Circuit Cellar INK®

THE COMPUTER APPLICATIONS JOURNAL

October/November 1989 — Issue 11

$3.95

Ctrl

Why 32 Bits?

EDITOR's INK

CIRCUIT CELLAR INK is a practical sort of magazine. We've built a reputation for publishing working projects, hands-on tips, the type of articles other computer magazines just can't seem to find space for. With all that established, why are we devoting an issue to topics surrounding the glamorous, high-priced world of 32-bit processors?

The first reason is simple curiosity. Our staff, like most of the engineers in the world, spends most of the time up to its collective elbows in 8-bit microprocessors and controllers. Without exception, however, they want to know what's happening on the leading edge of processor technology. It's hard to guess what the next project might require, and even harder to know which techniques will migrate down from the rarified atmosphere of 32-bit buses and huge caches, so it's best to learn what's possible now.

In this issue we look at three different 32-bit processor families. Two of them use 16-bit buses, but the lessons learned are valuable nonetheless. The 80386SX project that begins here is designed to show you how to build a complete AT-compatible motherboard or an 80386SX single-board controller. The BCCH16 uses the Hitachi HD641016 in the familiar BCC Bus. If you're a controller fan, don't miss Tom Cantrell's look at the Intel 80960, surely the ne plus ultra of microcontrollers available today. All in all, it's an issue that we're proud of. Even if you don't plan to use a 32-bit processor in your next data acquisition design, I think you'll find it interesting and useful.

ENGINEERING OF THE BEST SORT

A couple of issues back, there was a letter to the editor asking whether engineers (and by extension, magazines written for engineers) should be concerned about possible uses for the devices they design. This is a tangled and very serious philosophical question that I'll continue to skirt here, but I wanted to tell you about one engineer who does care.

Joe Sobieski is a retired engineer living in Pennsylvania. He decided, for a number of personal reasons, that the world needed his talents even though his company could get along without him. Joe turned his mind to the problems of the physically impaired, and he has developed a system that allows even quadraplegics to have an amazing degree of control over their career, their social life, and their environment. His designs aren't gold-plated (I think they illustrate the concept of appropriate technology at its finest) but they do their jobs in a simple, cost-effective, elegant fashion. He doesn't charge for his time or labor, preferring the less tangible rewards that come from simply helping a fellow human being. I'm proud to have him as a CIRCUIT CELLAR INK reader.

Joe would like to share what he's learned with others who want to help. If you are interested, send a large self-addressed envelope to me here at CIRCUIT CELLAR INK, and I'll see that you get a package of information. I'll also see that Joe Sobieski gets your name and address so that a correspondence may begin. There may be an article in the future, but until then this will get the ball rolling.

COMING SOON TO A MAILBOX NEAR YOU

We are always looking for new ways to present quality technical information to our readers. The newest idea, for our subscribers only, is the CIRCUIT CELLAR INK TECH DECK. Now, I know that you're familiar with most postcard decks, but we've added something different to this one. At least 20% of every deck will be CIRCUIT CELLAR INK TECH TIPS, cards that contain processor and peripheral chip descriptions, utility circuits, and technical shortcuts. The first deck should be mailed in mid-October. When you've had a chance to read the TECH TIPS, drop me a line and let me know what you think.

FUTURE PLANS

We have developed our editorial calendar for 1990. It includes the themes for six regular issues plus two special issues that we are planning. If you are interested in writing an article for one of our theme or special issues, call or write to request the Editorial Calendar and Author's Guide.

[illegible]

Curtis Franklin, Jr.
Editor-in-Chief

FOUNDER/ EDITORIAL DIRECTOR
Steve *Ciarcia*

PUBLISHER
Daniel Rodrigues

EDITOR-in-CHIEF
Curtis *Franklin, Jr.*

ASSOCIATE PUBLISHER
John Hayes

ENGINEERING STAFF
Ken Davidson
Jeff Bachiochi
Edward Nisley

CONTRIBUTING EDITORS
Thomas Cantrell
Jack Ganssle

CONSULTING EDITORS
Mark Dahmke
Larry Loeb

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION CONSULTANT
Gregory Spitzfaden

ART & PRODUCTION DIRECTOR
Tricia Dziedzinski

PRODUCTION ARTIST/ILLUSTRATOR
Lisa *Ferry*

BUSINESS MANAGER
Jeannette Walters

STAFF RESEARCHERS

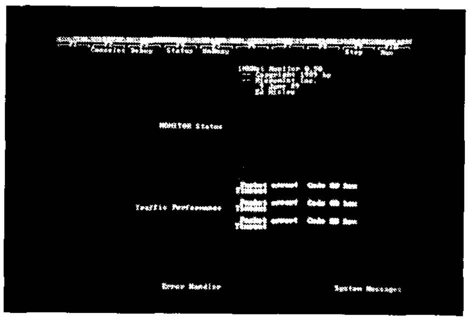
Northeast
Eric *Albert*
William *Curle* **w**
Richard Sawyer
Robert Stek
Midwest
Jon E/son
Tim McDonough
West Coast
Frank Kuechmann
Mark Voorhees

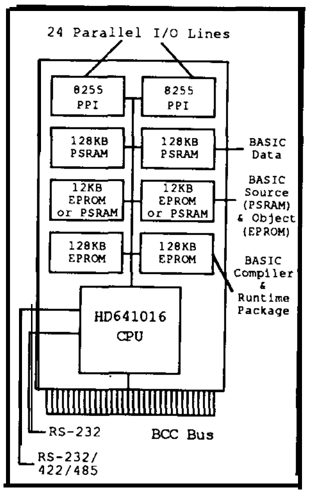
Cover Illustration by Robert Tinney

Circuit Cellar INK®

Ctrl

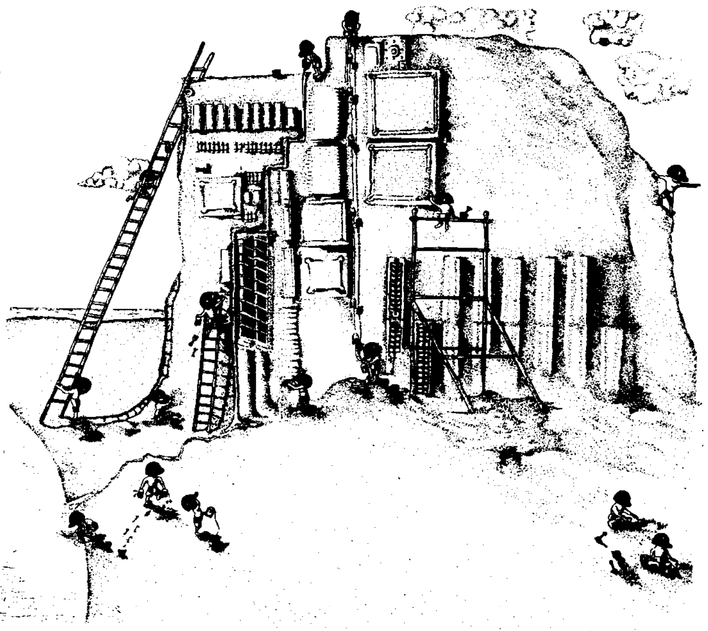
FEATURES





DEPARTMENTS

THE COMPUTER APPLICATIONS JOURNAL



Circuit Cellar BBS-24 Hrs. 300/1200/2400 bps, 8bits, no parity, 1 stop bit, (203) 87 1- 1988.

The schematics provided in Circuit Cellar INK are drawn using Schema from Omation Inc. All programs and schematics in Circuit Cellar INK have been carefully reviewed to ensure that their performance is in accordance with the specifications described, and programs are posted on the Circuit Cellar BBS for electronic transfer by subscribers.

CIRCUIT CELLAR INK (ISSN 0896-8985) Is published bimonthly by Circuit Cellar Incorporated, 4 Park Street.Suite 20, Vernon. CT 06066 (203) 875-275 1. Second-class postage paid at Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions $14.95, Canada $17.95, all other countries $26.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders to Circuit Cellar INK, Subscriptions, P.O. Box 2099, Mahopac. NY 10541 or call (203) 875-2 199.

-POSTMASTER: Please send address changes to Circuit Cellar INK. Circulation Dept., P.O. Box 2099, Mahopac, NY 10541.

Ctrl

READER'S INK

Letters to the Editor

COULD IT BE THEIR BREATH?

Great Magazine! I first got into hardware learning Forth on the Apple II ("Wow, I can influence the external world."), but it was sitting and examining the SB180 schematics closely with a TTL data book next to me that taught me address decoding, bus architecture, and system building. I was delighted to learn of CIRCUIT CELLAR INK and enjoy the fact that you don't aim at end users.

I'd like to see a basic article on assembling. It was an effort for me to learn how to construct projects, evolving from point-to-point wiring through wire wrap and etching. I sense that a piece on the practicalities of putting a schematic onto a board would be valuable to many readers.

I'm curious about why you avoid the *6xxx* series of chips. I learned on the very quick, very easy 6502 and love my 6809. I use a 6809 system for hardware design with multitasking OS-9. I'm aware that the Z80 is frequently used as a controller but Motorola's 680*x* family is a good series of processors. The evolution up to a 68008 is nice since it's still an 8-bit bus processor with no segments and a large address space. In addition, OS-9 and OS-9/68000 are often compatible at C source code level.

Anyway, keep it up and keep it practical. I don't care about the new VAX 9200-to-Unisys Mainframe interface, but I do care about SCSI, HPIB, homebrew LANs, 64180s, and controller applications. You're doing well so far.

Chuck Yerkes
Brooklyn, NY

Thanks for the letter of commendation and recommendation. We are dedicated to being a practical, hands-on source of information for all those who sfill roll up their sleeves to build computer and controller applications.

Thereisn't a conscious bias against any series of controllers orprocessors here. Thesimplefacf is that theengineers here have more experience with Intel controllers and processors than with those from any other vendor. We have taken great pains to include projects using the 68000 and 6502, and a 68HC11 project is in the planning stage.

ON WORKING SMARTER

In CIRCUIT CELLAR INK #9, I have two bones to pick:

The first, with "An Intelligent SCSI Data Acquisition System.. ." by John Eng, deals with the SCSI termination scheme shown in Figure 6. When the "termination enable jumpers" are removed, the termination resistors will induce serious cross-coupling between all SCSI bus signals! This is just the type of problem termination is supposed to avoid.

Termination of a variable-length bus, whether SCSI, Unibus, or any other, is best served by termination plugs. Real SCSI supports this with +5V supplied at one end of the bus specifically for this purpose.

"Cheating" on standards, whether by using physically different connectors or by playing fast and loose with electrical and timing specs, is a recipe for erratic and unreliableoperation. If you know exactly what isgoingon in all subsystems you can usually get away with it; otherwise it's long nights and great aggravation.

The other cavil is with "Computing CRCs in Parallel" by Jack Ganssle. It reminded me of the old saying "When your only tool is a hammer, all problems resemble a nail."

The first part of the article reminded me of a similar article, "Calculating CRCs by Bits and Bytes" in the September 1986 issue of BYTE, which was far clearer in both the mathematics and the circuit descriptions for the serial algorithm.

The most distressing part was the programming example in C for the so-called "parallel" solution. It seems a retrofit from his PLD equations; why else avoid use of the ^(XOR) operator? Worse, the algorithm is not parallel; it still works on one bit at a time. As the BYTE article shows, there is a far better parallel algorithm; in the case of CRC-CCITT, an 8086 code fragment is shown here:

```
MOV   AX,crc     ; get current CRC
XOR   AH,data    ; bring in data
MOV   BH,OFOh
AND   BH,AH      ; partial result
MOV   BL,AH      ; partial result
ROL   AX,1
ROL   AX,1
ROL   AX,1
XOR   AL,BL
ROL   AX,1
XOR   AH,BL
ROL   AX,1
ROL   AX,1
ROL   AX,1
XOR   AL,BH
ROL   AX,1
XOR   AH,BH
MOV   data,AX   ; store updated CRC
```

The code is reasonably fast and clean, and this method has been used by myself and others in fast routines for data communications. It is more efficient implemented in assembly since most computers have the ROL primitive but C does not.

I applaud your request for working smarter: Now, if you could just apply it in your articles.. .

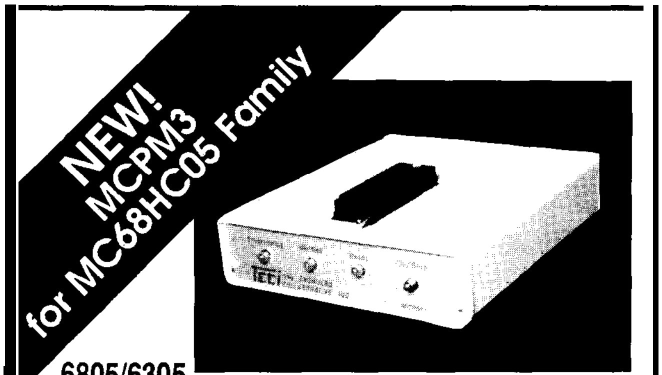
William D. A. Geary
Deer Park, NY

Thank you *for taking the time to write* with your *concerns about issue #9. You raise some* interesting points.

In ***discussing* your** concerns *over SCSI termination with the membersofourengineeringstaff, theconsensus was thatyouare mostly correct. The resistors will, if left in place, induce cross-coupling. At practical issueis whether this matters. Experience has shown (and Mr.* Engmakes *mentionof thefact) that SCSI is a rather "tolerant" specification. This tolerant nature has led to some difficulties (especially in the case of Macintosh SCSI devices versus the rest of the SCSI world), but it does allow for continued operation in a variety of configurations and circumstances. The device shown deals with SCSI termination in the same way that the devices from most Macintosh vendors do, and the device works well. if* you *have an application which requires a long (say, 50') run of SCSI cable passing high volumes of data, then the termination used becomes much more critical. I think you'll agree, though, that this represents an atypical SCSI application.*

As to your *second point, I am fast becoming convinced that CRC algorithms are like sorting algorithms: Everyone has a favorite about which they hold deep religious conviction. My. Ganssle states in his article that the C code is adapted from the PLD equations, and that it is not particularly good C code. It is, however, a fine example of the benefit to be derived from burning* this algorithm into a PLD instead of using RAM-running C *code. All of this is stipulated in the original article. If the object had been to provide a high-quality assembly* language *CRC, your algorithm would almost certainly have been presented.*

Working smarter is a goal towards which most of us strive. Knowing the rules, knowing when they can be broken, and learning from experience *(our* own *and others) all play important roles in helping us achieve our goal.*



6805/6305
SINGLE CHIP
MICROCOMPUTER DEVELOPMENT SYSTEMS

Each of three products allows the IBM PS2/PC/XT/AT to be used as a complete development system for the Motorola 6805 series single chip microcomputers. MCPM-1 supports the MC68705 family, MCPM-2 supports the MC1468705 family and MCPM-3 supports the MC68HC05 family. Each system is $495 and includes a programming circuit board or programmer with driver, cross assembler and simulator/debugger software. A system is also available for the HITACHI 63705 ZTAT micro.

TECI *The Engineers Collaborative, Inc.*
Route 3, Box 8C; Barton, Vermont 05822
Phone (802) 525-3458 ***Fax (802) 525-3451***

Reader Service #121

68000 for the PC-AT

Put a complete 66000 computer into your PC

An exciting 68000 development platform that gets your software out!

- Actually runs a 68008—no simulations required to test your code.
- Real-time embedded multitasking applications can be developed using VRTX32 and RTscope.
- Familiar tools (editor, cross-assembler) can be used on the PC to write and document programs.
- Downloading to the system is fast: on-board monitor provides direct communication with the PC through the I/O channel.

PLUS you can develop multiprocessing techniques and programs by installing two or more boards in the PC.

Features of the MISTER-8

• Low cost: operates in any PC, AT, or clone. • Two or more boards can be installed to do multiprocessing. • Communicate using the PC's serial port and your favorite modem program — or directly through the FIFO. • 126K ROM in three application sockets; 64K SRAM.

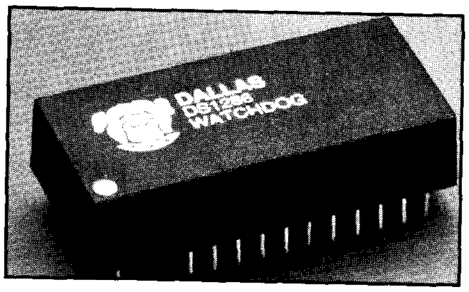
MISTER-8 with monitor EPROM . , ... , ... $495

MICRO RESOURCES ®
Making Technology Work
717-524-7390 or
60 SOUTH EIGHTH STREET, LEWISBURG, PA 17837 717-523-0777

Reader Service #14

NEWPRODUCTNEWSNEWPRODUCTNEWS

Watchdog Timer Mimics Bytewide Static RAM



A new chip from Dallas Semiconductor combines a real-time clock, alarm, watchdog timer, and interval timer in a standard 28-pin package that can be plugged into an existing memory socket. The **DS1286 Watchdog Timekeeper** contains an internal lithium battery and quartz crystal to eliminate the need for external circuitry. In the absence of system power, data is retained for more than ten years.

Calendar information can be read or written in the same manner as bytewide static RAM. Specific memory locations are designated as day, month, year, etc., and are automatically updated to the current time. Time intervals from hundredths of a second to years are tracked with its leap-year-compensated 400-year calendar.

The chip has three programmable timing functions in addition to the calendar. The first is a watchdog timer that guards against computer malfunction by restarting the microprocessor if the computer does not check in at user-specified time intervals. The second and third are calendar and interval timers set in conjunction with an alarm.

A unique double-buffering scheme allows the user to access accurate calendar data instantaneously. Two copies of the data are maintained so that the user can read one copy while the other is being updated. In addition, 50 bytes of nonvolatile RAM are provided for extra storage. Intelligent control circuitry write-protects this memory when power goes out of tolerance.

The DS1286 Watchdog Timer costs $13.75 in 100-piece quantities.

Dallas Semiconductor
4350 Beltwood Parkway
Dallas, TX 75244
(214) 450-0400

New 8096 Full-Function Simulator

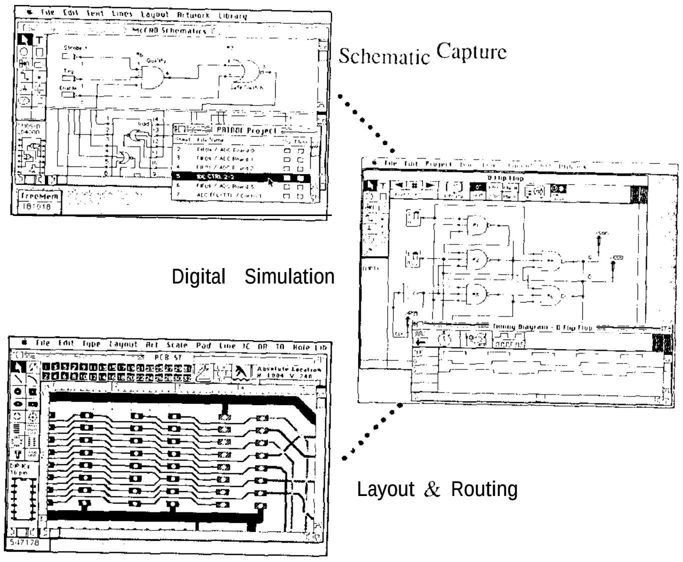
A PC-based full-function simulator for the 8096 family of microcontrollers has been announced by Lear Corn Company. The new simulator allows the PC to be used as a powerful development system to run and debug 8096 programs with complete function and interrupt support including A/D conversion simulation and full implementation of the HSI, HSO, and serial communications features of the 8096 family.

The simulator is fully interactive and permits the user to modify variables and instructions at will through the keyboard. Breakpoints are provided so that programs can be run one instruction at a time, or between selected addresses.

The 80% simulator (SIM96) is being offered at an introductory price of $300.00. The 8096 cross-assembler is available for **$100.00,** and the simulator plus assembler package will sell for $350.00.

Lear Corn Company
2440 Kipling Street, Suite 206
Lakewood, CO 80215
(303) 232-2226

CAE/CAD Software for the Macintosh



High-performance electronic design software for the Macintosh is available from VAMP Inc. McCAD offers an integrated family of software modules to cover the complete design cycle from schematic entry to printed circuit board design to fabrication.

The Schematic Capture module, Schematics **($495),** allows the designer to easily create and revise analog and digital circuit designs directly on the Macintosh. Features include on-line schematic capture, net list, SPICE net list, and parts list extraction, standard libraries, device and text rotation, file import and export via clipboard, and user-definable symbols and libraries. Two PCB layout packages are available. The **PCB-1** ($395) is an affordable, easy-to-use system for creating, editing, and revising circuit board artwork. It does not directly integrate with the other McCAD packages. PCB-ST **($995),** is a high-end circuit board layout package which integrates with Schematics and includes autorouting capability.

Schematics-DS ($895) integrates a digital simulator into the Schematics package. This gives the user the ability to test his captured design for functionality without the need for an actual breadboard. Circuits can be excited and their behavior recorded. Extraction of SPICE data from the Schematics module allows direct analog analysis with many commercially available simulators.

