-speed processing when necessary. Under such circumstances, it's beneficial to use one of the processor's built-in low-power operating modes while waiting for something significant to happen.

Like the 80C3 1, the 80C320 supports two power-saving modes of operation. IDLE mode suspends all processing operations by holding the program counter in a static state. This saves a considerable amount of power particularly when operating with external memory since the bus is kept inactive. While using about half the power of a fully operational system, IDLE mode keeps all clocks active. The serial port, timers, watchdog, and power monitor are all kept functional; the processor can instantaneously respond to any interrupt condition.

STOP mode provides the lowest power consumption by stopping all on-chip clocks. No processing is possible, and all timers and serial communications are shut down. This state results in current consumption of l-2 µA.

STOP mode can, of course, be exited via a reset. This reset can be externally generated or be the consequence of a power fail reset. Note that the power fail reset and power fail interrupt require the use of the 80C320's internal band gap used to monitor VCC. To conserve power, the 80C320's band gap is turned off by default when STOP mode is entered.

Although turning off the band gap results in significant power savings, the processor could potentially be thrown out of control should a power dip or brownout occur since a clean reset would not be generated. This is not the case with a full power failure since normal band-gap operation is restored on a power-on reset. In any case, should a brownout be a possibility, the band gap should be turned on via firmware prior to entering STOP mode. In some cases, this may be undesirable since it increases the current consumption to about 100 µA.

A second, more useful means of exiting STOP mode exists which involves the use of an interrupt. One restriction is that an internally generated interrupt can't be used since there is nothing going on inside the processor when it is STOPed. An external interrupt can be used without restriction, or a power fail interrupt can be used provided the band gap is enabled. Although keeping the processor in the STOP mode most of the time realizes maximum power savings, it's important to restrict full-speed operation to the absolute minimum amount of time.

In a traditional implementation, a significant amount of power can be wasted waiting for the oscillator to come up to speed and stabilize before meaningful operations can be performed. The crystal startup time could exceed the actual processing time if only a short operation need be performed. This scenario portrays the default condition for the 80C320 since the crystal oscillator must delay 65,536 clocks before full-speed execution can begin. Needless to say, this can really put a drag on performance. The 80C320 addresses this problem with an internal ring oscillator that can start instantaneously.

The extended interrupt SFR EXIF (91h) contains the bit EXIF. 1 that, when set by firmware, enables the 80C320 to use its ring oscillator to come out of STOP mode without waiting for the crystal oscillator to start. The ring oscillator provides an instantaneous start up, but is not accurate. Generally, this isn't a problem since the firmware usually performs a short task and returns to STOP mode. The crystal oscillator is started up when STOP mode is exited, but is not available until 65,536 clocks are counted. If an accurate timebase is required, the code has no choice but to wait for this startup delay to expire before beginning its time-critical operations.

EXIF.2 can be interrogated to determine from which clock source the processor is operating. This bit is set when the ring oscillator is used and is cleared by hardware when the crystal oscillator kicks in. If it turns out that you do need the accuracy of a crystal timebase, at least you can get your preliminary setup out of the way and be ready to roll by the time the crystal comes on-line.

Although the 80C320 takes fundamental 80C31 power-saving capabilities to new extremes, you can expect further refinement in upcoming processors in this family. Already, the 87C520 and 87C530 have defined new power-management modes: PMM1 and PMM2. Normally, the CPU performs an instruction cycle every 4 oscillator clocks. In PMM1, this time is extended to 64 clocks. With PMM2, this goes out to 1024 clocks. You can operate in PMM1, keeping the processor alive, and still consume less power than in IDLE mode. And, these newer processors let you totally shut down the crystal amplifier and operate exclusively from the internal ring oscillator. Even when running off the ring oscillator, you can realize the benefits of PMM1's 64-clock and PMM2's 1024-clock prescalers.

AND THE WINNER IS...

Next month I'll conclude my description of the ec.32 with a discussion of the analog and digital I/O, power supply, and other miscellaneous system elements. I've spent considerable time working with the ec.32, yet I still find myself at times surprised at just how quickly it runs. Still expecting the status quo, I guess. ❑

John Dybowski is an engineer involved in the design and manufacture of embedded controllers and communications equipment with a special focus on portable and battery-operated instruments. He is also owner of Mid-Tech Computing Devices. John may be reached at (203) 684-2442 or at john.dybowski@circellar.com.

SOURCES

For elements of this project, contact:

Mid-Tech Computing Devices
P.O. Box 218
Stafford Springs, CT 06075-0218
(203) 684-2442

Individual chips are available from

Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 27613
Phone/fax: (9 19) 676-4525

I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) 871-1988—Four incoming lines
Internet E-mail: sysop@circellar.com

We have quite the gamut of topics this month. First, we consider what's necessary for doing a closed-loop control scheme and how to trade off digital and analog sections. Next, we talk about whether it's best to condition low-level analog signals right on a plug-in board or outside the main computer.

Use an existing spec or come up with something tailored to the application? That's the question we debate in the next thread. Finally, back to ESD issues and a quest for reference books that spell it out in terms mere mortals can understand.

Control loop feedback

Msg#: 8154
From: JOHN DAVID COOK To: ALL USERS

Can anyone recommend a good simple book to explain control loops? Basically I'm looking for "Feedback Loops for Dummies" or something that will explain the dynamics of the darn things intuitively so I can read it.

I ask for my robot which uses two independently driven wheels tied together in a software control loop so that it goes straight when told to. The problem is that the error between the two wheels tends to "run away" and send the robot spinning, literally as the loop overcorrects and both wheels spin full in opposite directions.

Msg#: 8168
From: BEN MEHLMAN To: JOHN DAVID COOK

I can't recommend a book but I have two suggestions.

First, vary the amount of correction in proportion to the error; that is, if the robot is only slightly off course, speed up the slow wheel SLIGHTLY and give it time to catch up. What is happening is you haven't taken the inertia of the robot into account. You need to give the robot time to change course.

Second, limit how much correction you apply. Since you probably have a good idea how much slippage there's going to be, set realistic limits on the maximum correction and the maximum time it can be applied. This will prevent it from losing control.

Msg#: 8419
From: PAUL SHUBEL To: JOHN DAVID COOK

"Simple" control algorithms are not easy to come by. Most of the articles I see are only one notch below *Transactions of the IEEE.* Their use by anything but "control engineers" is doubtful. I always hark back to the early years of microprocessors (sigh] when magazines were filled with article after article on "basic" techniques with micros.

Msg#: 8603
From: JOHN DAVID COOK To: BEN MEHLMAN

There are a couple things I didn't mention before: when velocity is set to zero, the 'bot is very unstable; a slight bump can cause it to spin. Also, if the velocity of one wheel drops to zero while turning, it will spin. I theorize that the ratio of error over velocity is going to infinity as velocity drops to zero.

Is this a sure sign of positive feedback?

Msg#: 8673
From: JAMES MEYER To: JOHN DAVID COOK

Not necessarily. (How's that for an answer?)

If I were writing the code, I'd start with some simple, testable, routines that could be combined later into a more complete program.

Some fundamental routines to start with include:

- Measuring the distance that each wheel turns.
  Revolutions * Circumference of wheel.
  *Speed of each wheel. Distance / time unit.

Then use the speed as feedback in a loop so that you can command a particular speed and have the software maintain that speed in the face of variations in load on the motor.

Note that you can do this early in development without a complete 'bot-just a motor, tachometer, computer-controlled current source, and a method of inputting the speed command. Note carefully the term "current source." Motor speed control is *much* easier if you control the applied current rather than the voltage.

Until you can do something like this, trying to debug a more complicated system will probably be impossible. The idea is to "divide and conquer." Take a complex task and split it up into smaller tasks. Each so simple that they *have* to work. Then build on success. You'll always know at what point something new goes wrong, so fixing it should be easier.

CONNECTIME

Msg#: 8677
From: GEORGE NOVACEK To: JOHN DAVID COOK

There are numerous good books on the market (the classic here is "Automatic Control Systems" by Benjamin Kuo) dealing with closed-loop systems. Some books do not go beyond the PID controller, others go further into Kalman filters, predictive control and fuzzy controllers. The bad news is that unless you enjoy calculus, transforms, and similar advanced math disciplines, you will not enjoy these books. Still, they will explain to you the theory behind it, but will not tell you how to design it.