The McCAD Gerber Translator ($175) is a utility which converts any of its PCB databases into standard Gerber format for photoplotting. It is available as a stand-alone program or as part of the EDS-1 package.

The **EDS-1 ($1495)** is a bundled package consisting of the Schematic Capture, Printed Circuit Layout, Routing, and Gerber Translator. It is recommended for the Mac II, but can be utilized on a Mac Plus or Mac SE with a memory upgrade and a hard disk. The minimum configuration for any of the modules is a Mac Plus, with 2 megabytes of memory strongly recommended. All McCAD software is configured for use with the new ROMs.

Evaluation diskettes are available for most of the modules.

VAMP, Inc.
6753 Selma Avenue
Los Angeles, CA 90028
(213) 466-5533

Fuzzy Set Comparator Chip Simplifies Pattern Recognition

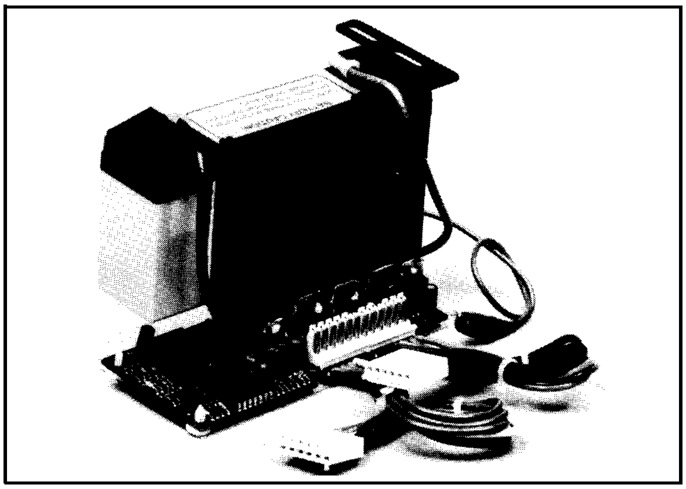
An artificial intelligence IC from Micro Devices Inc. has applications in embedded controllers as well as powerful, high-speed multiprocessor systems. The MD1210 Fuzzy Set Comparator is designed for real-time pattern recognition applications such as objects, characters, voices, etc. The MD1210 is trainable and can be taught to recognize patterns without the use of built-in algorithms or other preprogramming. It incorporates a digital neural network for processing fuzzy data (inaccurate, noisy, variable) to do pattern recognition.

A single MD1210 can simultaneously compare eight unknown data streams to one known data stream, or eight knowns to one unknown. Up to 32 units can be used together on an expansion bus to provide simultaneous comparison of 256 patterns against a single input. It can work in real time or from data stored in RAM or disk.

An Evaluation Kit is available to demonstrate the features and operating modes of the chip. An MD1210 is contained on the Evaluation Kit circuit board which is compatible with the IBM PC or compatible. A NTSC video interface, extensive software, user's manual and data sheets are also provided. The Evaluation Kit with memory costs $250; without memory $200.

Micro Devices, Inc.
Special Products Division of Chip Supply
56958 Beggs Road
Orlando, FL 3281 O-2603
(407) 299-0211

Device Provides Automatic Data Backup With Power Failure



The Boomerang, from Microsync Inc., is a unique system saver that automatically saves system data to hard disk when power is interrupted. Boomerang constantly monitors computer power, and when a brownout or outage is detected, supplies battery power to the computer while it saves the state of the entire system to hard disk. It then shuts down the computer and parks the hard disk heads. When normal power returns, a simple command restores the computer to its previous state with no loss of data.

Boomerang mounts inside any IBM PC or compatible and plugs between the power supply and motherboard. It does not require an expansion slot and supports VGA, Windows/286, and most 386 systems operating with DOS 2.1 or later. Other hardware requirements include 256K of RAM (uses only 18K to run), an internal hard disk drive, and any video monitor. Its internal battery recharges from the computer power supply in normal operation.

Boomerang allows you to turn off power while in an application, and reboot to the same place. This saves reloading the software and repositioning the cursor. Protection for accidently shutting down the computer without saving contents of a RAM disk is also provided.

Boomerang has a 30-day, money back guarantee and a one-year limited warranty. The suggested retail price is $299.00. AT systems with a hard disk capacity greater than 40 megabytes require an additional battery kit priced at $47.95.

Arrick/Microsync, Inc.
2107 W. Euless Blvd.
Euless, TX 76040
(800) 543-0161

C Compiler/Integrated Editor for 6805 Embedded Systems

The C6805 Code Development System from Byte Craft Limited is a fully integrated cross-development tool for the IBM PC and compatibles. It features a C compiler with a built-in macro assembler and integrated editor to enable designers of embedded systems to use the C language for development on 6805/6305 single-chip microprocessors. The Integrated Development Environment accesses in-circuit emulators, PROM programmers, debuggers, and simulators with a few keystrokes to provide complete integration for an entire toolbox.

The C6805 compiles at greater than 6000 lines/minute on an 8-MHz IBM PC/AT or PS/2 Model 50 and produces fast, tight code with an artificially intelligent optimizer. The object code is generated in either S1 or Intel hex formats. The C6805 checks the source code against the target hardware definition, giving on-line error information. Extensive C language support for interrupt routines is also provided, and development time is further reduced by directly generating ROMable code.

Minimum hardware requirements are an IBM PC or compatible with one floppy drive and 512K of RAM. The price for the C6805 Code Development System is $795.00, and a demo diskette is available.

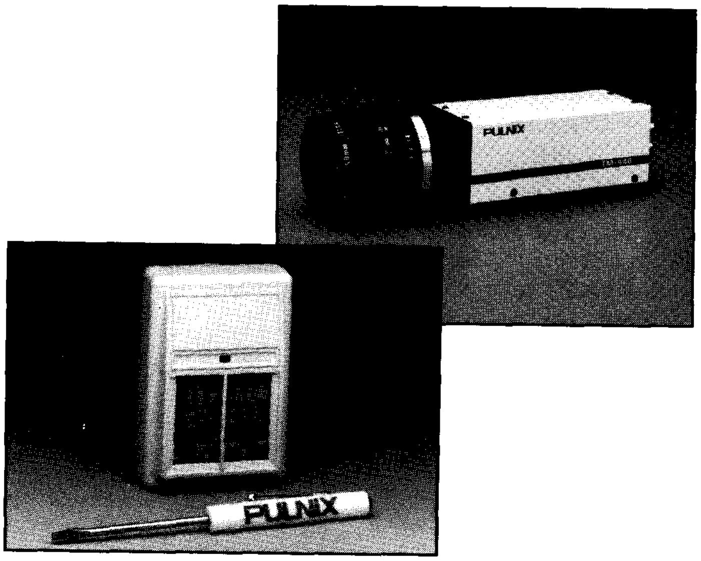
Byte Craft Limited
421 King Street North
Waterloo, Ontario N2J 4E4
(519) 888-6911

New Hardware for Security Applications

Two new products from PULNiX America Inc. offer many possibilities for surveillance and motion detection. The Excel is an attractive, dual-element sensor, passive infrared (PIR) motion detector with models that feature three lens patterns: wide angle (40' x 40'), long range (100' x 20'), and curtain (40' x 10'). The sensitivity of the unit is automatically adjusted relative to the temperature of the environment. For example, if the room becomes warmer, the sensitivity increases; as the room cools, the sensitivity decreases. This eliminates the need for temperature compensation and provides optimal performance.

The unit features an alarm memory to indicate that it has activated an alarm during its armed period. An alarm signal is sent if the supply voltage drops below the 9-volt minimum, and a remote LED allows the walk test light to be enabled or disabled remotely from a control panel. The dimensions of the unit are 3.7 inches high, 2.6 inches wide, and 1.6 inches deep.

The **TM-440/460** is a low-cost full-feature CCD camera with a 0.50-inch high-resolution imager (422 H x 489 V NTSC, 422 H x 579 V CCIR). It features external horizontal, vertical, and composite sync capability, and 2:1 interlace scanning. It has a sensitivity of 3 lux (f1.4) and provides a video signal for auto iris capability. The unit operates from 12 volts DC or 24 volts AC (TM-440L/460L) and weighs less than half a pound. A C-mount with back focus lens mount and tripod mounting provisions are provided. Pricing was not available at press time.



PULNiX America, Inc.
770-A **Lucerne Drive**
Sunnyvale, CA 94086
(408) **733-** 1560

live Video Merges with Computer Graphics

The SPECTRUM NTSC, from Redlake Corporation, merges standard baseband video with computer graphics on a multisync monitor. This is accomplished by digitizing, deinterlacing, and speeding up the video signal to match the scan rates of the computer video output. A proprietary synchronization technique is used to digitize video with poor time base or sync properties, such as that from a VCR in pause mode.

The SPECTRUM NTSC uses an 8-bit digitizer to produce 16.8 million colors and provides much better video quality than a 5-bit system. VGA compatibility, single-screen operation, and live video in all modes are included, and the unit can be programmed to operate with most window shell programs. Other features include resolution up to 800 x 600, fully preserved 4:3 aspect ratio in any mode, hardware pan and zoom, and NTSC output of the digitized image to a VCR or monitor.

System requirements include an IBM PC/AT, or compatible, with 640K of RAM, an analog multisync monitor, and a VGA card. A set of drivers and a demonstration software package are provided with each board. The SPECTRUM NTSC is priced at $2495.

Redlake Corporation
15005 Concord Circle
Morgan Hill, CA 95037
(408) 779-6464
(800) 543-6563

Attention Manufacturers!

Circuit Cellar INK provides its readers with news about significant product developments in hardware, software, and development tools. If your company has a new product that our readers should know about, please send your product announcements to:

Circuit Cellar INK
New Products Editor
4 Park St., Suite 20
Vernon, CT 06066

Ctrl

VISIBLE INK

Letters to the INK Research Staff

Answers; Clear and Simple

AND NOW FOR SOMETHING COMPLETELY DIFFERENT...

We have a problem here which I have never seen mentioned anywhere-that of mildew attackingdiskettes. I imagine it is a common problem in tropical climates in situations where air conditioning is not heavily used. We cannot use air conditioning all the time due to blackouts.

The mildew problem arises when diskettes are not used for a number of months, and seems to favor backup diskettesespecially! Two possibilities suggest themselves: either low-humidity storage using a mini-air conditioner or silica gel to discourage mildew growth; or using a volatile fungicidal agent in storage boxes to kill the mildew and spores. The latter seems the best but we have not been able to locate any suitable fungicide which will not damage the diskettes. Any suggestions?

Incidentally, the mildew can be removed with alcohol but this is very tedious and the surface of the diskette may have been permanently damaged by the mildew.

Andrew Mancey
Golden Grove, Guyana

You have no idea what a discussion your letter kicked up! We have solved many technical problems, but The Mildew That Ate The Diskette isn't one of them. Around here the worst problem seems to be mice storing seeds in the hub hole of a stack of diskettes.

We think your idea of usingsilicagel in a sealed box is much better than trying to find a diskette-safe fungicide. The gel is reasonably inexpensive, easily reconstituted, and doesn't give off horrible chemicalfumes. You should keep all of your diskettes in d y storage boxes to prevent the spores from taking hold.

You should clean your drives on a regular basis; the crud on your read/write heads must beasight to behold. Take thedrives out of your PC and clean them with cotton and alcohol; be careful not to damage delicate mechanisms as you're poking around.

CHIP SELECTION BLUES

Most of your construction articles include a microprocessor or microcontroller and RAM and/or EPROM. When I look at the memory devices which are available, they are offered at a variety of access speeds. How do 1 determine the needed speed for memory to be used with, for example, an 8031 or 80C52-BASIC?

Also, in CIRCUIT CELLAR INK #7 the article on the Image-Wise/PC shows the use of 74LS ICs with 82C54 and 74ALS devices. The article on the Home Satellite Weather Center use 74LS devices with a 74HC154. Why are the logic families mixed? Can you give some guidelines on what type of mixing might be reasonable and desirable?

Bill Dornbush
Farmington Hills, MI

The topic of matching memo y to processors can (and does) fillentirebooks. It turnsout that, as withmanysimplequestions, there are no simple answers!

What you need to do is sit down with the manual for the microprocessor you're using, figure out the timing for all the control signals and data lines (taking into account decoding logic and bus buffers), then examine the RAMs to find out when the data will arriveafter the last control line becomes valid. That's how it's got to be done, eve y time.

When it comes to logic families, you can think of them as fitting under a bell curve, with the high-speed, high-fan-out devices at one end, the low-speed, low-fan-out devices at the other, and most of the families somewhere in the middle. You usually won't have trouble mixing devices from somewhere in the middle in the same circuit, but be careful when trying to use devices from opposite ends at the same time. In general, the circuit's performance will only be as good as the lowest-performance family used. Mixing 74LS, 74HCT, and similar parts as you'll sometimes find in CIRCUIT CELLAR INK articles won't present major problems. For the final say, though, you should consult the data books for whatever families you're using.

COMPUTING ON THE RUN

I am currently working on a mobile, autonomous robot system. My background is primarily mechanical engineering with a fair amount of motor application experience, but I have limited experience with computers. I am in the preliminary design stages and need assistance in choosing the central computer to be mounted on the robot.

I have started construction of the design, which is basically a small equipment cabinet on wheels. The lower section houses the drive motors and batteries. The upper section is for the electronics and some sensors. Above this will be mounted the extendible arm and, perhaps, the vision system. I've designed the system so it will be equally functional both indoors and out.

The computer will need to be battery powered and have the ability to work with a variety of sensors, motors, and other interfaces. Ideally, I would like the control computer to talk to smaller motor controllers and other intelligent boards to off-load some of the computation. The central computer would be used primarily for navigation and communication via an RF modem to a remote computer. Other nice features would be that it is rack mountable and have high-level programming software available. I am currently thinking of using an AT-bus motherboard because of the availability of plug-in boards and software, but reaiize there may be some drawbacks.

Thank you for any assistance you can provide me.

Charles Scott **MacGillivray**
Lakewood, CA

Your personal robot project sounds like quite an undertaking. Regarding *thechoice of a central computer system, a fast AT 80286 or 80386 motherboard would certainly be workable, but evenat thatyoumaywellfind thatwriting the real-time software to manage vision systems, rangefinders, proximity sensors, etc. will require timinq capabilities beyond those* found *in most high-*level *languages. In* ahdition, *keep* in mind that the libraries *that typically come with those languages will be primarily designed for graphics, scientific, and business data processing. It will be up to you to supply theassembly language drivers and interface routines needed to talk to your robot's specific hardware.*

*Before you begin, you **might** want to read "Real-Time Software Design: A Guide for Microprocessor Systems" by **Phillip** Heller. The book is published by Birkhauser in* Boston, *Mass, and its ISBN is O-8176-3201 -8. The book is not a tutorial in assembly language programming but it gives practical examples ofsome real-timesoftwareproblems and* presents *various techniques that can beapplied indeveloping thecontrolsoftware.*

In **Visible** INK, the Circuit Cellar Research Staff answers **microcom**puting questions from the readership. The representative questions are published each month as space permits. Send your inquiries to:

INK Research Staff
c/o Circuit Cellar INK
Box 772
Vernon, CT 06066

All fetters and photos become the property of **CCINK** and cannot be returned.

IRS

201 Very Useful
202 Moderately Useful
203 Not Useful

STOCKS..
.................OPTIONS I.....................
..FUTURES

Turn your **PC** into a
MARKET QUOTATION MONITOR
New book covers complete information on financial news and market quotes for your PC from satellite and **FM** radio. Topics include:

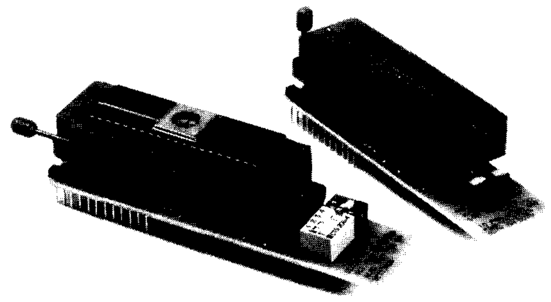
- Data Encryption
- Password Methods
- Receiver Unit Design

Covers quotation processing and data broadcasting from the trading floor to the desktop, $19 plus $2 S/H (includes demo diskette).

Send for FREE catalog of
- DATA RECEIVER **KITS**
- QUOTE DISPLAY SOFTWARE
- **DESCRAMBLING** UTILITIES

CALL — (303) 223-2120 (anytime>
DATArx
111 E. Drake Rd., Suite 7041
Fort Collins, CO 80525

ReaderService #118



87C51 PROGRAMMER $125.

Logical Systems brings you support for the Intel 87C51. rhe UPA87C51 programs thls popular microcontroller on general purpose programmers that support the 2732A. With the UPA87C51 you can program the 8751 and 87C51 security bits and the 87C51 encryption array. Logical Systems, helping you get the most out of your programming equipment with our growing line of adapters. OEM Inquiries welcome.

ADAPTER	PROGRAMS	PRICE
JPA8751	C8751, 8751 H, AMD8753H, 0744	$95.00
JPA87C51	C8751, 8751H,AMD8753H, 0744 87C51, 87C51FA	125.00
UPA63701V	Hitachi HD63701V0	65.00
JPA451 N	Signetics SC87C451 (64 pin DIP)	125.00
JPA63701X	Hitachi HD63701X0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
UPA63701Y	Hitachi HD63701Y0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
JPA63705V	Hitachi HD63705V0	65.00

CALL (315) 478-0722 or FAX (315) 475-8460

LOGICAL SYSTEMS CORPORATION
P.O. Box 6184, Syracuse NY 13217-6184 USA, TLX 6715617 LOGS

eader Service #134

INKnet Part 2

Writing Software for Distributed Control

by Ed Nisley

Now that you know how to share network bandwidth and keep those nodes under control, I can explore message structures, BASIC language extensions, and the MONITOR program. While INKnet serves as a sample implementation, the principles are applicable to any network.

MEET THE MESSAGES

All of the data sent between INKnet nodes travels in messages with the structure shown in Figure 1. While nearly any arrangement for the bytes would suffice, every message on a network must have the same structure so the firmware can locate the control fields. *[Editor's Note:* **Executable** *INKnet software is available for downloading from the Circuit Cellar BBS or on* CIRCUIT CELLAR *INK Software On Disk #11. For downloading and ordering information, see page 78.]*

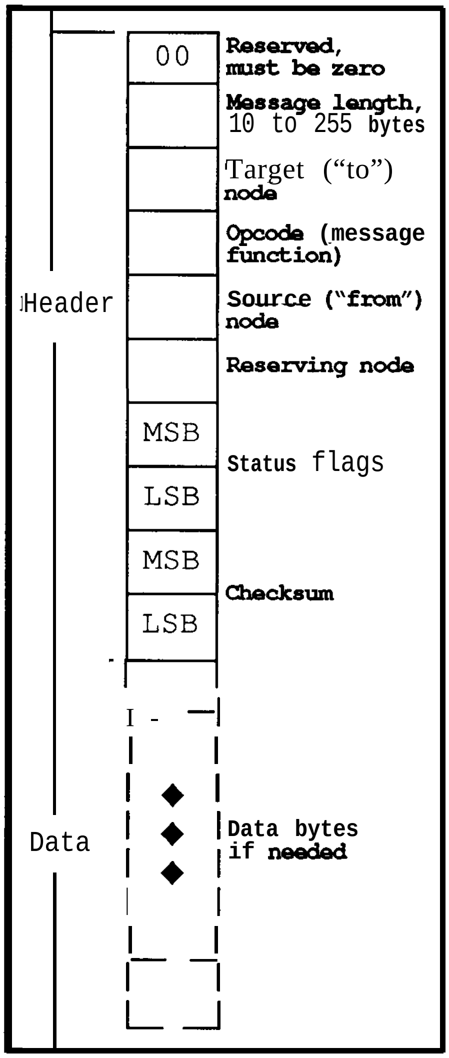


Figure 1—*All INKnet messages have two sections: a header containing routing information and an optional data section.*

The first ten bytes of every INKnet message are the header and carry similar information regardless of the rest of the data. The total number of message bytes is contained in the *LENGTH* field.

The maximum message length is 255 bytes, which is the largest possible *LENGTH* field value. There may be up to 245 data bytes after the header, so longer blocks of data must be broken up by the sending node.

The *OPCODE* field identifies the message contents so the recipient knows what to do with it. Figure 2 is the list of current INKnet opcodes, but others may be added as the need arises. Your application programs may use any opcode values from 00 through 7F hex for whatever purposes you can imagine. Opcodes between 80 and FF hex are reserved for INKnet firmware and software; messages with these opcodes will not be passed on to your application and may interfere with network operations if you use them.

The *SOURCE* and *TARGETNODE* fields indicate which node sent the message and which node should receive it. Because every node "hears" all network traffic, any node can eavesdrop on messages to any other node. The INKnet firmware, however, discards messages not addressed to your application's node.

The *RESERVING NODE* field indicates whether the sourcenode (the one that sent the message) is reserved for the exclusive use of another node. Nodes ordinarily accept messages from any other node as long as the target field matches their network address. Reserving a node simply means that the reserved node will respond only to messages from the reserving node.