Math is no longer such a problem thanks to the PC, as there are numerous programs out there which will let you calculate as well as simulate closed control loops. They range from freeware through a-couple-thousand-dollar packages such as Mathlab to $20,000 wonders such as Easy 5. The problem is it is not the controller but the actuators and the mechanical parts which will eventually defeat you. Unless you can get a reasonably accurate transfer function of these mechanical parts (in my years in design of embedded controllers I rarely saw them), your simulations will only confirm the old computer adage "garbage in, garbage out." Therefore, it is more than likely you will end up tweaking. I hasten to defend analytical methods: High performance in closed-loop controllers (such as head positioning in hard drives] would never be achievable by tweaking. But keep in mind these people manufacture millions of those at close tolerances and, therefore, can develop proper data. You can't.

Speaking out of experience: if there is no special reason to close the loop digitally, don't! It is hard to beat a $1 op-amp where no special adjustments calling for digital control are needed. As a rule, I use a micro to generate the position command to the loop, then in many cases go to a DAC and close the loop in analog. Even many dynamic adjustments of the loop function can be done in the analog loop by using digital gain control, multiplying DACs, and so forth. If you need a PID and don't want to play with different RC values, you can be inventive and do the proportional term with an op-amp while generating the integrating and derivative terms in software. There are many ways to skin a cat.

The main problem you are going to run into with a real-time controller is the lack of time. You suggested that you are controlling a robot. I venture to guess that, if it is a motion of the robot you are controlling, the critical frequency of your system will be in the 10-Hz area. Unless you can acquire all the data within no more than a IO-ms period (100-Hz sampling rate), you are asking for aliasing problems. But things get even worse. You cannot just use each and every sample you acquire for control. Digitization noise will kill you. You have to filter the signal: averaging is most common, convolution can give you almost an order of magnitude better S/N ratio. Depending on the system, you will need probably 5 to 9 samples. This takes not only processing time, but inherently introduces lag into the system, which will not be very beneficial to its stability.

The lag that the frequency limiting of the feedback is introducing at this rate can also cause deterioration of your phase margin and lead to oscillations. Furthermore, you must update your loop calculation at the same rate (every 10 ms), or you will have a *very* hard time keeping the loop stable due to the processing lag. If not accurately controlled, you can actually go into positive feedback. The integrating term will need some limiting (it can saturate the controller; many systems enable the I term only within a small position error range). You must also make sure there is no roll-over of the integration. A 68HC11 will be hard pressed to keep up at this rate, even if there is nothing else for it to do, especially if you code in C. So, if you do not have to use Kalman filters and have a DSP on board, don't mess around with digital.

Because you probably will not have transfer functions to do any meaningful calculations, you will end up having to tweak the system, analog or digital. Disable both I and D terms and get the proportional term working correctly. Then add I and tweak it. Last, add the derivative term. The *Circuit Cellar INK* had a good article several years ago with a very detailed procedure how to do it.

Analog signal conditioning

Msg#: 8540
From: **DARRICK** WEST To: ALL USERS

Is it good design practice to do analog signal conditioning inside the PC? As an example, let's just say I would like to boost the gain of an incoming signal by 100 or 1000. Would it be better to condition the signal outside the PC's environment at that level of gain? And are the ±12-VDC power supply lines on the bus clean enough to use, or is it better to regulate and filter this it down to a lower voltage.

Msg#: 8674
From: GEORGE NOVACEK To: **DARRICK** WEST

It depends. From the packaging, construction, and convenience standpoint, it is much easier to design a plug-in card, stick it into an expansion slot inside the PC, and forget about it. But there are caveats!

One, the power inside the PC is limited and is not very clean. There are methods to get around it; in my view the best bet is in putting an independent regulator on board. Getting clean, 12-V signal from the digital 5-V supply is no longer a big deal.

Two, the EM1 environment is not exactly very clean either, be it through conducted (mainly ground bounce) as well as radiated emissions.