Figure 3 defines the *STATUS* field bits, which report the network firmware status to other nodes. The most important bit is *BUSY*, which indi-

Hex	Opcode	Function
80	POLL	Poll from master to node
81	STATUS	Status response from node to master
82	UNBUSY	Force node "unbusy" regardless of actual status (use with caution!)
88	RESERVE	Reserve node for exclusive use of "from" node (except for polls & responses)
89	RELEASE	Release node for general use (may be sent by any node)
90	CONSOUT	Console output (to display)
91	CONSIN	Console input (from keyboard)
92	CONSECHO	Echo of console input message
93	CONSFILE	Console input (from file)

Figure 2— lhe *Opcode In each message identifies the message function and determines ifthere is any data following the header. Network programs should use opcodes between 00 and 7F for internode messages. Opcodes between 80 and FF are used for network management; user programs will not receive these messages.*

cates whether the node can accept a new message. Because the RTC52 nodes have limited memory, they can handle only one incoming message at a time. If a message arrives while the node is busy, the firmware will discard the first message and set the *OVERRUN* status bit.

The *CHECKSUM* field is computed from all of the rest of the bytes in the message by treating pairs of bytes as one 16-bit number and adding them together. If the message has an odd number of bytes, the last byte is padded with a low-order zero byte before theaddition. Theresultingsum is truncated to 16 bits, complemented, and inserted into the message. If you are implementing this algorithm, remember to zero the checksum field before you add up all the bytes in the message!

The firmware eavesdrops on all network messages and sets the *CHECKSUM* status bit if any message has an invalid checksum. Damaged messages are discarded; because any byte may be incorrect, there is no way to determine which node should have received the message and no way to “patch up” the results. While we could have used an error-correction code, 8051 processors can be swamped very easily.. .and running network firmware is not the processor’s most important job.

Figure 4 shows the bytes in a series of INKnet messages. Notice that the master node sends a poll to itself during each polling cycle, then responds with an ordinary status message. This simplifies the node firmware by treating the master node as “just one of the gang” for all messages other than the polls themselves.

TIMING IS EVERYTHING

Although all INKnet data travels in messages, some information is conveyed by message timing. The INKnet master node imposes restrictions on the net to ensure orderly traffic flow. As with all designs, the timing specs are a compromise among conflicting requirements.

Net messages must be separated by a minimum of 5 milliseconds to give an 8052 node ample time to move all 255 bytes of a message from one spot inmemory to another while doing some “in-flight” data analysis. If your net design uses slower processors, you may need to increase this delay time.

A node must start its response within 15 ms of the end of a poll. If the master node doesn’t detect a response within that time, it concludes that the node is missing or dead and continues with the polling cycle. Longer delays allow the nodes to run with interrupts disabled during critical code sections, but would slow largenetworksdown.

The INKnet MONITOR program polls the nodes in ascending order starting with Node 0 and stopping at a specified maximum node. You can improve the performance of your network by assigning node addresses contiguously between zero and the highest node number. The net master could simply stop polling nodes that do not respond, but then there must be another timeout to allow new nodes to join the network conversation.

The polling scheme can be as clever as your programming skills allow, but we think simpler algorithms are more robust, easier to debug, and certainly more practical to implement on smaller processors with limited RAM and processing power.

Because all of the nodes are connected in parallel, any node that transmits out of turn can disrupt communication on the whole network. The RTC boards have an LED on the transmitter-enable signal, so you can determine if a board is “stuck on” by simply looking at the LED. This problem is common to all parallel bus networks; finding Ethernet coax cable shorts is an Olympic sport in LAN installations worldwide!

Nodes may use any timing method to meet the specifications, but the easiest is to measure delays against a free-running internal clock ticking every 5 ms. Because the end of an incoming message will not normally

Bit	Name	Function
15	BUSY	Node is busy handling previous message
14	OVERRUN	At least one previous message was lost
13	ECHO	Node will echo all CONSIN messages
12	LFSTRIP	Node strips LF after CR on CONSIN msgs
11	CSUM	Node detected a checksum error
10	reserved	
9	reserved	
8	MASTER	Node is network master

Flag bits 0 through 7 are reserved.

Figure 3—*The Status Flags field indicates the node state. Each node monitors the flags from all other nodes to update its own tables.*

	Zr	Ln	To	Op	Fr	Rs	Flags		Chksm							
Poll to node 0	00	OA	00	80	1F	FF	31	00	AE	77						
status response	00	OA	1F	81	00	FF	30	00	AF	76						
Poll to node 1	00	OA	01	80	1F	FF	31	00	AD	77						
Message to node 0	00	10	00	90	01	FF	30	00	A9	6F	48	65	6C	6C	6F	21
Poll to node 2	00	OA	02	80	1F	FF	31	00	AC	77						
Message to node 1	00	0B	01	01	02	FF	30	00	DE	F5	ED					
Poll to node 31	00	OA	1F	80	1F	FF	31	00	8F	77						
Status response	00	OA	1F	81	1F	FF	31	00	8F	76						

ASCII for "Hello! "

Binary data

Figure 4—*One poll cycle on a net with four nodes:* **RTC52** *boards at nodes 0 through 2 and the master node at 3 l (lF hex).*

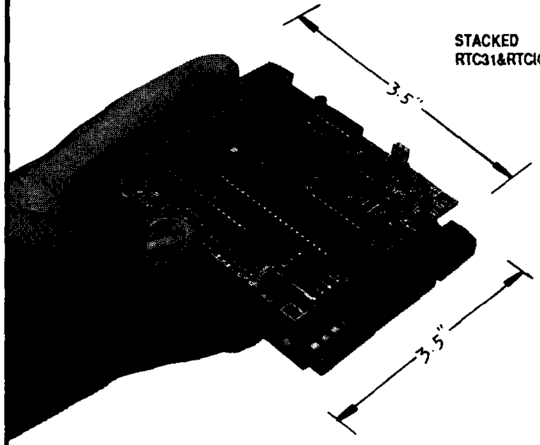
occur exactly at a clock tick, the firmware must wait for two ticks before starting the outgoing message. The resulting delay can range from 5 to 10 ms depending on whether the timing starts just after or just before the first clock tick.

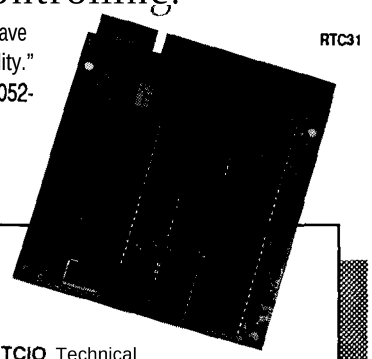
The node clocks are not synchronized, so the delay timings will vary between the minimumand maximum values from message to message. The timing specifications take these variations into account; it can be instructive (and fun) to watch the bus traffic on an oscilloscope to see the shifting responses.

Using a higher clock rate is possible, but the instructions required to handle the timer interrupts every millisecond (for instance) would probably go to better use running the

MICROMINT Introduces *"Micro"* Controlling!

After years of experience in manufacturing OEM controller boards and talking to customers, we think we have hit upon just the right combination of format and function to satisfy even the toughest case of "relay mentality." Realizing that not every computer-controller application warrants a Cray X/MP, Micromint offers a tiny 8031/8052-based controller board for those dedicated and cost-sensitive installations.





RTC31 and RTC52
Technical Specifications
•8031 processor (RTC31) or Micromint 80C52-BASIC processor (RTC52)
•11.05-MHz system clock
Uses 8K or 32K memory chips
•Up to 64K bytes of RAM or EPROM
•5-volt-only operation
•11o-19200 bps RS-232 and/or RS-485 serial port
-Use stand-alone or networked
•12 bit of parallel I/O
*Vertical-stacking expansion bus
•Screw terminal connections
•Small 3.5"x3.5" format
•80 mA typical ooerating current (RTC52)

RTC31-1 8031 Controller	$119.90
RTC31-1 OEM 100-Quanity Price	$79.00
RTC52-1 80C52 Controller	$139.00
RTC52-1 OEM 100-Quanity Price	$99.00

RTCIO Technical Specifications
•Three bidirectional parallel ports (24 bits)
•8-channel, 8-bit A/D (0-5V); 9,000 samples/sec
•4-channel, 8-bit D/A (C-5V); 2-µs response time
•Battery-backed clock/calendar and presettable time-interrupted capability
•DC To DC conversion-5-volt-only operation
•Screw terminal connections
*Small 3.5"x3.5" format

RTCIO RTCIO board with parallel I/O arid A/D converter	$129.90
RTCIO OEM 100-Quantity Price	$99.00

MICRO MINT

Micromint, Inc.
4 Park Street, Vernon, Connecticut 06066
Tel: (203) 871-6170 • Fax: (203) 872-2204

Reader Service #138

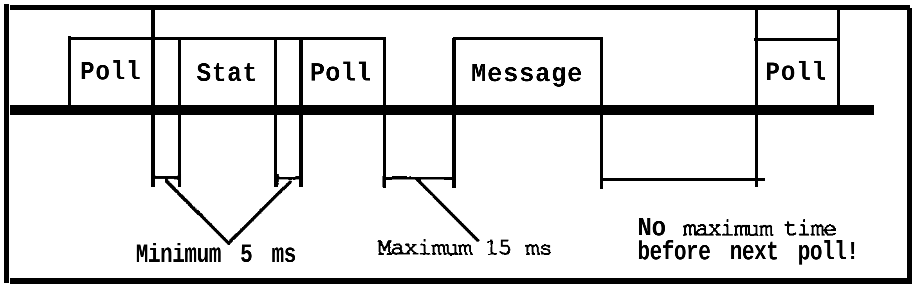


Figure 5—INKnet message timing requirements. The nodes may use any timing method as long as the result meets specifications!

node's control algorithm. The real problem occurs with MONITOR, because the AT's clock normally interrupts once every 54.9 ms: we had to pump that up by a factor of 11! You can try faster clocks on your network to find the best tradeoff between traffic and computations.

Figure 5 summarizes the INKnet timing requirements. It is worth mentioning that there is no maximum time limit between the end of a message and the next poll; this simplifies debugging by single-stepping poll messages.

TERMINAL CONNECTIONS

The BASIC-52 interpreter is a (moderately) high-level language in a single-chipcontroller. Theinterpreter communicates with theoutside world through the 8052's on-chip serial port. Whatever else the network does had better allow this to continue!

A simple ASCII terminal won't work, however, because the network hardware uses an RS-485 transceiver and network messages aren't legible when translated into ASCII characters. The solution is a chunk of 8052 firmware to convince the BASIC interpreter that it is still connected to a serial terminal, while really shuffling characters among net messages.

On the other end of the network, MONITOR collects keystrokes into console input messages and displays the contents of console output messages on the screen. Because the AT has lots of RAM, MONITOR can maintain console output histories for all the BASIC network nodes; switching between nodes is a matter of a few keystrokes.

The 8052 firmware gets control during the BASIC-52's power-on reset sequence. It checks the memory size (which must be either 8K or 32K bytes) and reserves space for the network buffers at the high end of RAM. The value of MTOP (Memory TOP) used by the interpreter marks the last byte below these buffers, so BASIC won't try to write over the buffers.

A peculiarity of theBASIC-52code surfaced while I was writing the firmware: The Death of Ten Thousand Nibbles. The interpreter checks incoming bytes for special control characters such as Ctrl-S, Ctrl-Q, and Ctrl-C, but discards each character after the test. This is reasonable on a serial terminal because the characters come directly from the keyboard, but the effect is to drain the network's ring buffer before the BASIC program can examine the characters! I'm looking for a way to allow type-ahead while not eliminating the control characters, but everything so far looks like a force fit.

EXTENDING BASIC

The console I/O functions handle communication with the display and keyboard, but the nodes must communicate with each other, too. Most languages have a fixed set of keywords that represent everything the language can do for you, but generally that doesn't include networking! That is true of BASIC-52, but, unlike most other languages, it has a clean way to add new keywords for special applications. The firmware adds a NET statement to give a simple interface between your BASIC code and the network routines.

Figure 6 lists the new BASIC statements created using the NET keyword and their arguments and return values. These are functional in both direct and program mode, so you can try something out by typing it at the prompt, then incorporate the result in your program with no changes. The current firmware does not allow any BASIC statements other than REM on the same line as a NET statement.

TheNET OUTPUTstatementsends a message to another node. Your program must build the message in a RAM buffer, then execute a NET OUTPUT with the buffer address. The firmware copies the message into a transmit buffer, fills out the header, and sends the message when it receives the next poll from the master node. Your program will regain control immediately after the message is copied to the firmwarebuffer; the firmware uses interrupts to send the message.

Similarly, NET INPUT specifiesan addressat which the firmware will copy a new message from the receive buffer. Your program can then analyze the message opcode and decide how to handle it. The firmware returns only messages addressed to your node address, so NET INPUT cannot be used to eavesdrop on messages for other nodes.

Both NET INPUT and NET OUTPUT return status information on the BASIC-

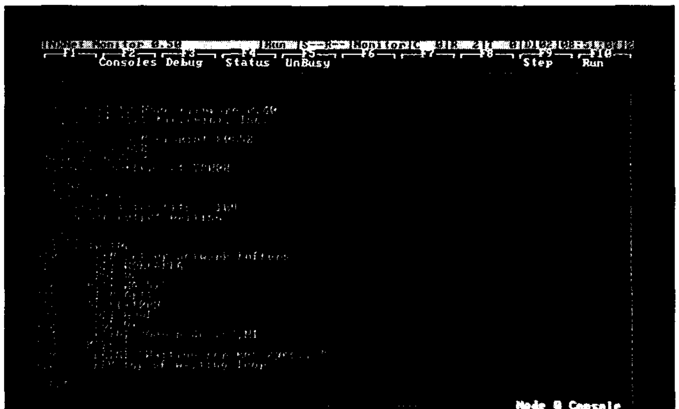


Photo 1 — MONITOR can simulate up to 31 serial terminals, each with a screen similar to this. Each node produces output on a separate screen.

```
IET                Displays copyright & useful info
IET NODE           Returns node's own network address
IET SET,NODE,x     Sets the node's network address to x
IET SET,NODE=x     Ditto
IET MNODE          Returns the master node's address
IET CNODE          Returns the console node's address
IET WORKAREA       Returns network buffer start address
                     (equals MTOP+1 after power-on reset)
IET STATUS          Internal status for own node
       Bit definitions are:
         l-unused, will be zero
         6-checksum error on a previous packet
         5-console output sending in progress
         4-console output waiting for a poll
         3-NET OUTPUT sending in progress
         2-NET OUTPUT waiting for a poll
         l-NET INPUT overrun, newest packet preserved
         O-NET INPUT message ready
       Bit 6 (checksum) is cleared by calling this function
IET STATUS,x        Network status for node x
       High byte contains:
         FF-if node is not reserved
         y -if node is reserved by node y
       Low byte contains:
         7-BUSY handling previous packet
         6-overrun occurred
         S-echoes console input
         4-strips LFs after CRs
         3-checksum error occurred
         2-unused
         l-unused
         O-node is current net master
YET RESERVE,x       Reserves node x for exclusive use
       Return values:
         0-success
         l-could not send packet
YET RELEASE,x       Releases node x for use by other nodes
       Return values:
         o-success
         l-could not send packet
NET OUTPUT, addr    Sends message packet from addr
       Required header format, byte offsets:
       Items marked with *must be filled in prior to call!
         0-must be zero
       ★ l-message length, including header (minimum 10)
       ★ 2-target node
       ★ 3-opcode
         4-source node
         S-reserving node
         6-flags high byte
         7-flags low byte
         8-checksum high-order byte
         g-checksum low-order byte
       ★ 10-optional message data starts here
       Return values:
         0-message sent (actually, message in progress)
         l-another message was pending
NET INPUT, addr     Receives message packet at addr
       Header starting at addr, byte offsets:
         0-must be zero
         l-length of message, including header (minimum 10)
         2-target node
         3-opcode
         4-source node
         5-reserving node
         6-flags
         7-flags
         8-checksum
         g-checksum
         lo-message data starts here
       Return values:
         0-message received in buffer
         l-no message available
         2-overrun occurred, data is most recent packet
       Bit 1 (overrun) and bit 0 (message ready) of the result
             returned by NET STATUS are cleared by this function.
```

Figure 6— The *NET keyword adds several new statements to the BASIC-52 language. Each statement deals with a different aspect of network control.*

uCONTROL

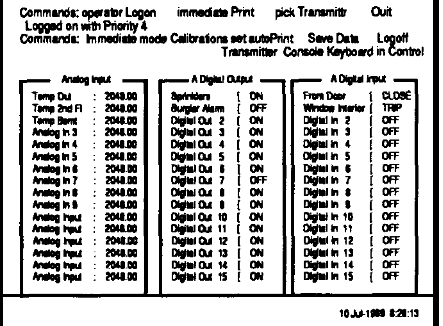
Menu-driven software to monitor, display, and control your home or production system from the site or from a Remote location.

uControl Features
o Display:
 up to 16 analog inputs
 up to 32 discrete inputs
 up to 32 discrete outputs
0 Sample Rate
 update all within 1 sec.
o Alarms
 Switch Discrete Output on:
  analog threshold
  'trip' of discrete line
o Password Protection
  4 Priorities
Price: $150.

uControl-Remote ADDS
o Dial up from remote to
 access all features
o Automatic dial-out on
 Fault Condition
Price: $100. (requires **uCONTROL)**

uControl-History ADDS
o MS-Windows base d display
 customization program.
o Historical Plotting of any
 input or output vs. time.
Price: $150. (requires **uCONTROL)**

Requirements:
uControl. IBM PC/XT/AT (compatible) system **with** 512k memory; compatible data acquisition card (inquire about boards and systems supported.)
-Remote. also Hayes compatible modems at each end.
-History. MS Windows Release 2.1 or later.



```
Commands: operator Logon    immediate Print    pick Transmittr    Quit
 Logged on with Priority 4
Commands: Immediate mode Calibrations set autoPrint   Save Data   Logoff
                                     Transmitter  Console Keyboard in Control

 Analog Input                A Digital Output           A Digital Input
Temp Out      : 2048.00   Sprinklers      [ ON     Front Door      [ CLOSE
Temp 2nd Fl   : 2048.00   Burglar Alarm   [ OFF    Window Interior [ TRIP
Temp Bsmt     : 2048.00   Digital Out  2  [ ON     Digital In  2   [ OFF
Analog In 3   : 2048.00   Digital Out  3  [ ON     Digital In  3   [ OFF
Analog In 4   : 2048.00   Digital Out  4  [ ON     Digital In  4   [ OFF
Analog In 5   : 2048.00   Digital Out  5  [ ON     Digital In  5   [ OFF
Analog In 6   : 2048.00   Digital Out  6  [ ON     Digital In  6   [ OFF
Analog In 7   : 2048.00   Digital Out  7  [ OFF    Digital In  7   [ OFF
Analog In 8   : 2048.00   Digital Out  8  [ ON     Digital In  8   [ OFF
Analog In 9   : 2048.00   Digital Out  9  [ ON     Digital In  9   [ OFF
Analog Input  : 2048.00   Digital Out 10  [ ON     Digital In 10   [ OFF
Analog Input  : 2048.00   Digital Out 11  [ ON     Digital In 11   [ OFF
Analog Input  : 2048.00   Digital Out 12  [ ON     Digital In 12   [ OFF
Analog Input  : 2048.00   Digital Out 13  [ ON     Digital In 13   [ OFF
Analog Input  : 2048.00   Digital Out 14  [ ON     Digital In 14   [ OFF
Analog Input  : 2048.00   Digital Out 15  [ ON     Digital In 15   [ OFF

                                                          10-Jul-1989 8:28:13
```

30-day money-back **quarantee**

(617) 861-0181
FAX (617) 861-1850

Unkel Software *Inc.*

62 **Bridge Street,** *Lexington, MA 02173*

Reader Service #15

52 argument stack. Your program must POP the status into a variable to test it. What your code does with the result **dependson your** application, of course.

Listing 1 shows two BASIC program fragments that exchange a simple "HI" message. Of course, most applications will be sending raw binary sensor readings and switch settings, but the mechanics of setting up the message header and data area are similar. The BASIC-52 XBY function is a combination of the **PEEK** and **POKE** functions found in other BASICs; it refers to a byte in External Data Memory and can either **PEEK** or set it depending on which side of the equal sign it's on.

Although you can locate the message buffers used in **NET INPUT** and **NET OUTPUT** anywhere in RAM, the most convenient spot is often just below the network firmware buffers at the high end of RAM. Listing 2 shows how to use the **NET** WORKAREA statement to get the addressofthenetworkbuffers, thenreset **MTOP** to allocate space for your messages. You can use a single buffer for both input and output, but it may be easier to have at least one buffer for each direction.

The **NET OUTPUT** statement will not return control to your program until your message is copied into the firmware buffer. If you call **NET OUTPUT** while that buffer is in use, your program will pause until the first messageistransmitted. While some programs can tolerate this delay, you may want to continue processing until the buffer is empty again before sending the next message.

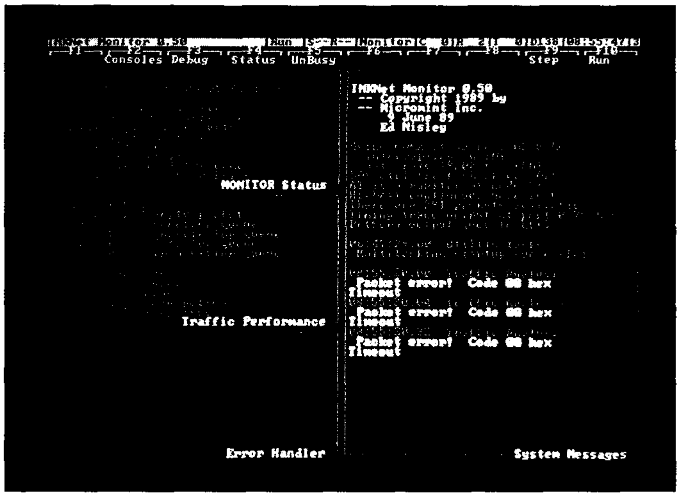
The **NET STATUS** statementsprovideinformationon the state of the firmware and buffers. If you include a node number (as in **NET STATUS, 3**), the return value will be the most recent status bits from that node. If you omit the node, the return valueapplies to the node running the program and provides more detailed information.

```
Transmitting node, using RAM buffer at address B1:

        100 XBY(B1+1)=12       : REM message length (10 + 2)
        110 XBY(B1+2)=2        :REM target node
        120 XBY(B1+3)=12h      :REM opcode chosen by application
        130 XBY(B1+10)=ASC(H)  : REM data values (2 bytes)
        140 XBY(B1+11)=ASC(I)
        200 NET OUTPUT,B1      : REM ship it off!

Receiving node using RAM buffer at address B2:

        100 NET INPUT,B2       : REM get any waiting input
        110 POP S1     : REM recover from stack
        120 ON S1 GOTO 130,100,125   : REM decode status value
        125 PRINT "Overrun!"
        130 PRINT "Message from node",XBY(B2+4):REM show source
        140 PRINT "Opcode is",
        145 PH1.XBY(B2+3)      :REM and opcode
        150 ML=XBY(B2+1)       :REM show length
        160 PRINT 'Message length is",ML
        170 PRINT "Data values are:"
        180 FOR I=B2+10 TO B2+ML-1    : REM show message data
        190 IF (XBY(I)<>0AH).AND.(XBY(I)<>0DH) THEN PRINT
              CHR(XBY(I)),
        200 NEXT I
        210 PRINT      : REM skip to next line
```

listing 1 *-Simple* **message exchange.**

Listing 3 shows how to find out if a previous message was sent before using NET OUTPUT; this eliminates the risk of "hanging up" while waiting for the first one to clear.

The network firmware maintains an internal table that records the message header status bits from each node. The firmware marks any node receiving a message (other than a poll) as "BUSY" until it sends out a message indicating that it is no longer busy. This ensures that two nodes will not transmit messages to a single node during one polling cycle.



INMNet Monitor 0.50
-- Copyright 1989 by
-- Micromint Inc.
9 June 89
Ed Nisley

Packet error! Code 00 hex
Timeout
Packet error! Code 00 hex
Timeout
Packet error! Code 00 hex
Timeout

Photo 2-The *status* **screen displays information about** MONITOR's **internal operations and the network functions.** *I* **induced some errrors** *by shorting a wire across the twistedpair to provoke the diagnostic messages.*

Many BASIC-52 programs **CALL** assembly language subroutines for bit-banging while handling all the user interface code in BASIC. While **NET** statements must be used fromBASIC, you can still invoke high-speed code on demand. In fact, your program might spend almost all its time in assembler with only **a** few BASIC state ments for net and console I/O.

Remember that the BASIC interpreter provides a convenient user interface as well as a programming language. You would have to duplicate the interface in your assembly code in order to control the program across the network; the more you think about it, the less you want to do it.

THE VISIBLE NET

I have concentrated on an "RTC52's eye" view of the network, because that is wheremost of your effort will be focused when you write your code. Because a networked application is more complex than a stand-alone controller program, you may need some help getting things sorted out. That is where MONITOR comes in.

LM311 comparator that compares a reference against the battery voltage fills the bill. When the battery falls below the reference voltage, the LM311 output changes state (either high to low or low to high, depending on your circuitry) and your CPU starts saving things.

The frick is that you've got to be able to correlate the battery voltage with remaining charge. The voltage varies with ambient temperature, as does the available charge, so fhis isn't quite as simple as it sounds. We don't recall the parf number offhand, but we think National Semiconductor makes a battery-voltage monitor IC that handles some of the grisly details. Once you get the data books you can take a walk through fhe "building blocks" section and find something useful.

You can get rapid answers to questions like these by signing onto fhe Circuit Cellar BBS. There are a bunch of redly competent folks hanging out there who can probably answer any question you can dream up. Best of all, the whole group can suggest approaches you haven't thought of and describe all the details based on real experience.

A Problem of Power

I have a car restoration shop. My problem is that my CAD system (AT clone running at 10 MHz) **sends out beeps and** stops when we use our plasma cutter, which has no HF to start its arc.

Our telephone system, a genuine Bell **setup, lights up all six buttons and won't ring when our** HF piggyback arc stabilizer **runs when we're using the heliarc welder.**

Is the fix as simple as adding ground wires? Or should I build a screen room around the welder/cutter?

How **about this as a project for your** magazine— **"Clean Power in an Ugly World"?**

Robert J. Schumann
Kansas City, MO

Dear Robert,

Every now and again we get a letter that reminds us of just how odd things gef out in fhe real world.. .

The hash in your AT and phone system is probably coming through the power lines. It could be radiated (we bet you don't have any background music playing while you're welding, do you?), but we think the place to start is wifh the line cords.

Any electric arc will generate energy across the entire electromagnetic spectrum, quite literally from DC to daylight in the case of a welder. Naturally enough, the more power in the arc, the more power shows up as interference. What you need is a filter to remove fhe objecfionable frequencies while letting the arc burn normally. From our experience with EMI (electromagnetic interference) generated by computer systems, a ferrite filter is fhe way to go.

Ferrite filters are made up of a finely divided iron compound with a high resistance to electrical current. A current-carrying wire passing through a ferrite slug induces a current in the ferrite, but the ferrife's resistance dissipates the energy veywell. The resistance increases with frequency, so the slug forms a quite effective low-pass filter.

One catch is that the ferrite slug must be able to handle the induced currenf without saturating. In the case of your welders and cutters, that's going to take a pretty big chunk! Rather fhan paying real money for this project, take a trip to the local junk yard and scavenge the yokes from a couple of TV sets—the older, the better. If you've never rummaged around in a TV before, what you're looking for is the deflection hardware around the neck of the picture tube. Along with all the coils is a big hunk of black ferrite, which is jusf what you wan t. You may have to break the tube togef the yoke off, but that's why you do this trick in the junk yard instead of Sears.

Do be careful, though, because the implosion resulting from shattering the picture tube can blow glass all over the neighborhood. Wear a face shield and gloves. The safest method is to put the neck of fhe tube in a plastic bag and rap it with a screwdriver. This presumes, of course, that the junk yard has no further interest in the tube!

With a few yokes in hand, simply string the power lines through them. If the power lines go directly to a junction box on the wall, run the cables on the secondary side through the yokes. The key point is that the ferrite slug should form a continuous path around the current carrying wire, with one wire per yoke. Don't bother wrapping fhe wire around the yoke, because one "turn" is enough; more furns simply builds a step-down transformer with a single-turn shorted secondary.

You should see an immediate improvement, but if nof, try moving fhe yokes closer to the workpiece. Because the arcs are the source of the hash, the less wire carrying the current, the less interference will be induced elsewhere in your building.

As an alternative, try putting a yoke or two on the power lines leading to the AT and phone system. After all, if doesn't matter where you filter the hash ouf as long as it doesn't gef to fhe circuity. You will have to filter all of the power lines leading to the AT; don't forget the printer and modem!

We've been thinking of doing an article on RFI control and your letter has pushed us over the edge. It'll probably show up around the end of 1989, simply because we've got so much other stuff to do between now and then.

IRS

204 Very Useful
205 Moderately Useful
206 Not Useful

In Visible INK, the Circuit Cellar Research Staff answers microcomputing questions from the readership. The representative questions are published each month as space permits. Send your inquiries to:

INK Research Staff
c/o Circuit Cellar INK
Box 772
Vernon, CT 06066

All letters and photos become the property of CCINK and cannot be returned.

PRODUCT REVIEWS:
The Next Generation
Circuit Cellar INK se//s out and enjoys ii!

It's not often that one is witness to a major event in human history, but if you're reading this, you're not only a witness, but an eyewitness! Yes, you see before you an occurrence of epoch-changing proportions. Mere human inventions pale in significance before this. Polio vaccines, atomic energy production, and those little plastic things that slowly slither down your walls all fade away when compared to what you hold before you.

What am I talking about? Why the Circuit Cellar INK Benchmarking, Product Evaluation, and Junk-Food Consumption Testing Facility.

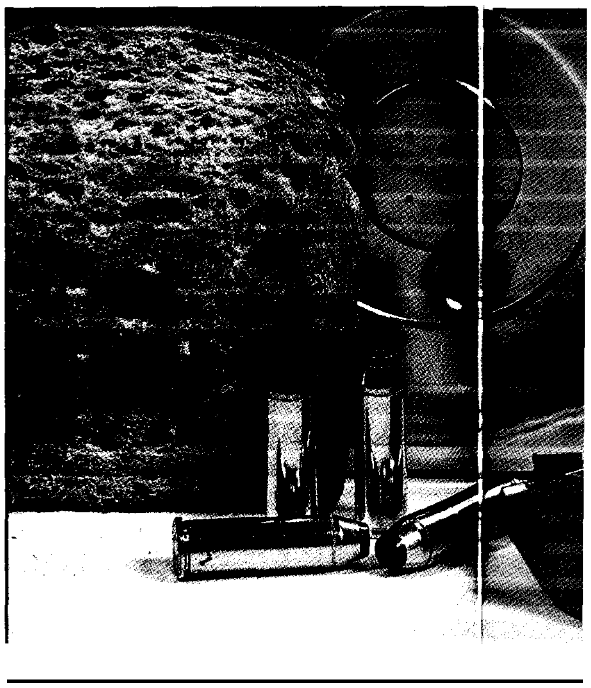
Now, I know we said that you wouldn't see any "me-too" reviews or evaluations in Circuit Cellar INK, but hey, times change. First, we saw what was being passed off as product reviews by other magazines and realized that no publication in this end of the universe wasbetter qualified to write technical, hard-hitting reviews than were we. Second, we were getting miffed that companies weren't sending neat toys (free of charge) to our offices. Third, it seemed like a good way to get paid for breaking stuff.

No Dweebs Here

At Circuit Cellar INK, when we decide to do something, we don't go in for half-measures. Before we opened the CCINK BPE&JFCTF, we performed an exhaustive study of what "the other guys" were doing. What did we find? Wimpy software benchmarks and effete instrumentation, that's what! Where were the calisthenics for individual instructions, the particle accelerators, the blast craters? They were not to be found, at least not until now.

We have assembled the only testing facility of its type in the free world. On the hardware side, we put together a state-of-the-art laboratory filled with expensive gadgets, heavy industrial machinery, and scary electronic stuff. For wringing out the software, we contracted with a little-known hacker/genius to write assembly code that's so compactly written, so immune to optimization, and. so thorough in its exercising of undocumented features, that we're not sure what the hell it really does. Finally, and most importantly, we've assembled the strangest group of gonzo testing weenies ever to don lab coats. These guys live to trash computers, and subsist on Chinese take-out, Chee•tos, and Jolt soda. Is our crew qualified, or what?

Let the Games Begin



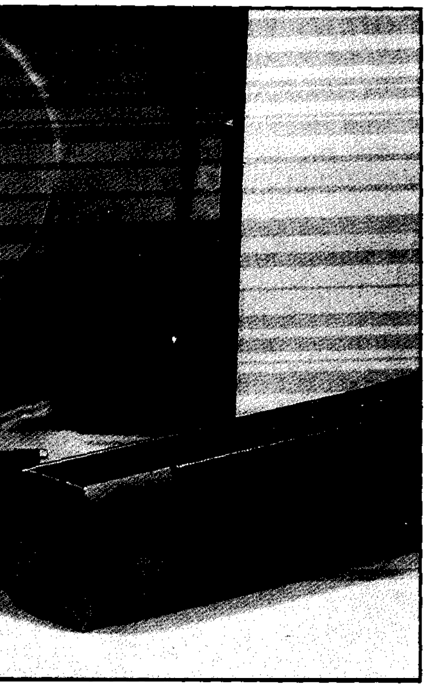
As we began looking through the literature on evaluating systems, we worked our way through the Whetstones, Dhrystones, Savages, Livermore Loops, and many more. We found that none of them really told volume purchase influencers what they needed to know about the system at hand. We thought about presenting all of the tests in an easy-to-read chart format so that readerscould form their own opinions based on the information presented. Then we came to our senses and realized that what readers really want is to be led by the nose, bludgeoned by a single sledgehammer factoid that they can point to in selfdefense when things fall apart. If it's a flattering number that companies can use in advertising to justify their high prices, all the better.

With all of the above in mind, we formed our "Magic Number Search Committee." We knew that the most important piece of the puzzle was the name. After hours of research, pages of dictionary work, and several intense rounds of Balderdash, we found The Word. Henceforth, when systems of any type are compared, they will be compared in our devastatingly elegant unit of measure, the Mopoke. The range of Mopoke is from -53 to 177, using a logarithmic scale. Using the logarithmic scale makes for snappier graphics and often makes many mediocre products look far better than they truly are.

Inside the Mopoke

Where most benchmarking programs content themselves with simple performance measurements, you must understand that the Mopoke is a unit for measuring far more than mere performance. In deciding to use the Mopoke, with all its subtle complexities, we acknowledge that most users are interested in more than how fast the processor can idle, or how many bits can be blasted to the disk per second. No, most users really want, and Circuit Cellar INK alone is now able to offer, an objective evaluation of the true gestalt of the system.

We start, of course, with performance. The first stage of Mopoke evaluation is to individually exercise every single instruction possibleina given system. Each instruction is performed, with no setup, overhead, environment, or operator intervention, for one million iterations. The time for each individual iteration (timed and recorded by a special nearly noninterventionist external CCINK BPE&JFCTF clock that is linked by special optical links to the Atomic Clock at the National Bureau of Standards) is logged, and the results are averaged using a geometric mean. Next, polar Fourier transforms are performed on the total data set at 5-degree intervals. These results, along with the geometric mean from the first evaluation, become fodder for the Mopoke cannon.



Next, we fearlessly dive into the human factors morass. Our dedicated staff painstakingly measures keyboard feel (key travel, key cap size, depth at roll-over, angle of attack, height of the little bumps on the home keys), display ergonomics (dot size, dot pitch, glare, aspect ratio, jitter, swim, bugaloo, accessibility of controls, versatility of inputs, and how hard it is to sneak the thing home to use with your VCR), and main unit construction (footprint, depth, height, hat size, presence of nifty LED displays, decibel level of the fan, pitch of the fan, aesthetic considerations). We then go where no review has gone before and quantify what can only be called "Ego Appeal Factor" (EAF). The EAF takes into account important issues such as: Will this system make your boss jealous? Does the color of the system complement your power ties? Do the noises of start-up guarantee that everyone in the office knows when you fire that sucker up? We run the results from all of these scientific tests through a weighted averaging system, where weights are determined using a complex system based on the researcher's biorhythm, the hourly exchange rate between the U.S. dollar and the Greek drachma, and the Solunar Table published in *Field and Stream.*

We don't overlook reliability in the Mopoke, either. We are the only publication that runs each and every test unit through both a steel annealing oven and a cryogenic life-extensionchamber to check for continued operation at temperature extremes. We also operate each system in booth **#7** at the "Tans for the Memories" tanning salon and at the bottom of a washtub filled with Evian water to test for susceptibility to damage from humidity **extremes.** Our line-noise isolation test includes operating the system on the same line as a heliarc welder, and injecting a signal taken from side B of Def Leppard's most recent single into the test AC line.

As thorough as are these tests, we recognized that most failures involve hard disks. To stress-check hard disks of portable computers, we bolt the little monsters to the main oscillating mixer down at "Merle's Paint'N'Plaster World." Desktop units are subjected to a patent-pending procedure we like to call the "Extreme Prejudice Test." Suffice it to say that this rigorous examination of Winchester technology involves expensive disk drives, high-velocity ammunition, and a very low "Pass" ratio.

Now, most magazines would stop right there, but we're committed to press on. We know that most users are far more afraid of their **system's manufacturer** failing than of the system itself failing. So, in an action unprecedented in computer journalism, we rate the reliability of the manufacturer. First, we use the standard ratings based on information from Standard & Poor, Dun & Bradstreet, and the men's room attendant at the New York Stock Exchange. Next, we test based on "look and feel" issues such as color (or presence) of the CEO's hair, number of company executives wearing brown shoes with blue suits, glossiness of the annual report, and quantity of shrimp served at the company's COMDEX party. Finally, we take a hard look at marketing expertise including public relations budget (did they bribe us?), advertising budget (do they advertise with us?), and use of inappropriate celebrities, water fowl, or hors d'oeuvres in their ad campaign.

As with the other categories of data, company reliability information is not left to peacefully ferment. No, we run the raw numbers through a data colander unmatched in subtletey and precision, The results are factored with numbers taken from deep between the lines of the *Wall* Street *Journal, New* York Times, and Daily Racing *Form.* When we're through, you're presented with the one number that can indicate beyond a shadow of a doubt whether you should place your trust in a company by purchasing its product. Several Wall Street heavies have come to us begging **for** our numbers, but we have **steadfastly** refused. You see, we value the welfare of our readers far more than we value the paltry few million dollars offered for our secrets. Knowing that you will be able to make major purchasing decisions based upon what you read here is worth more to us than miles of yachts or buckets of caviar.

So now you know the why and the wherefore of our review process. As you turn the page, do so with the proper reverence, for you are truly taking part in a new era: The Age of MOPOKE. ❑

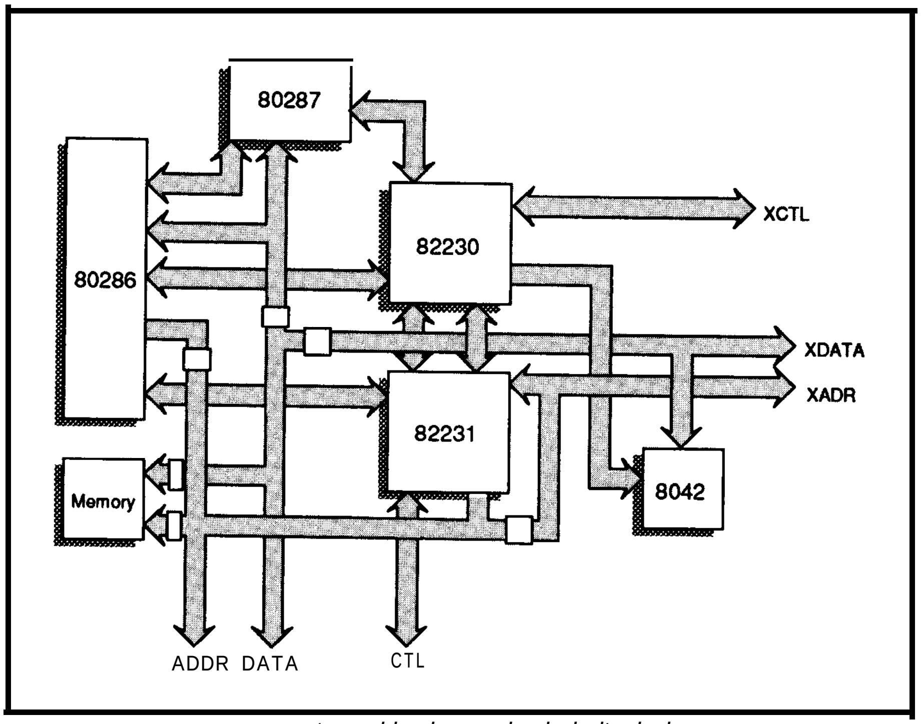


Figure 1—A PC/AT built using custom chips is very basic in its design.

Three main differences distinguish the 386SX from the 386DX:

1. The 386SX has a 16-bit external data bus; the 386DX has a 32-bit external data bus.
2. The 386SX has a 24-bit external address bus; the 386DX has a 32-bit external address bus.
3. The 386SX is available in 16 MHz only; the 386DX is available in 16-, 20-, 25-, and 33-MHz versions.

Internally, these two chips function identically. They both have 32-bit internal data and address paths and they are object-code compatible with one another. The external data bus difference allows the 386SX to use less expensive support and peripheral hardware than that required by the 386DX.