So it mainly depends on the impedance of the circuits you are going to design, especially at the low signal level and the bandwidth you will be working with. You will have to keep the PCB traces as short as possible; use SMT components if you can and in the end you may have to put a metal can around the analog front end. If you have never used I2 or more bit digitizers, be prepared for a little blood bath. PCB layout and proper grounding is something you will have to learn by experimenting. I would strongly recommend that you use a four-layer board with proper ground and power planes. Keep in mind basic rules such as the 5-V digital power traces must not run on top of the analog ground. You are dealing with fast-rise-time pulses, which generate very high frequency spectrum and even minute capacitances between PCB layers can cause unwanted coupling.

It all depends on your requirements. Eight-bit resolution is easy. Ten bits will get slightly demanding. Anything over ten is unforgiving. I am talking about digitization, but it applies equally to just pure amplification. If you have a 5-mV signal which you are jacking up to 5-V range and then digitizing to eight bits, you still have to look at the least-significant bit level divided by the gain to establish the noise level your system will accept. If you do not have experience with low-level circuits, it might be easier for you to put the module outside the PC, with its very own power supply. Then, at least, you will know that the problem is within your module, not some radiated coupling through the air from God knows where.

Msg#: 8801
From: PETER HAND To: **DARRICK** WEST

It's pretty noisy in there, but you can do it if you're careful about shielding and grounding. A lot depends on what's in the next slot, too. You may have problems if it's a video adapter.

The ±12-VDC power supply lines on the bus are usually horrible, and it's easier to start with a reasonably clean supply than try to filter a dirty one. Consider using a DC/DC converter to generate ±15 V from the 5-V supply; so much the better if the output ground is isolated. Working at ±15 V gives you a bit more headroom to work at higher voltages.

I presume you're about to tackle a 12-bit ADC project. Be warned-it will take your knowledge and patience to the limits and beyond. I'd use a multilayer PCB with a continuous ground plane, and not allow any digital tracks—including the 5-V supply-anywhere near the front end.

DMX-512 or something else?

Msg#: 6230
From: VIC MANZO To: ALL USERS

I'm Looking for a microprocessor-controlled UART that does serial transmitting and receiving in xl6 mode at 250 kbps. If possible, a dual version. I've tried all the Philips and National Semiconductor products but have not come up with acceptable availability times on then. If anyone has used or knows of one, please let me know.

Msg#: 6280
From: BEN MEHLMAN To: VIC MANZO

The National PC16550 can do 250 kbps. This chip might be in short supply now due to strong demand in the PC market, but at least you know it will be in production for a while! It uses a xl6 clock, so a 4- or 8-MHz clock will get you 250 kbps with a divisor of 2 or 4.

I assume you are generating DMX-5 12 for lighting control? I am specifically interested in lighting control projects of this nature. I have constructed a prototype circuit using the 16550 to generate Colortran protocol (DMX at a lower speed). Unfortunately, I am having a channel jitter problem with it which I originally thought was due to an unstable mark-after-break period. After a little troubleshooting, I believe the trouble is in my UART-to-computer (PC parallel port] interface timing, so the chip itself has not been ruled out as suitable for the task.

I'm also trying to get info on the Siemens 80?? 166 microcontroller and ilk. This microcontroller has two internal UARTs capable of greater than 250 kbps, and sufficient processing speed and internal RAM to offload the entire communications task to the one chip. Unfortunately, I can't find Siemens to get the info on it! Does anyone have their USA phone number? I know they have an evaluation kit which they give away, lend, or sell real cheap.. .

Msg#: 8598
From: VIC MANZO To: BEN MEHLMAN

As a matter of fact, I am going to use it for DMX-512 as well as other lighting protocols including Colortran. How did you become interested in lighting control? I work for a company called BASH lighting in New Jersey. Maybe you have heard of us! I also own my own company called Logical Lighting Interface Inc. I don't think you've heard of that one. Our company designs and manufactures interfaces for lighting. We are starting to get into designing specialty items such as radio-controlled dimmers. If you need any help with you project, let me know.

I think I'm going to have to design my circuitry for the National 16C550 or the 16C552. Because it is used in PCs, I think I would have a good chance of it being more available than others. I am also looking into the Philips version of the same UART. They apparently just redesigned their part and it has become available.