Software based on the 386 offers many advantages. Possibly the most important is the ability to efficiently multitask. The on-chip memory

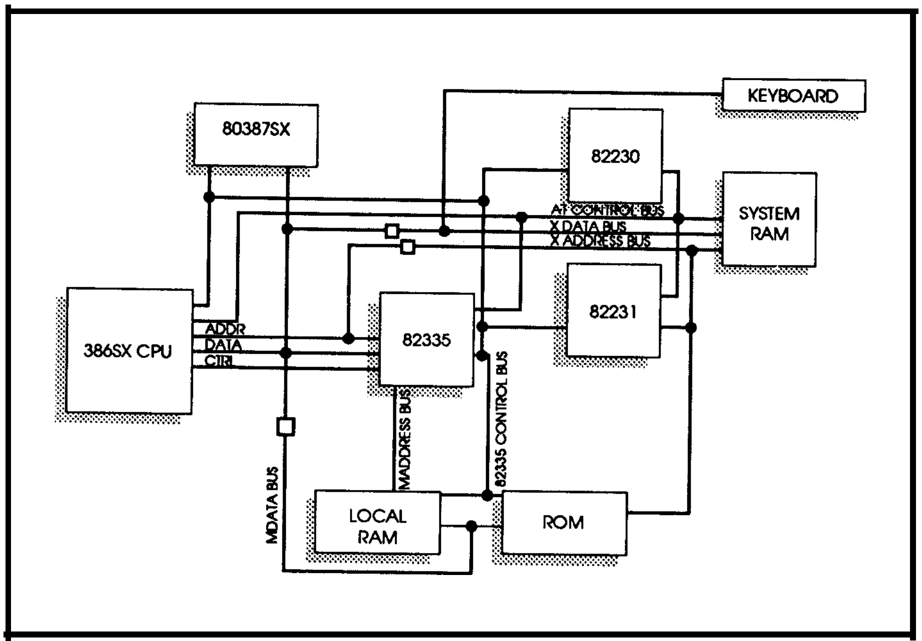


Figure 2— The block diagram for the Terminator looks very similar to the diagram above.

DEVELOPMENT TOOLS

For the IBM PC and Compatibles

2764.87664 ROM EMULATOR

Develop and test ROM code without burning EPROMs

- Surface-mount technology makes entire emulator ***only slightly larger than an actual 2764***
- Plugs into target ROM socket and connects to PC parallel port via modular telephone cable
- Accepts 8K x 8 SRAM or EEPROM (non-volatile)
- I-state reset restarts target after downloading
- Uses HC/HCT logic for CMOS & TTL compatibility
- Loads Intel Hex, Motorola S, hex, and binary files
- Command line software can be run from batch files for automatic downloading after assembly
- Adapter for 87C64 available in November. $49

$129	$149	$159
(without memory)	(with 100ns SRAM)	(with 250ns EEPROM)

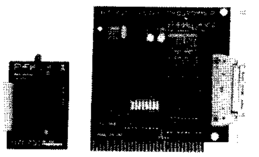
8051 FAMILYASSEMBLER

- Works on all 8051 derivatives
- Supports standard Intel syntax
- Allows local labels and include files
- Labels may be up to 32 characters
- Generates assembly listings
- Outputs Intel Hex

$95

Quick . Dependable . Clean Operation

876751 DEVELOPMENT PACKAGE



87C751 Programmer & 3051 Family Assembler

- Programs 87C751&87C752 microcontrollers
- Handles EPROM, encryption key, and security bits
- Loads and saves Intel Hex, binary, hex file formats
- Fully complies with Signetics specifications

De". **Package-DIP**	**$259**	87C751 DIP adapter	$39
(includes choice of DIP adapter)		87C751 PLCC adapter	$70
Dev. Package - PLCC	**$299**	87C752 DIP adapter	$39
(includes choice of PLCC adapter)		87C752 PLCC adapter	$79

PARALLAX INC

(916) 721-8217

Parallax, Inc.
6200 Desimone Lane, #69A
Citrus Heights, CA 95621

VISA
MasterCard

California residents add 6.5% sales tax
Shipping: No charge for UPS ground,
$7.00 for UPS 2nd day, $15.00 for UPS next day

Reader Service #146

management, large virtual address space (64 terabytes), hardware-enforced protection, and hardware-supported task switching are the fundamentalbuildingblocksofmultitasking software. The 386 also has Virtual Machine capabilities which allow several different applications to run independently, all controlled by a master application program or operating system.

Software for the 386 also takes advantage of the 386's 32-bit internal data and address paths. Software written for the 286 that is recompiled or specifically rewritten for the 386's 32-bit architecture will normally see substantial performance gains. These performance gains can range from two to five times over the 286-based code.

These cost and performance figuresare therationalebehind the386SX. While users may want the power of a 386 system, they may not be ready to spend the money required for a full-blown 386DX-based machine. The 386SX allows these users to get on the "386 track" at a lower cost than was previously possible.

SYMBOL	TYPE	NAME & FUNCTION
CLK2	I	CLK2 provides fundamental timina for the 80386SX
RESET	I	RESET suspends operation and resets the 80386SX
D15—D0	I/O	Data Bus
A23—A1	0	Address Bus
W/R#	0	Write/Read distinguishes write cycles from read cycles
D/C#	0	Data/Control distinauishes data cvcles from control cvcles
M/IO#	0	Memory/IO distinguishes memory cycles from I/O cycles
LOCK#	0	Bus Lock keeps other bus masters from gaining control when active
ADS#	0	Address Status indicates valid bus cycle and address
NA#	I	Next Address requests address pipelining
READY#	I	Bus Ready terminates the bus cycle
BHE#,BLE#	0	Byte Enables show which data bytes take part in a bus cycle
HOLD	I	Bus Hold Request lets another bus master reauest control
HLDA	0	Bus Hold Acknowledge shows that the 80386SX has given up control
INTR	I	Interrupt Request tells the 80386SX to acknowledge interrupt request
NMI	I	Non-Maskable Interrupt Request
BUSY#	I	Busy signals a busy condition from a processor extension
ERROR#	I	Error sianals an error condition from a processor extension
PEREQ	I	Processor Extension Reauest tells the 80386SX to receive from 80387
N/C		No Connects should be left unconnected
vcc	I	System Power is the +5V DC supply
Vss	I	System Ground is the OV connection from which all I/O is measured

Table 1 -*Many of the signals found on the 386SX are similar to those on other* **processors**

A PC/AT-COMPATIBLE SYSTEM

Figure 1 is a block diagram of a typical 286-based PC/AT-compatible system. The "kernel" consists of an 80286 CPU, an 80287 numeric coprocessor, local memory, and the associated buses and buffers.

Outside the kernel lies the 82230/82231 chip set which contains all the PC/AT peripherals. The peripherals include DMA Controllers, Programmable Interrupt Controllers (PICs), and other suchdevices. These devices are all based on 80286/8086/8088 bus signals. Also outside the kernel are the 8042 keyboard controller, the expansion bus, and various buses and buffers.

THE TERMINATOR SYSTEM

Figure 2 is a block diagram of the 386SX-based PC/AT-compatible motherboard described in this article. We referred to this board as the 'Terminator" during its development, so I'll continue with the name here.

This system is really a part 386 and part 286 system. The 386 part of the system provides the performance. The 286 part of the system provides the compatibility.

Much of the Terminator is the same as the system in Figure 1. First of all, the expansion bus remains the same. This is an essential part of the Terminator's PC/AT compatibility. Any add-in card that plugs-in to the Terminator's expansion bus will operate just as if it were plugged-in to an IBM PC/AT.

Secondly, the PC/AT on-board peripherals have remained the same. When interacting with either the peripheralscontainedinthe82230/82231 or the 8042 keyboard controller, software will see no difference between the Terminator and a 286-based PC/AT machine.

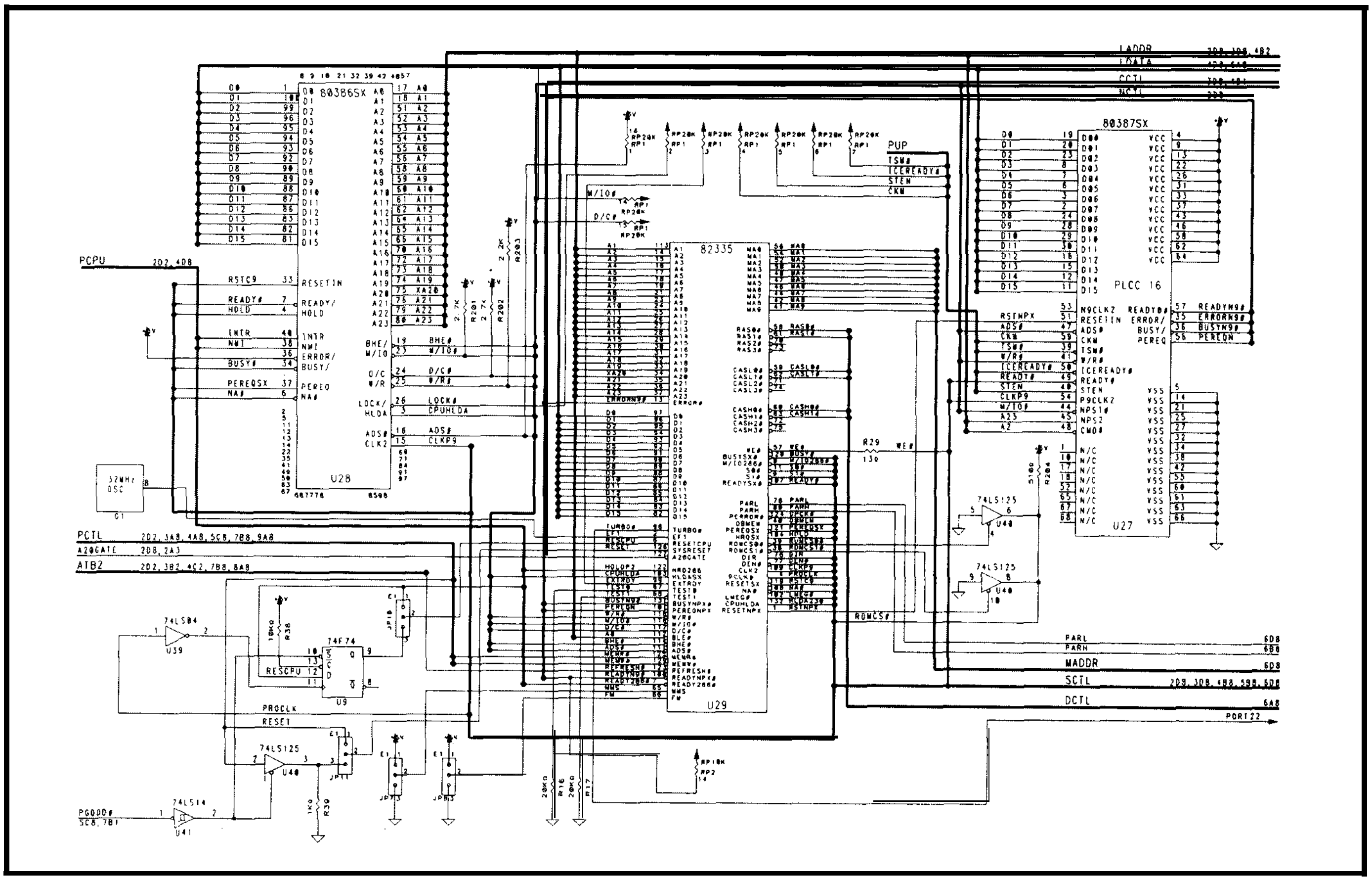
Theimportantdifferencebetween Figure 1 and the Terminator is the CPU kernel. The Terminator kernel has:

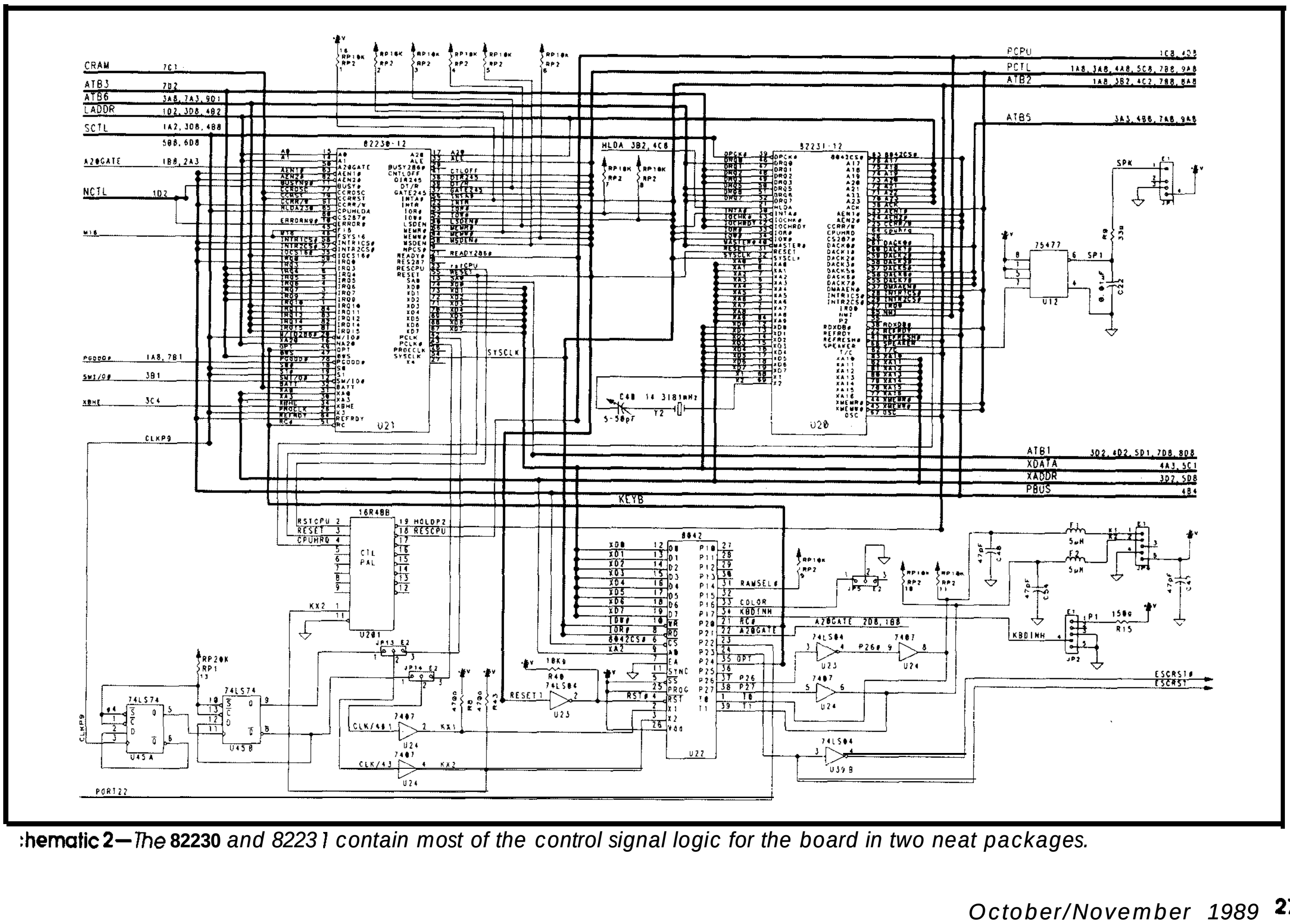
1. Replaced the 286 CPU with a 386SX CPU
2. Replaced the 287 coprocessor with a 387TMSX coprocessor
3. Changed memory systems
4. Added the 82335 interface device.

CPU KERNEL

The Terminator board is based, of course, on the 386SX. The important characteristics of the processor have already been discussed, and Table 1 gives the 386SX pin definitions.

The Terminator board uses a 387SX numeric processor extension, or coprocessor, to speed numeric performance. The 387SX fully conforms to the IEEE Floating-Point Standard #754 (1985) and is object-code compatible with Intel's 8087, 80287, and 387DX numeric coprocessors. *[Editor's Note: For a more detailed description of the Intel components, reference their respective data sheets available in the 1989 Microprocessor and Peripheral Handbook Volume 1 (Intel order #230843-006). Intel also publishes a Hardware Reference Manual, a Programmers Reference Manual, and a System Software*


Schematic 1 -The 80386.5X microprocessor makes up the core of *this PC/AT-compatible* computer motherboard.


:hematic 2—*The* **82230** *and 8223 1 contain most of the control signal logic for the board in two neat packages.*

Writer's Guide for *the* 386 *architecture. This liferafurecan beobtained bycontacting a local Intel sales office.1* It is available in a 68-pin Plastic Leaded Chip Carrier (PLCC) package.

MEMORY SYSTEM

The local DRAM resides on the memory bus. The Memory Address (MA) bus is a IO-bit (MA0–MA9) multiplexed address bus which goes from the 82335 to the local DRAMs only. The Memory Data (MD) bus is a 16-bit bus (MD0–MD15) that is connected via two 74F245s to the local data bus.

The Terminator contains 1 MB of local DRAM in the form of two banks of 256K x 9 DRAM SIMMs. Each bank is 18 bits wide (2 bytes + 2 parity bits). The DRAMs are 100-ns fast-page-modechips. Fast-page-modeDRAMs have a short page-mode cycle time which allows the 386SX to run 0-wait-state bus cycles.

When determining whether a particular DRAM is fast-page-mode, three DRAM specifications are critical:

1. *tCAS*—Column Address Strobe Pulse Width
2. *tCAC*—Column Address Strobe Access Time
3. *f CP* -Column Address Strobe Precharge Time.

To be fast page mode:

1. tCAS must be <= 35 ns
2. tCAC must be <= 35 ns
3. tCP must be <= 20 ns.

Understanding 386 bus cycles is crucial to understanding wait states. While a full explanation of the bus cycles will be given in the Part 2 of this article, a brief explanation of the 386's bus cycles and wait states will help here.

A 386SX bus cycle is made up of T-states. The first T-state of a bus cycle is called T1 and every T-state after that is called T2. Each T-state is two processor clock (32 MHz) cycles long. The first clock cycle in a T-state is phase 1 and the second is phase 2.

The 386SX will terminate a bus cycle when it receives a READY back from the accessed I/O device or memory location. If, at the end of T2 (the second T-state), the CPU has not received READY, it will keep waiting until READY is active. A two-T-state bus cycle is the fastest possible386 bus cycle. Each additional T-state required is considered a wait state. As you can see, a O-wait-state 386SX bus cycle is four processor clock cycles (two T-states) long.

The CPU will always check for READY at the end of every T2. The 82335 generates the READY for local DRAM accesses. The 82335 passes READY through to the CPU, from the coprocessor on a coprocessor cycle, or from the 82230 on a PC/AT cycle.

The 82230 automatically generates READY after one wait state for memory accesses and four wait states for I/O accesses. This default number of wait states can be dynamically overridden by use of inputs to the 82230/82231 such as 0WS\, IOCS16\, IOCHRDY\, F16, and FSYS16. I'll explain these inputs and their relation to PC/AT bus cycles further in the section on expansion bus cycles.

High Speed Data Acquisition & Arbitrary Waveform Generation

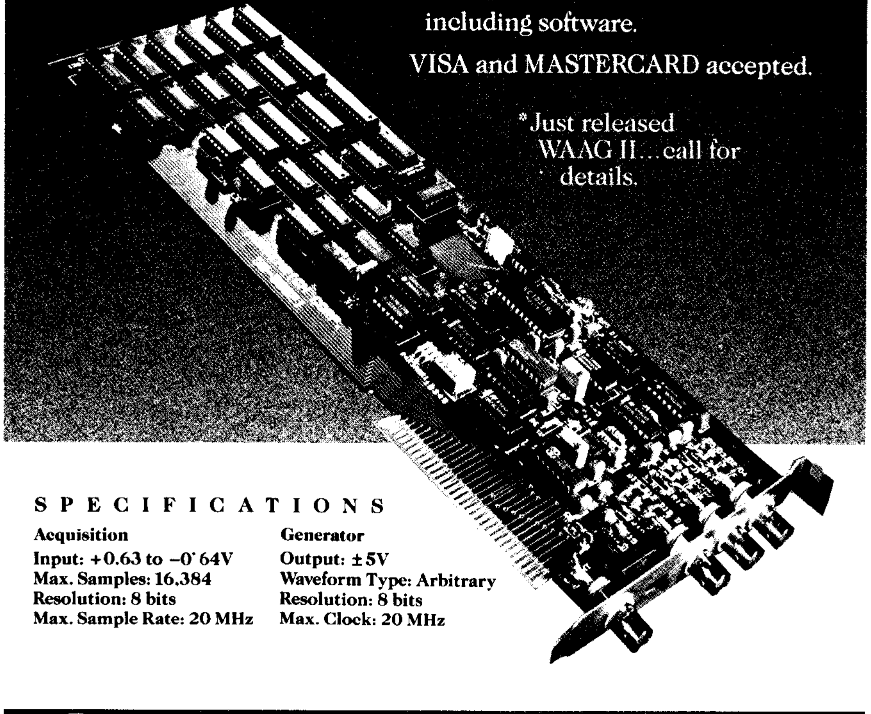
Add Waveform Acquisition and Arbitrary Generator (WAAG) capabilities to your PC or compatible. The WAAG board is a fast waveform capturing device as well as an arbitrary waveform generator on a single card. It is the ideal solution to any need requiring transient or repetitive signal capture. The WAAG...a perfect tool for laboratories, R & D facilities, medical and educational institutions and general industrial use.

PC is a registered trademark of International Business Machines Corp.

Price: **$945⁰⁰** including software.

VISA and MASTERCARD accepted.

*Just released WAAG II...call for details.



S P E C I F I C A T I O N S

Acquisition	Generator
Input: +0.63 to -0.64V	Output: ±5V
Max. Samples: 16,384	Waveform Type: Arbitrary
Resolution: 8 bits	Resolution: 8 bits
Max. Sample Rate: 20 MHz	Max. Clock: 20 MHz

MARKENRICH CORPORATION

Reader Service #136

fm	mms	Mode	Max Pages Active	Wait States					DRAM Type	Critical DRAM Specifications
				Page Hit		Page Miss		New RAS		
				Bank Hit	Bank Miss	Bank Hit	Bank Miss			
1	1	F4	4	0	0	2	2	1	100 ns Fast Page Mode	tCAS ≤ 35 ns tCAC ≤ 35 ns tCP ≤ 20 ns
1	0	F1	1	0	NA	2	1	1	100 ns Fast Page Mode	tCAS ≤ 35 ns tCAC ≤ 35 ns tCP ≤ 20 ns
0	1	WO1	4	1	0	2	2	1	All 100 ns DRAM	
0	0	wo2	4	2	0	3	3	2	120 ns DRAM	

Nofes:

1. This table assumes the following input status: TURBO # ≤ Vil max., EXTRDY ≥ Vih min.
2. The first local memory access following an idle cycle or bus cycle other than a local memory access requires one extra wait-state to *switch from non-pipelined to pipelined operation.*
3. Definitions:
New RAS-The RAS# signal for a given bank transition from an inactive to an active state.
Page hit-An access made to an active memory page.
Page miss-An access made to a memory page that is not currently active.
Bank hit-An access to the same memory bank accessed in the immediately preceding bus cycle.
Bank miss-An access to a memory bank not immediately preceded by an access to the same bank.
TCAS-Column Address Strobe pulse width
TCAC-Column Address Strobe access time.
TCP-Column Address Strobe precharge time.

Table **2—The** *82335 supports numerous DRAM configurations.*

82335

The Terminator kernel also contains an 82335. The 82335 is a high-integration interface device designed for 386SX-based PC/AT systems. The 82335 runs at 16 MHz and is available in 132-pin PQFP packaging. This device has two main functions:

1. Local memory controller.
2. Interface between the 386SX, 387SX, and the PC/AT peripherals and expansion bus.

The 82335 can operate with 1, 2, or 4 banks of memory. The 82335 operates in page-mode and interleaves the banks to improve performance.

While I mentioned "page-mode," do not confuse the 82335's paging with the 386SX's paging. The paging unit in the 386SX implements virtual memory by translating a 32-bit linear address into a 24-bit physical address. Paging in the 82335, on the other hand, is a hardware technique used to increase memory subsystem performance. Once the BIOS has programmed the 82335 with the DRAM information it needs, 82335 paging is totally transparent to software.

The 82335 puts out multiplexed addresses to the local DRAM. These consist of a row and column address. The 82335 activates a RAS (Row Address Strobe) then a CAS (Column Address Strobe) to cause the DRAMs to strobe in the row or column address that is now valid.

Memory locations sharing the same row address are said to be sharing the same page. The 82335 can keep a page in each memory bank active by keeping the RAS to that bank active. The next access to a location in that same page (a page hit) in an active bank requires only a column address and CAS pulse. No new row address and RAS pulse are needed. Page hits with 100-ns fast-page-mode DRAMs are 0-wait-state accesses. If the DRAMs are not fast enough to support fast mode, the 82335 interleaves its banks by words.

Fast page-mode DRAMs (fast mode) have a small page-mode cycle time. This means that soon after a page-mode access, another access to that same page can take place. Because of this small cycle time, successive memory accesses in the same page of the same bank can be O-wait-state. This is why the 82335 interleaves its banks by pages in fast mode.

Slower page-mode DRAMs, however, have a longer page-mode cycle time. This means that a DRAM bank will not be ready for a second consecutive access to the same page. This **access** will therefore **incur** at least a l-wait-state penalty since the 82335 must wait for the DRAM to become ready for another access. By interleaving slow memory by words, most consecutive accesses are in the same page but in a different bank, thereby avoiding the cycle time delay.

Page/bank hits/misses and DRAM speed all affect the speed of local memory accesses. Table 2 shows the wait states required for each of the various possible DRAM configurations.

The second major function of the 82335 is that of bus cycle translator. The 82335 "talks" to the 386SX and 387SX with 386 bus signals. It then translates that to 286 bus signals to "talk" to the PC/AT part of the system. This translation will be more fully described in the expansion bus cycle section. Descriptions for each of the pins on the 82335 are given in Table 3.

LOCAL BUS

The local bus connects the CPU core components. The local data bus is 16 bits wide (D0–D15). It connects the CPU to the coprocessor and the 82335. The local data bus also connects to the MD bus through two 74F245s and to the System Data (SD) bus through two 74F646 buffers. Any device on the local data bus can drive data onto it.

The local address bus is 24 bits wide (A0–A23). This bus comes from the CPU and goes to the 82335 as well as to buffers that connect to the System Address (SA) bus. A0 and A
connect to the 82230 and A17–A23 also connect to the 82231.

A0 is used by the 82230 to determine byte and word
82230 inputs A0 and outputs the buffered SAO to the system address bus.

Al is input to the 8
used in conjunction with M/IO\, SO,
S1 to detect a CPU
condition.

A17–A23 are outputs of both the 82231 and the CPU. They are tristated by the 82231 unless HLDA is active

EMBEDDED SYSTEMS DESIGNERS START USING INTEL's 8096

★★★NOW★★★

COMPLETE DESIGNERS TOOL-KIT UNDER

$500.00

MATE97
Single board Computer-l 2MHz 8097 Processor, single voltage supply, RS232 serial support, 50-pin i/O header, 32K EPROM, 32K RAM

MON97 Debugger/Monitor EPROMs- uses terminal or PC, download HEX, assemble, disassemble, step, dump RAM, set breakpoints, over 40 BIOS calls available.

SMALL C-6096 Compiler

MATECOM-Terminal Emulator (IBM-PC)

Complete Developers Tool-Kit with wail mount power supply and prototyping board
$495.00

6096 Assembier (IBM-PC)
$125.00

Other tools available

Call or write today *for data* sheets *and information.*

B. G. Associates
P.O. Box 66
Bowie, MD 207150066
(301) 509-6748

Reader Service #106

Symbol	Type	Name and Function
A1—A23		Address Inputs: select DRAM address for read or write
A20GATE		Used by keyboard controller to force A20 low
ADS#		Address Status: indicates valid bus cycle definition & address from 386SX
BHE#		Byte High Enable: indicates data is being transferred on D8—D15
BLE#		Byte Low Enable: indicates data is being transferred on D0—D7
BUSYNPX#	I	Busy NP: used by the 387SX to indicate that it is busy
BUSYSX#	0	Busy SX: indicates to the 386SX that the 387SX is busy
CASH0#— CASH3#	0	Column Address Strobe (High Byte): used by the high byte of the DRAM array to latch the column address present on MAO-MA9
CASL0#— CASL3#	0	Column Address Strobe (Low Byte): used by the low byte of the DRAM array to latch the column address present on MAO-MA9
CLK2	0	Clock2: drives the 386SX and 387SX inout clocks
D/C#	I	Data/Control Select: distinguishes between data and control bus cycles
DEN#	0	Data Enable: enables data transfer between DRAM array and local data bus
DIR	0	Direction: controls the direction of data between DRAM array and local data bus
D15—D0	I/O	Data Bus: used bv the 82335 for paritv aeneration and data checkina
EFI	I	External Frequency In: driven by the external oscillator
ERROR#	I	Error: indicates when a numeric coprocessr error has occurred
EXTRDY	I	External Ready: used to insert additional wait states into local memory bus cycles
FM MMS	I I	FM and MMS are used to select DRAM operating modes
HLDA	O	Hold Acknowledge: indicates that the 386SX has relinquished control of the local bus
HLDASX	I	Hold Acknowledge SX: indicates that the 386SX has relinquished control of the local bus
HRQ286	I	CPU Hold Request Input: receives hold request signals for the 386SX
HRQSX	O	CPU Hold Request Output: drives the 386SX HOLD output
LMEGCS#	O	Lower Meg Chip Select: held low during local DRAM access to memory below 0FFFFFH
MA9—MA0	O	Multiplexed Address: provide row and column addresses for CPU or DMA access
MEMR#	I	Memory Read Command: indicates when a DMA memory read cycle is performed
MEMW#	I	Memory Write Command: indicates when a DMA memory write cycle is performed
M/QO#	I	Memory I/O Select: distinguishes between memory and I/O accesses
M/IO286#	O	Memory I/O Select 286: emulates the M/IO# output of the 80286
NA#	O	Next Address: controls the address pipelining of the 386SX
OBMEM	O	On-Board Memory: indicates local DRAM access in progress
PARH	I/O	Parity High Byte: used for the upper byte parity bit of data on the local bus
PARL	I/O	Parity Low Byte: used for the lower byte parity bit of data on the local bus
PCLK#	O	Peripheral Clock: generated by dividing the EFI input by two
PEREQNPX	I	Processor Extension Request NP: indicates that the 387SX requires a data transfer
PEREQSX	I	Processor Extension Request SX: requests data transfer to or from the 387SX
PERROR#	O	Parity Error: indicates a parity error in data from the DRAM array
RAS0#- RAS3#	0	Row Address Strobe: latch the row address present on MAO-MA9
READY286#	I	Ready 286: indicates completion of I/O bus cvcles
READYNPX#	I	Ready NP: indicates completion of 387SX bus cycles
READYSXX	0	Ready SX: indcates completion of current bus cycle to the CPU
REFRESHX	I	Refresh: notifies DRAM controller that the DRAM array requires refresh
RESETCPU	I	Reset CPU: generates the RESETSX output which resets the 386SX
RESETNPX	0	Reset NPX: drives the RESETIN pin of the 387SX
RESETSX	0	Reset SX: drives the RESET pin of the 386SX
ROMCSOX ROMCS1#	0	ROM Chip Select: support shadow RAM--select either ROMs or EPROMs during initialization
SO# S1#	0	Bus Cycle Status: initiate and define system (non-local) bus signals
SYSRESET	I	System Reset: combines with RESETCPU to generate RESETSX and RESETNPX outputs
TEST0 TEST1	I	Test Mode: used for special test modes— must connect to Vss during normal operation
TURBO#	I	Turbo Mode Select: selects 386SX native or 80286 emulation modes of operation
Vcc	—	Power Supply: 11 pins totd
Vss	—	Ground: 11 pins total
WE#	0	Write Enable: enables DRAM input for a write operation
W/R#	I	Write/Read Select: distinguishes between read and write cycles

Table 3—*The* **82335** *handles a plethora of functions including DRAM control and bus cycle translator.*

SIGNAL	TYPE	DESCRIPTION
SAO —SA19	I/O	Used to address memory and I/O devices within the system. These, in addition to LA17—LA23 allow access of up to 16Mb of memory.
LA17—LA23	I/O	Used to address memory and I/O devices within the system. They generate memory decodes for 16-bit, 1 wait-state memory cycles.
CLK	0	8 MHz system clock signal. Cycle time of 125 ns.
RESET DRV	0	Used to reset system logic at power up or low voltage.
SD0—SD15	I/O	Bus bits C- 15 for the microprocessor, memory and I/O devices. Data of D8—d15 is gated to D0—D7 for 8-bit transfers.
BALE (BUFFERED]	O	Latches valid addresses and memory decodes from the CW.
-I/O CH CK	I	Provides parity information about memorv or devices on I/O channel.
I/O CH RDY	I	Used to insert wait states. Should be held low for 2.5µs or less.
IRQ3—IRQ7 IRQ9—IRQ12 IRQ14, IRQ15		Signal the microprocessor that an I/O device needs attention. Prioritized with 9— 12.14. and 15 having highest (9 highest) and 3-7 the lowest (7 lowest) priority.
-IOR -low	I/O I/O	Instructs an I/O device to drive data onto the data bus. Instructs and I/O device to read data from the data bus.
-SMEMR -MEMR	0 I/O	Instruct memory devices to drive data onto the data bus. -SMEMR is active only when memory decode is in lowest 1 Mb of memory.
-SMEMW -MEMW	0 I/O	Instruct memory devices to store data from the data bus. -SMEMW is active only when memory decode is in lowest 1 Mb of memory.
DRQ0—DRQ3 DRQ5—DRQ7	I I	Used by peripheral device to request access to DMA service. O-3 perform 8-bit DMA transfers: 5-7 perform 16-bit transfers,
-DACK0— -DACK3 -DACK5— -DACK7	0 0	Used to acknowledge DMA requests.
AEN	o	Used to degate the CPU from the I/O channel to allow DMA transfers.
REFRESH	I/O	Indicates a refresh cycle.
T/C	0	Provides a high puke when the terminal count for any DMA channel hits.
SBHE	I/O	Inidcates transfer of dat on the upper byte of the data bus.
-MASTER	I	Used with a DRQ line to gain control of the system
-MEM CS16	I	Indicates that current data transfer is 1 wait state, 16-bit memory cycle
-I/O CS16	I	Indicates that current data transfer is 1 wait state, 16-bit I/O cycle
OSC	0	14.31818 MHaz oscillator
ows	I	Complete the current bu cycle with no more wait states

Table 4-The *PC/AT* **expansion bus includes everything** *necessary* **for** ***16-bit I/O*** **boards.**

(and MASTER\ is not active). When HLDA is active, they are the outputs of the 74LS612 memory mapper (in the 82231) and supply page information for DMA transfers. HOLD/HLDA and DMA transfers will be discussed in more detail in the HOLD bus cycle section.

All of the address lines on the local address bus come directly from the CPU with the exception of A20 which is treated differently. A20 from the CPU is called XA20 on the Terminator, however it is not part of the X address bus. It is called XA20 to differentiate it from A20 on the local bus.

XA20 from the CPU connects to the NA20 input on the 82230. The 82230 conditions NA20 with A20GATE from the 8042 keyboard controller to produce A20. This A20 output from the 82230 is A20 on the local address bus.

THE THINGS WE DO FOR COMPATIBILITY

A20 is treated differently by the 80286-based PC/AT architecture for software compatibility with the 8088-based PC/XT. The 8088 had address lines A0–A19, therefore i t had a physical address space of 1 MB. If software incremented an address at the very top of the 1-MB address space, the 8088 would "roll-over" the address when it went past the 1-MB boundary. In other words, if software incremented the address FFFFFH (A0–A19 high) by 1, the next address generated by the8088 would be 00000H (A0–A19 low). Some 8086/8088-based software actually relied on this characteristic.

The 80286 can operate in two modes: real and protected. When the 80286 is in real mode, it is objectcode compatible with the 8086/8088. Since the80286 has 24 address lines, though, A20 becomes a 1 at the 1-MB boundary when AO-A19 go to 0. When this software incremented address FFFFFH on an 80286, the next address produced would be 100000H, not OOOOOH. This software would not get back down into lower memory by incrementing a high address, and therefore would not workcorrectlyon an 80286-based machine running in real mode.

To circumvent this possible problem, the PC/AT architecture uses A20GATE low to force A20 low. Therefore, even though the CPU puts out address 100000H, thememory sees address OOOOOH. So the software is accessing the low memory it intended to access.

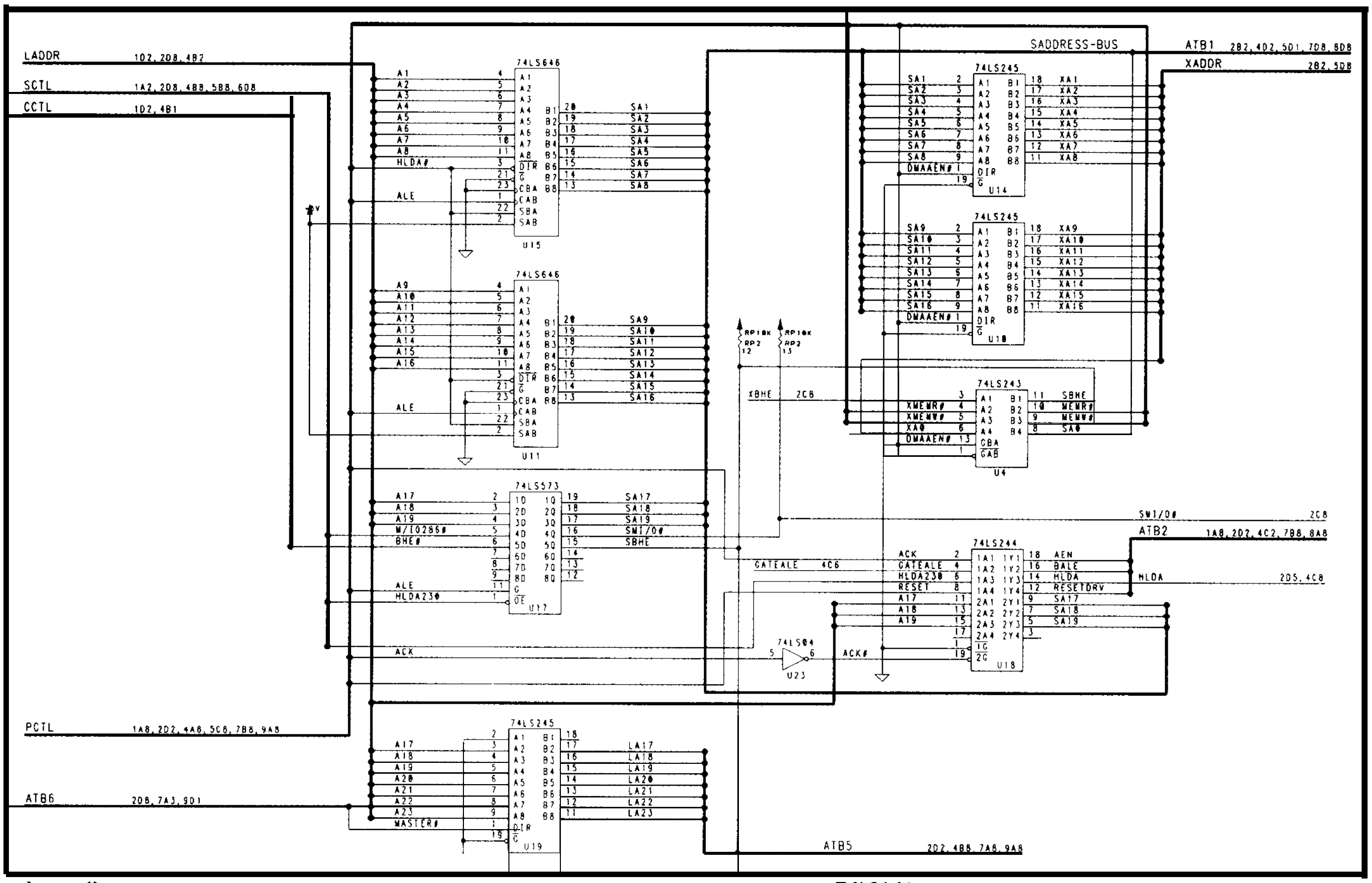
When the 80286 is in protected mode, it **no longer** has 8086/8088 object code compatibility and can therefore use all 24 bits of address space. This is why the PC/AT conditioned A20 by A20GATE, which is an output of the 8042 Programmable Keyboard Controller, instead of hard-wiringA20low. Because the PC/AT comes out of the BIOS start-up routine in real mode, the BIOS initializes the keyboard controller to bring A20GATE low thereby forcing A20 low. When a PC/AT software application wants to switch to protected mode and access memory above the 1-MB boundary, it must make a BIOS call. This BIOS call will program the keyboard controller to bring A20GATE high. This will in turn cause the 82230's A20 output to follow its NA20 input (XA20 on the Terminator), thereby removing the 1-MB boundary.

The address on the local address bus can come either from the CPU or from the system address bus through the 74LS646 transceiver/latches.