Thanks for your suggestion.

If you are considering using a micro for you project, try the Dallas 80C320 out. It has two UARTs and a bunch of other neat little toys.

Msg#: 8761
From: BEN MEHLMAN To: **VIC MANZO**

What you're doing is interesting to me. I am mostly into software now, and I've been working on lighting console software for an embedded PC-type board. It's coming along well although the concepts have shifted rather dramatically since I started! I'm using a test system now which consists of a DMX/Colortran protocol adapter and a small manual console with just the grand master, one split crossfader, and a few of the more important buttons on it. Both of these plug into my PC or laptop. I have been recently working to license this software to a board manufacturer. Of course, as usual, finishing it would be a good idea, too. ;-)

Msg#: 8681
From: PETE CHOMAK To: **VIC MANZO**

I just happened to read this message, and it looked like you might have some info I need. I built a large camera crane/boom with a remote control pan-tilt head a while back (2-week, 24 hour-per-day rush project), and now I am working on a proper servo system for the head (I used a variable-voltage drive and center-off momentary toggles initially). I plan on using a PIC for the control, with the position commands received from a PC-based controller via a serial link (preferably on XLR audio cable). It occurred to me that I should give the controller an address so I can build other similar controllers and use them on the same bus. Then it occurred to me that that is what DMX-512 does. I don't want to reinvent the wheel, so do you have, or know where I can get the full DMX-5 12 specs?

Msg#: 8760
From: BEN MEHLMAN To: PETE CHOMAK

The DMX spec can be purchased from the USITT (U.S. Institute for Theatrical Technology) for less than $20. The spec is simple, so I'll explain it.

The physical interface is a standard RS-422 line. The connector type is specified as a 5-pin XLR-type connector, with an optional second pair of wires for two-way communications (few DMX devices use this, and no software spec is written for reverse communications).

The serial format is standard 8-bit asynchronous, with 1 start and 2 stop bits running at 250 kbps. The entire set of 8-bit levels for up to 5 12 dimmers are sent in a packet, and packets are sent repeatedly, generally no less than 10x per second, usually 20x or more. The spec allows a much lower repeat rate but, no one uses it.

Now for the packet itself: Send a "break" for three byte times, followed by a I-bit "mark after break" followed by a 1 -byte DMX start code which is always zero. DMX receivers are supposed to ignore any packet with a nonzero start code. The start code is immediately followed by from 1 to 5 12 bytes of data, with one byte per channel to be controlled.

Note that there is no error correction. Dimmers don't respond fast enough to need it (much). Your application is different. You might want to defer responding to any value that deviates from the last value received by more than some amount.

Even though it's kind of a hokey spec, you should seriously consider using it since it means that hundreds of computerized lighting consoles will be able to control your unit. Some of them could have it doing some really wild stuff.

Msg#: 8822
From: PETE CHOMAK To: BEN MEHLMAN

Thanks, it sounds fairly simple, the only difference is that I was planning on 16-bit position values, but 8 bits may do, or I could use two addresses per axis. I would like to get the "official spec" if you happen to find the address, but this should get me going.

Msg#: 8829
From: BEN MEHLMAN To: PETE CHOMAK

Well, I can't find the address. But here's the phone number for the USITT: (212) 924-9088. They are very nice and you can order the document over the phone with a credit card. They have other documents which are interesting, too.

Regarding 16-bit values, this has become a problem in lighting as well. Eight bits are enough to control the brightness of a lamp, but not enough to control the position of moving lights as smoothly or accurately as needed. But there are a lot of manufacturers doing it anyway, either using "smoothing" algorithms to generate intermediate data points for less jumpy motion, or using two channels for 16 bits of resolution.

The two-channel method has been gaining some popularity, and there are some consoles, built with features to support moving-light programming, that give the user 16 bits of control range on one control, sending the output on two DMX channels. There has also been some activity lately of trying to nail down a formal spec for 16-bit control. I am not up on the latest status of this.

Anyway, depending on how your camera is used, I have a suggestion for you. Allow one DMX channel to control the upper 8 bits of your motor position, giving that control a "coarse" control of the entire pan range. Then, allow a second channel to control an 8-bit range of values which may or may not overlap the range of the first channel. In other words, allow the second channel to modulate the first channel, and allow the full-scale value of that modulation to be settable by the user.