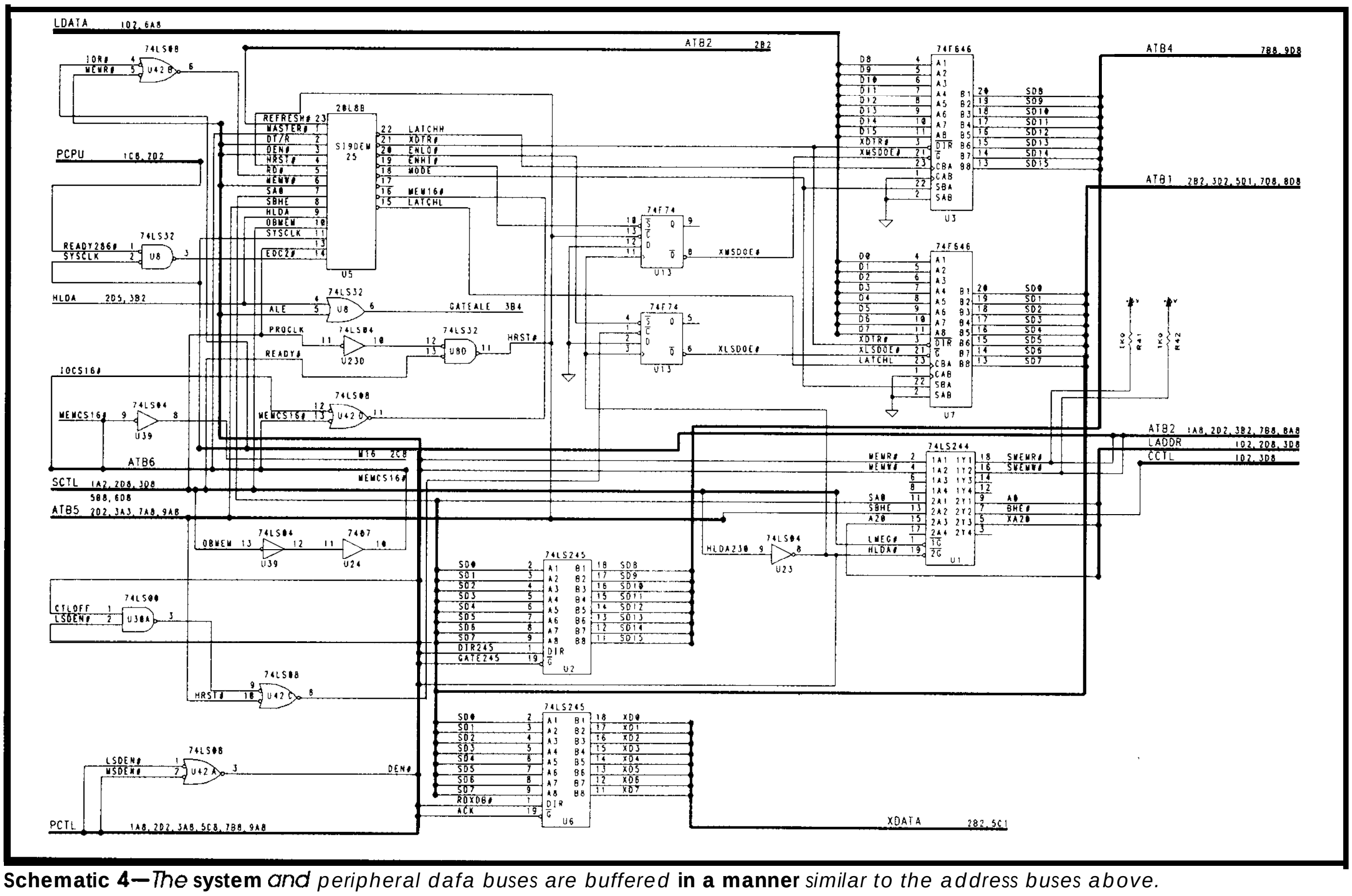
Most of the rest of the CPU and coprocessor signals stay within the CPU kernel. These signals include the bus control, bus arbitration, coprocessor signalling, and bus cycle definition signal groups.

PC/AT SYSTEM

The 82230/82231 is a two -chip set containing the PC/AT control and



chematic 3—The *system address bus is separated from the local address bus by 74LS646s. The X address bus is buffered by 74LS245s.*



Schematic 4—*The* system *and* peripheral *dafa buses are buffered* in a manner *similar to the address buses above.*

peripheral logic. The 82230/82231 contain:

- 2 8259A Programmable Interrupt Controllers (PICs)
- 2 8237 DMA Controllers
- 1 82288 Bus Controller
- 1 8254 Programmable Interval Timer
- 1 8284A Clock Generator
- 1 82284 Clock Generator and Ready Interface
- 1 6818 Real-Time Clock (RTC)
- 1 74LS612 Memory Mapper
- Miscellaneous control logic

The PC/AT schematics in the IBM PC/AT Technical Reference Manual are a good source of information about the 82230/82231 because their internal logic corresponds to those schematics. The peripherals' individual data sheets can also be referenced for operation information.

The 82230 controls PC/AT bus cycles. From the outputs of the 82335, the 82230 generates the necessary memory/IO, read/write signals to start a PC/AT bus cycle. The 82230 puts out buffer control signals at the appropriate time. Finally, the 82230 terminates the cycle by producing a READY after a certain number of wait states.

The 82230/82231 have many other functions as well. The 82230's PICs arbitrate between the many possible sources of interrupts. The 82231's DMA controllers handle DMA accesses; one example being floppy disk accesses. The 82230 handles RESET generation and also produces the PC/AT system clocks. The 82231 generates the REFRESH request and also includes the parity check logic.

EXPANSION BUS

The system address and system data buses are a part of the expansion bus. The system address bus is 20 bits wide (SA0–SA19). The system data bus is 16 bits wide (SD0–SD15). The large address bus (LA17–LA23) is an unlatched bus also included in the expansion bus.

The PC/AT expansion bus is defined in the IBM PC/AT Technical Reference Manual. The signal definitions are shown in Table 5.

PERIPHERAL BUS

The peripheral, or X, bus is the bus on which the on-board peripherals reside. The X address bus is 17 bits wide (XA0–XA16) and the X data bus is 8 bits wide (XD0-XD7). On-board peripherals in the Terminator, except for the 8042, are contained in the 82230/82231.

BUFFERS

There are various latches, transceivers, and buffers that separate the many data and address buses in the Terminator board.

The local address and data buses are separated from the system address and data buses by 74LS646s. The 646s are used for isolation, direction control, and buffering as well as for their latching capabilities. The system address bus is defined as a latched bus. Since the CPU can take the address away before the end of a bus cycle this address must be latched to remain valid throughout the cycle. A17–A19 and some control signals are passed through a 74LS573. These signals need latching but do not require bidirectional flow.

A17–A23 go through a 74LS245 to become LA17–LA23, the large address bus. Since this is defined by the PC/AT architecture to be an unlatched bus, latching capabilities are not required.

74LS245 bidirectional transceivers are used for the byte swap (explained in the expansion bus cycle section) and also between the system and X bus, and between the local and M data bus. Latching is not required in these cases either.

74LS244s are used for some unidirectional control signals and also for the RD bus, neither of which require latching.

CONTROL PAL

In position U5 there is a 20L8B PAL that controls the previously mentioned latches, transceivers, and

Heathkit®

A leader in quality electronics for the technically sophisticated customer.

When you need kit or assembled electronic products for work, home or hobby, you can be sure Heathkit products are designed to perform reliably and effectively...year after year.

See what we have to offer. To get your ***FREE* Heathkit** Catalog, fill out and mail the coupon below or call toll-free today!

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

YES! *Please send me a **FREE** copy of the Heathkit Catalog.*

Send To: Heath Company, Dept. 026-774 Benton Harbor, Michigan 49022

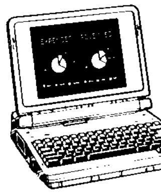
Name______________________

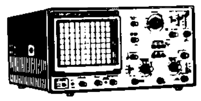
Address____________________ Apt________

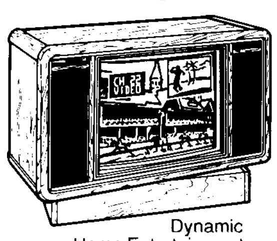
City_______________________

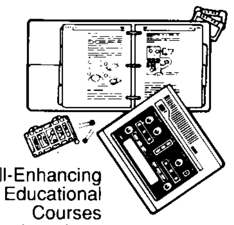
State__________ Zip__________

A subsidiary of Zenith Electronics Corporation **026-834**

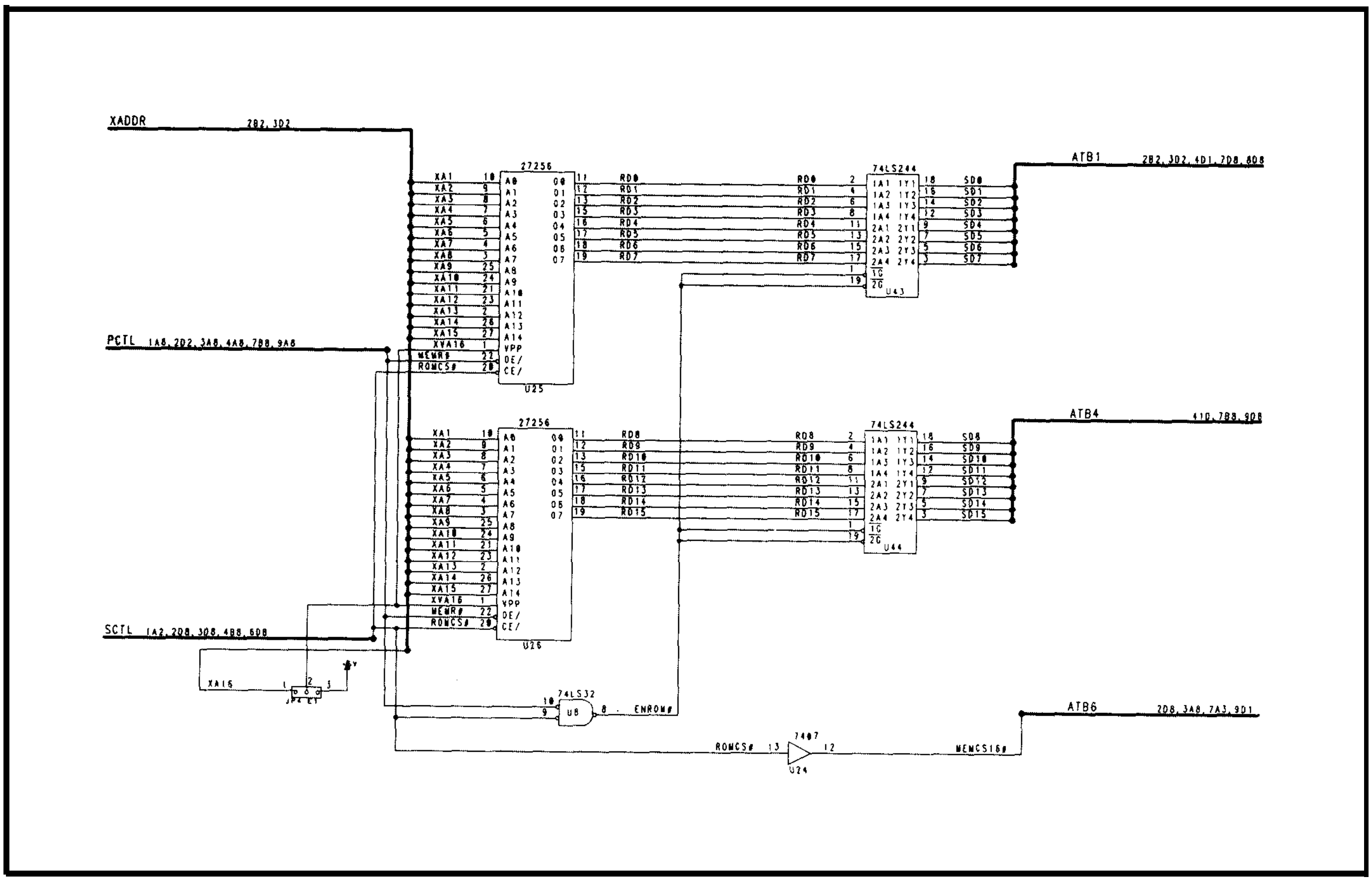




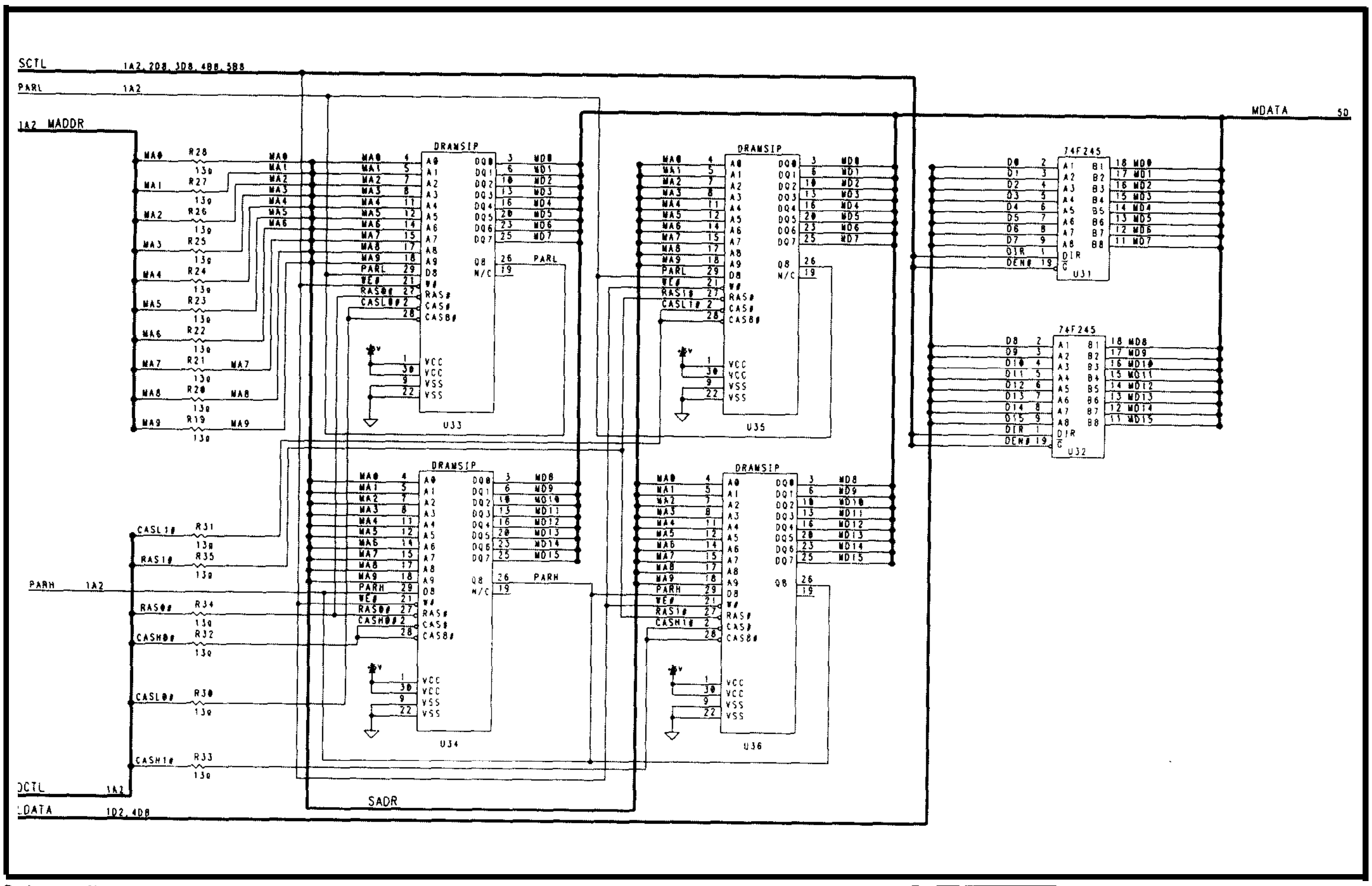




Reader Service #125



Schematic 5-A *pair* **of 27256 EPROMs provide the system with 64K bytes** *(32K* **x** *16* **bits)** *of storage for* **the BIOS.**



Schematic 6—*One megabyte of RAM is provided on the board through the use of four 256K x 9 SIMMS.*

buffers. This PAL controls the direction, enabling, and latching of these devices.

Some buffer control signals can be used directly from the 82230/82231 or the 82335. However, because this system is a hybrid of a 286 and a 386 system, the control PAL must handle the interfacebetween the two "pieces" of the system.

One example of this interface is how the control PAL deals with the local to system data buffers. In a PC/AT the upper data byte is never latched and the lower data byte is only latched on a byte swap. This is because the PC/AT can assume that the bus cycle will be complete at the end of the T-state in which READY was asserted. Due to **translation delay,** however, this is not true in the Terminator. Since the PC/AT will take away its data at the end of the T-state in which it returns READY, all 16 bits must be latched by the control PAL so that the data will be valid when the CPU is ready to read it.

Another difference is that addresses must flow from the SA bus to the local bus in the Terminator, whereasitdoes not have to in a PC/AT system. This is because the MA bus is controlled by the 82335, which is on the local bus in the Terminator, whereas the MA bus connects to the system address bus in a PC/AT.

Each term in the PAL codes is for a particular type of bus cycle. The bus cycle types are described in the Bus Cycle section in Part 2.

Hex Range	Device
000-01 F	DMA controller 1
020—03F	Interrupt controller 1, Master
040—05F	Timer
050—06F	8042
070—07F	Real-time Clock, NMI mask
080—09F	DMA page register
OAO-OBF	Interupt Controller 2
OCO-ODF	DMA controller 2
OFO	Clear Math Coproc. Busy
OF1	Reset Math Coproc.
OW-OFF	Math Coproc.
1FO—1F8	Fixed Disk
200-207	Game I/O
20C—20D	Reserved
21F	Reserved
278—27F	Parallel printer port 2
2BO—2DF	Alternate Enhanced Graphics Adapter
2E1	GPIB (Adapter 0)
2E2&2E3	Data Acquisition (Adapter 0)
2F8—2FF	Serial port 2
300-3 1 F	Prototype Card
360—363	PC Network (low address)
364-367	Reserved
368—36B	PC Network (high address)
36C—36F	Reserved
378—37F	Parallel printer port 1
380—38F	SDLC, bisynchronous 2
390-393	Cluster
3A0—3AF	Bisynchronous 1
3B0—3BF	Monochrome Display and Printer Adapter
3C0—3CF	Enhanced Graphics Adapter
3D0—3DF	Color/Graphics Monitor Adapter
3F0—3F7	Diskette controller
3F8—3FF	Serial port 1
6E2&6E3	Data Acquisition (Adapter 1)
790-793	Cluster (Adapter 1)
AE2&AE3	Data Acquisition (Adapter 2)
B90—B93	Cluster (Adapter 2)
EE2&EE3	Data Acquisition (Adapter 3)
1390—1393	Cluster (Adapter 3)
22E1	GPIB (Adapter 1)
2390-2393	Cluster (Adapter 4)
42E1	GPIB (Adapter 2)
62E 1	GPIB (Adapter 3)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)

Note: I/O *Addresses 000h—0FFh are reserved for the system board I/O. 100h—3FFh are available on the I/O channel.*

Table 5—*The PC/AT I/O* address *map* **shows how loose** *the decoding is in the I/O space.*

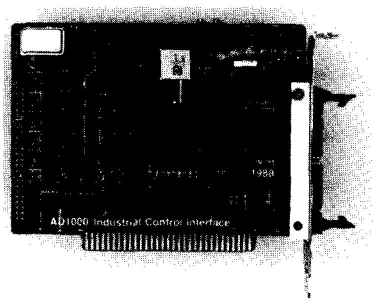
BIOS

The Terminator has 64K of system BIOS ROM in the form of two 256K-bit EPROMs located on the XA bus. The 16-bit ROM data bus (RD0-RD15) is connected to the SD bus by two 74LS244s.

The BIOS start-up routine programs the 82335's registers with memory configuration information gained from a memory autoscan. The BIOS also enables various ROM shadowing options according to information stored in the CMOS RAM. Thereare64 bytes of battery backed-up CMOS RAM in the 146818 Real-Time Clock

PC Bus Data Acquisition and Control Cards

STOP paying a small fortune for PC data acquisition and control interfaces!



AD1000 — $295

Real Time Devices, Inc. designs and manufactures the lowest cost industrial/scientific interface cards for the PC/XT/AT bus. Our commitment is to offer only high quality U.S. designed and manufactured interfaces — not cheap imports All our cards are backed by a one year warranty, 30 day NO RISK return policy, and free technical support!

AD1000 — 8 channel 12-bit 20 uS A/D; sample & hold; three 5 MHz timer/counters; 24 TTL digital I/O lines. $295

AD200 — 4 channel 12-bit 125 uS A/D; three 5 MHz timer/counter; resistor configurable gains; 24 digital I/O lines. $239

AD500 — 8 channel 12-bit integrating A/D; programmable gains of 1, 10, & 100. Extremely stable & accurate! $239

AD100 — Single channel version of AD500; optional D/A; gains of 1, 10, & 100. Plus 10 digital I/O lines. **$149/$198**

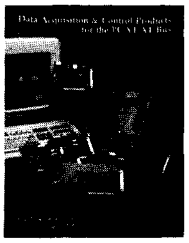
ADA300 — 8 channel B-bit 25 uS A/D; single 8-bit D/A; 24 TTL digital I/O lines. $239

DA600 — Fast settling dual/quad 12-bit D/A, internal double buffering. **$179/$239**

DG24/96 — 24/48/72/96 line TTL compatible digital I/O cards; 10 MHz, 8255.based. Optional buffers and pull-ups. **$95/$239**

TC24 — Five 5 MHz timer/counters; uses powerful Am9513 chip. Also 24 digital lines. **$189**

FREE CATALOG



CALL or WRITE today for your copy of the latest catalog and price list for our complete line of PC bus data acquisition, control, prototyping, extender cards, technical books and accessories!

OEM discounts begin at a quantity of three!