So if the user, for example, is going to have all their detail work being done in a 45" window, they set the coarse channel to one edge of the range of interest, and use the fine channel to move within the range. You may find that you don't need a true 16 bits of resolution after all. Use 16 bits of control to generate perhaps only 10 or 12 bits of motor position information.

Msg#: 8955
From: PETE CHOMAK To: BEN MEHLMAN

Interesting. I figured on 16 bits just because it is a neat number, but 10 bits is probably enough, with better than 0.5" resolution for a 360" range. The smoothness and update rate are particularly critical for camera positioning to allow for acceleration and deceleration of a pan or tilt and for fine framing, especially with a tight zoom. Thanks!

Msg#:10702
From: VIC **MANZO** To: PETE CHOMAK

DMX-512 runs at 250 kbps. As you probably know, emulating asynchronous communications at this rate with software using a super fast micro is hard. Using a PIC would probably be impossible.

I would suggest using an RS-485 data link but using a synchronous style link. You shouldn't think of creating a protocol as "reinventing the wheel." Protocols should be created for a particular application. Creating your own protocol also allows YOU to set the standard for your controlling equipment. You don't really need Joe Electronic making controllers for your equipment.

Getting back to the point, I would also suggest you use 16-bit pan and tilt values if the system is to be digital. I have found that using 8 bits doesn't give the resolution or the repeatability that is required, especially for camera applications, Bidirectional links also have great advantages; you should look into at least providing the hardware for later software revisions.

Msg#: 10777
From: PETE CHOMAK To: VIC **MANZO**

You are probably right. I was thinking that by using DMX-5 12, my controller would be able to also control DMX-5 12 lighting gear. But I can probably build a translator if I need to do that. Bidirectional data is an interesting idea, but I can't think of any uses for it yet. Thanks.

ESD design references

Msg#: 7611
From: ROBERT MCILVAINE To: ALL USERS

Can anybody recommend a source for info on ESD protection design that might be obtained on the Internet or a BBS? Something with tips and protection techniques or design examples?

Msg#: 7635
From: ED NISLEY To: ROBERT MCILVAINE

How about on a bookshelf? The book you want is "Electrostatic Discharge and Electronic Equipment: A practical guide for designing to prevent ESD problems," by Warren Boxleitner. It's published by IEEE Press, ISBN 0-87942-244-O, IEEE order number PC02352, and will set you back the usual 50 bucks or so that technical books seem to cost nowadays.

You get lots of drawings that just don't come through very well in Internet-standard flat ASCII, too!

Msg#: 7642
From: ROBERT MCILVAINE To: ED NISLEY

Thanks for the recommendation; I'll pick it up. Basically, I have general knowledge of the topic. What I need in the short run is to brush up so I can answer some questions on Monday in a knowledgeable fashion.

If you have any overview type words of wisdom it word be greatly appreciated. I will definitely check out the local Barnes & Noble for the book.

Msg#: 8065
From: GEORGE NOVACEK To: ROBERT MCILVAINE

Beside Ed's suggested book, there is also "EM1 Control Methodology and Procedures," by Donald R. J. White (Interface Control Technologies) and "Lightning Protection of Aircraft," by J.A. Plumer (Lightning Technologies). The books are great references with lots of theory. I have not seen any "cookbook" type publication. I am going to check the one Ed suggested.

Msg#: 8652
From: ED NISLEY To: GEORGE NOVACEK

Do tell me (and everyone else!) what you think of Boxleitner's book in comparison to the others; I haven't done a review of the field and ought to know how it stacks up. Thanks!

CONNECTIME

Msg#: 8675
From: GEORGE NOVACEK To: ED NISLEY

I have not read the book yet. I was going to order it, wrote the particulars on a piece of a paper, and lost it. If you would not mind giving it to me again, I *promise' not to lose it this time.

I have read a few books on the subject, but have never seen one which could be called a cookbook. All the works I have seen (including a couple of magazines catering to the ESD aficionados) are hardly understandable to people without formal education in electrical engineering or physics (and still remember their math). By the same token, if you can understand those books, you really don't have to read them. They are forever rehashing what the phenomenon is and how it manifests itself. Not one tells you how to get rid of it.

I have my personal suspicion (also due to knowing several authors) that these guys are consultants. They want to overwhelm the reader, show him how clever they are, and convince him they are the only ones who can harness the monster (for a price). They just stop short of calling it "black magic." I have not seen one work which would, in specific terms, give the reader instructions how to proceed in designing his equipment such that the ESD (EMI, EMP, LSS) requirements are satisfied. And yet, the guidelines are very specific and could be almost concentrated into a list of dos and don'ts.

Msg#: 8718
From: ED NISLEY To: GEORGE NOVACEK

One of those days, eh?

I think you'll like Boxleitner's book. For example, from the summary in the "Enclosure Design Guidelines" chapter:

1. The enclosure design must ensure that uninsulated electronic components and lines have at least 2 cm arcing distance from ungrounded metal objects that may be touched by the operator.

6. All shield material must have an EMF within 0.75 V (in the electrochemical series) of the metal they connect to. If not, an intermediate metal connection device must be used.

Can't get much more specific than that, although, as he points out, some of the guidelines are diametrically opposed to each other and to guidelines required by, say, RF design. That's what engineering is all about.. ..

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

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

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 875-2199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

428 Very Useful 429 Moderately Useful 430 Not Useful

STEVE'S OWN INK

I'll Put My Money Where My Mouth Is

As you probably already know, I am an avid proponent of home control systems (HCSs). When I installed my first system in '85, I justified its expense on the energy savings alone. Today it does so many more things that I think of it as an electronic servant.

Besides turning inside and outside lights on and off, the HCS provides environmental control, security, and entertainment system coordination, In fact, the environs which can be enhanced with a little electronic interfacing are almost limitless.

Of course, a quick reality check reveals that home control isn't a quick drop-in installation like a toaster or TV. Lights and appliances have to be wired to the HCS (or have wireless interfaces added). All this takes resources, persistence, and a compelling purpose.

The thousand or so current Circuit Cellar HCS owners are a breed apart. So compelling was their goal that if we hadn't presented a packaged solution, they would have engineered their own. It was hardly a tough selling job.

Once you have satisfied the radical fringe, however, can you market an HCS like any other consumer product? In my opinion, at present, no. Until "HCS" becomes a universally understood term like "VCR," only the technically astute will appreciate its value enough to make a purchasing decision. That "next" group is still the same CAJ readership.

Rather than a compelling goal, new users will base their decision on the three elements of any good investment: confidence that they aren't alone, justification that it makes sense, and instinct that the purchase is cost effective.

Well, have confidence that home control is not a flash in the pan. Meeting future national environmental targets dictates more intelligent coordination among heating, air conditioning, and lighting. The Computer *Applications Journal* plans extensive coverage of home control and building automation starting next year, complete with bonus editorial sections and supplemental offerings. We plan to offer an extensive collection of construction projects including a CEBus interface and broad coverage of the industry in general.

Second, with energy management and security as its primary use, a home control system hardly needs justification. The hundreds of dollars saved yearly from extinguishing unneeded lights and intelligently controlling air conditioning and heating will help pad the wallet. But it's watching an unwanted visitor, at the HCS's prompting, carefully retracing steps off your property as if negotiating a mine field that brings a smile to your face and a feeling of security to your mind.

And yes, finally there is the element of cost. Only you know what you will spend on a new idea. That's something I can't predict.

What I can do, however, is make your decision easier by putting my money where my mouth is. I truly believe that you are missing a great experience if cost is the inhibiting factor. To minimize your cost of entry into home control, I'll give any *CAJ* subscriber an HCS2-DX printed circuit board and the software to build a Circuit Cellar Home Control System free. The only cost to you is shipping and perhaps a BBS call. See page 32 for details.

No, I haven't lost my senses, nor have I hit the lottery. What I do have is a dedicated interest in advancing technology for a common goal. Nothing would make me happier than giving away thousands of DX boards. With that many users communicating and contributing, the day that home control becomes as easy to understand as a VCR becomes that much closer.

Steve