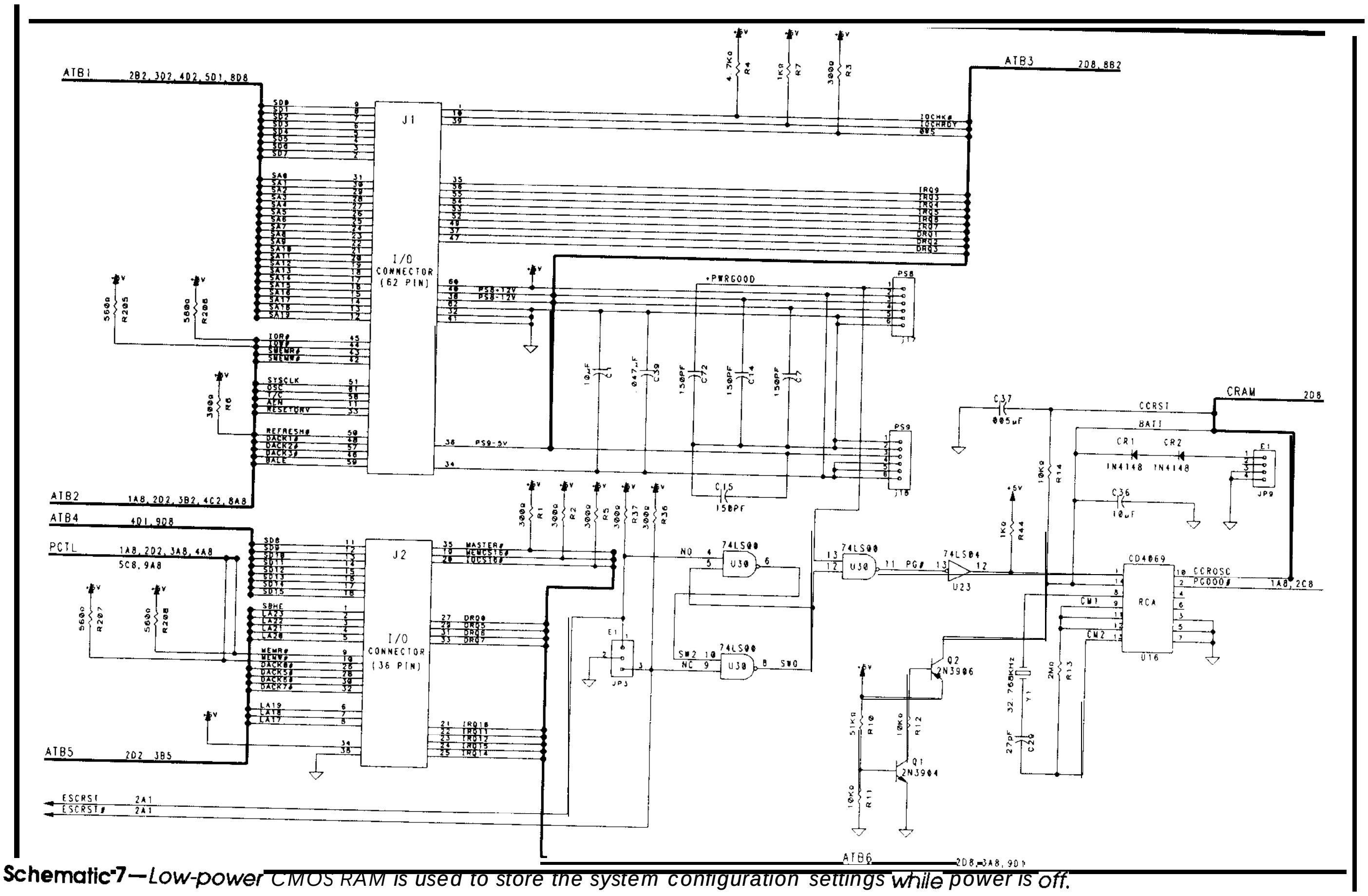
Real Time Devices, Inc.

rtd

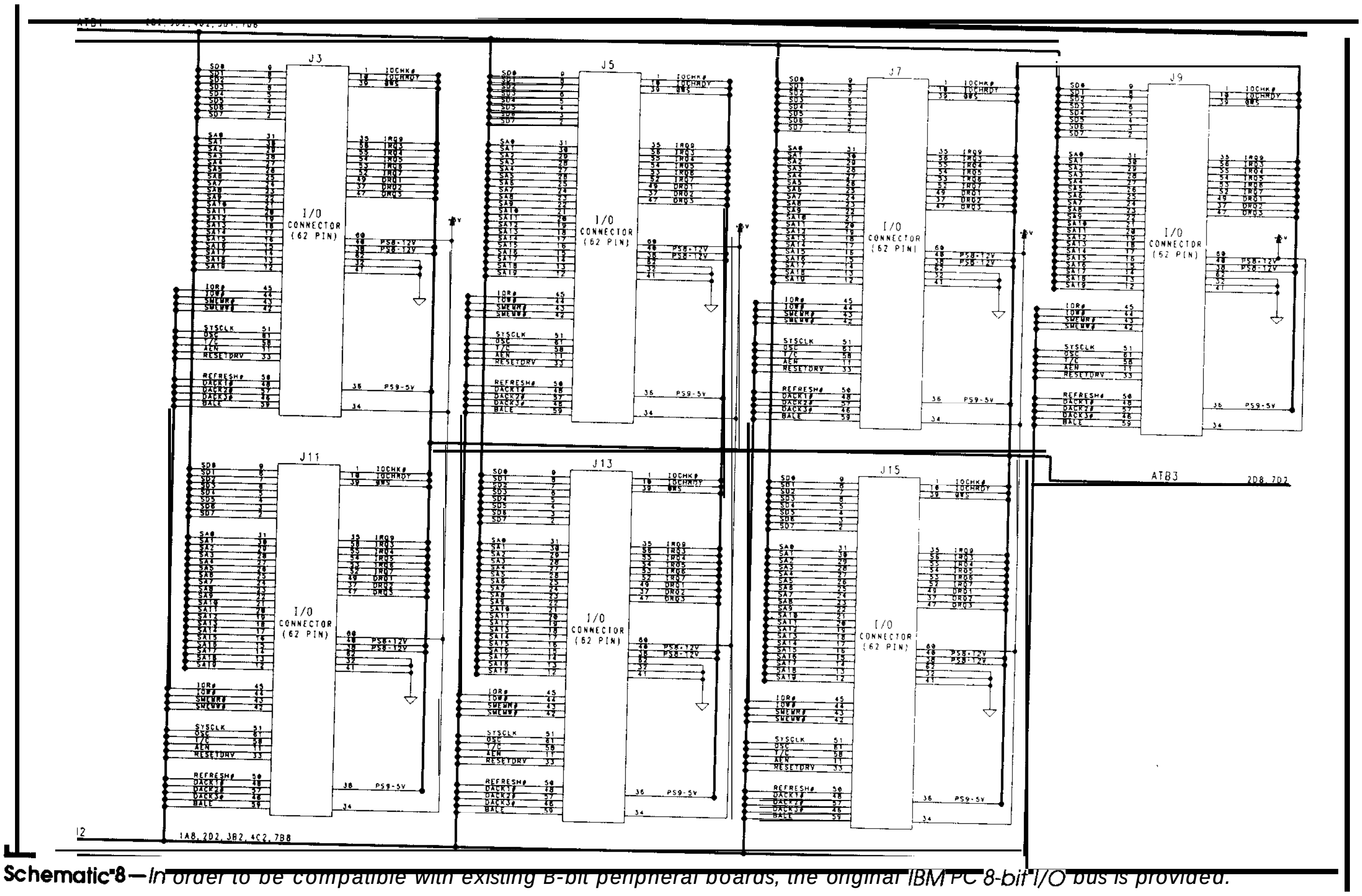
531 E. Marylyn Avenue
P.O. Box 906
State College. PA 16804 USA

814/234-8087

Reader Service #150



Schematic 7—*Low-power CMOS RAM is used to store the system configuration settings while power is off.*



Schematic 8—*In order to be compatible with existing 8-bit peripheral boards, the original IBM PC 8-bit I/O bus is provided.*

(RTC) in the 82230. This RAM is used for storing system information required at power-up. Because of the battery back-up, this information stays valid when the system is turned off.

After the 82335 is programmed, the PC/AT part of the system needs initialization. This involves programming the keyboard controller and the various 82230/82231 peripherals. This is part of the Power-On Self Test (POST).

When it is finished with the POST, the BIOS sets the LOCK bit in the 82335's configuration register. This causes the 82335's registers to become inaccessible until the next system reset. In other words these registers basically "disappear" from the Terminator's I/O address space. This is necessary to retain PC/AT compatibility since these registers are not a part of the PC/AT I/O address map. The PC/AT I/O address map is shown in Table 5.

The BIOS initialization will also set up add-in cards. The BIOS scans though the memory address space looking for an adaptor ROM "signature." This is a special bit pattern that signifies that an add-in card has its own BIOS at this location. The system BIOS will turn over control to the add-in card BIOS which then sets up the card. The add-in card BIOS is free to use system information available in the system BIOS data area and the CMOS RAM.

Lastly, the BIOS will check the disk for the operating system bootstrap program. If it is available, it will load the bootstrap and transfer control to it.

VGA

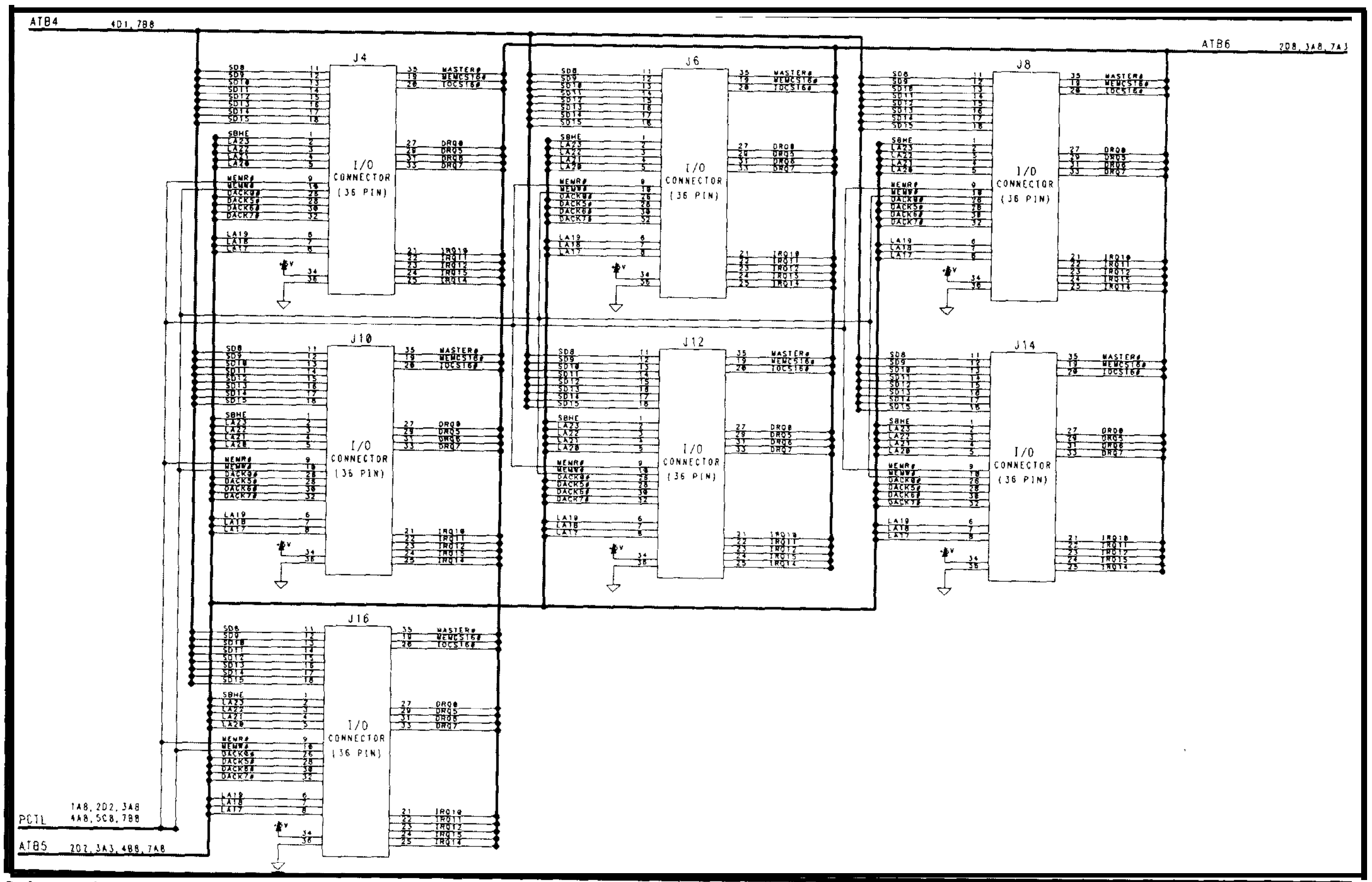
Although the VGA circuitry physically resides on the motherboard, it is logically treated as an add-in card. It interfaces with the Terminator only through the expansion bus, as would any VGA add-in card. The low chip count of the Terminator system has basically allowed us to place an Intel-designed VGA add-in card physically onto the motherboard without changing the add-in card's operation. Therefore a VGA access is considered an off-board or expansion bus access. Since it is logically not part of the Terminator system, I will not further discuss the VGA circuitry in this article.

AND ON TO THE BUS CYCLES.. .

So far I have described the basic logic of the Terminator board. In Part 2, I'll describe the Terminator's bus cycles and present the VGA portion of the schema tic. This should round out your understanding of the components and explain how they all function together as a system. ✣

Daryl Rinaldi is an application engineer at Intel Corporation and works on the 386SX and associated products. In his spare time, he likes to explore California.

IRS
207 Very Useful
208 Moderately Useful
209 Not Useful



Schematic 9—For complete PC/AT compatibility, 16-bit expansion boards are supported in addition to 8-bit boards.

FEATURE ARTICLE

Part 1

The BCCH 16

A 16-/32-bit Multitasking Single-board Computer

by Tom Cantrell

Microelectronics have certainly benefited all in the form of calculators, watches, PCs, audio, cars, and a myriad of other mass-produced items. However, an important segment of the economy-small manufacturers-have been poorly served by computer technology.

I encourage all technocrats to practice a little self-enlightenment. Make a visit to a small industrial park-you know, over there on the other side of the tracks where you never go. Don't be intimidated by the ramshackleappearance, funny smells, and loud noises. Look around. What you'll see are small outfits offering myriad mundane industrial products and manufacturing services. But you'll also be shocked to see how "low-tech" these places are.

For every industrial giant trying to decide whether to spend $1 or $2B on the latest factory automation effort, therearea **100small businesses—theinvisibleenginesof theeconomy—** who can't computerize their way out of a paper bag. What? You mean VME-based "cells" connected with a broadband fiber-optic LAN aren't the answer for Joe Smallco who needs a remote display for his truck scale? RISC, ASIC, CASE, AI, and others aren't the solution either.

The most important criteria isn't MIPS, LIPS, MOPS, or FLOPS. Indeed, manyautomatable **tasks require** minimal performance. What'sneeded are systems that are easy to use for noncomputertypes;systemsthatdon't require an army of engineers and lab full of expensive tools to get working.

We decided to design a computerized controller that would fill the needs of those who need advanced control but don't need to become experts in obscure assembler languages. We settled on a BASIC-language controller sitting on an established control bus (in our case, the BCC bus) as a general direction to follow.

BCCH 16 HARDWARE DESIGN

The goals of the BCCH16 hardware design are:

- Maximum performanceat reasonable cost
- Hardware compatibility with the BCC bus
- Software compatibility with existing BCC180 BASIC programs
- Small form factor and low-power operation

Of course, I think the BCCH16 meets the goals handily, but let me explain the design and you can judge for yourself. First, take a look at the BCCH16 block diagram (Figure 1) and then follow along as I describe each of the major subsystems: CPU, memory, I/O, and BCC bus interface.

THE H16 CPU

The Hitachi HD641016 (the "H16") is the middle member of the recently introduced "H Series" comprising the H8, H16, and H32 devices. Though the CPUs aren't object code compatible, they are pretty much assembly language upward compatible. This means assembly programs for a lower numbered chip (i.e., H8<H16<H32) will assemble and run on a higher numbered chip. The "pretty much" proviso relates to on-chip I/O devices; each existing (and future) device has a different mix of on-chip I/O. Obviously code that accesses an I/O device on one H-Series CPU will have to be reworked when ported to a different H-Series CPU that doesn't have that I/O device!

Though not object code compatible-a dubious idea at best for chips

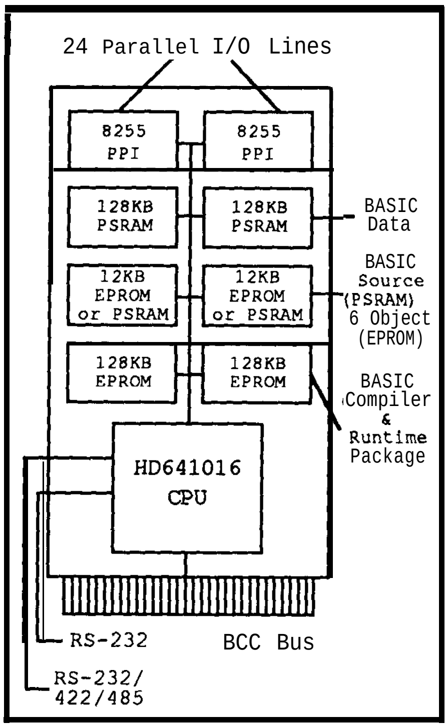


Figure 1—*The BCCH16 contains a 16/32 HD64 10 16 (H16) CPU. up to 768KB of memory (256KB* **each for system,** *program and data). and* **48** *parallel I/O lines. The CPU itselfpacks two serialports* **and (not shown)** *timers and DMA channels as well. The board talks* **to** *existing* **B-bit** *BCC-bus-compatible boards and accommodates 16-bit I/O board designs as well.*

that span the application spectrum from phones and VCRs to workstations and multi-FPU image processors-the members of the family do sharesomecommonarchitecturaland implementation aspects.

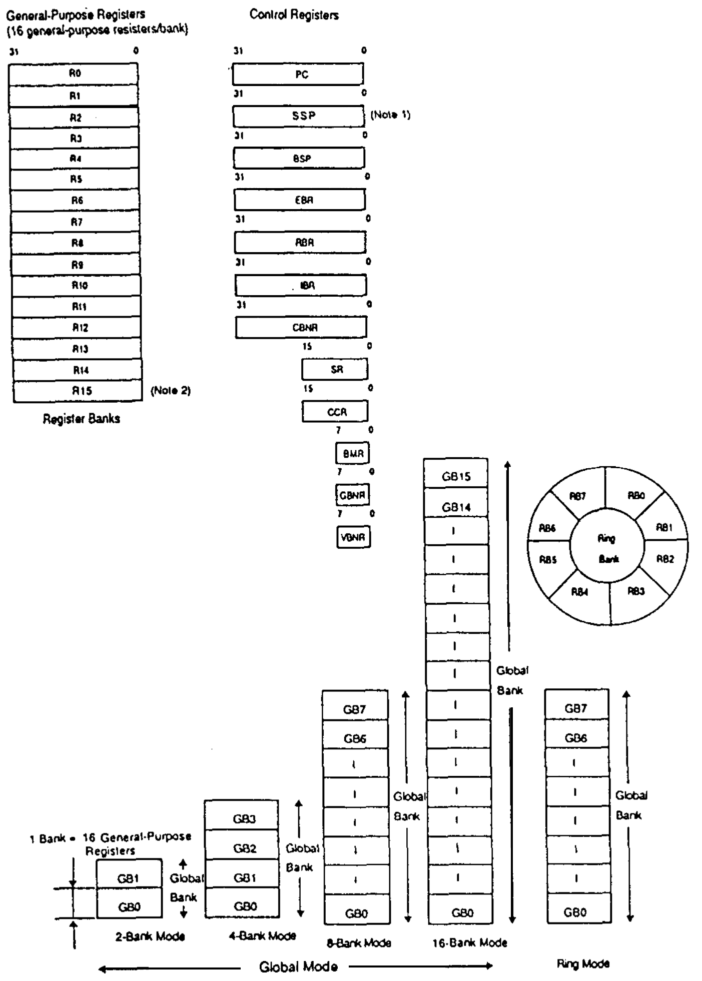
General-Purpose Registers

The machines have lots of registers. The **H16** has 16 32-bit registers, RO to R15, shown in Figure 2 (ignore all the other stuff in the figure for **now)** which all work exactly the same. This is far different than **CPUs** such as the Intel "86" family in which registers are limited to special functions and even the Motorola *68xxx* family, which restricts some registers for addresses and some for data. The only thing to remember about H16 registers is that R15 serves as the stack pointer (actually, R15 is general purpose as **well—** you can hit it with any instruction you want, but be careful!).

Regular Instruction Set

Without getting into the "ISC" (RISC, CISC, CRISC, etc.) religious wars the story on the H-Series is as follows:

- Few basic instructions-MOV, ADD, BRA, AND, etc.
- Lots of addressing modes and data types.
- The instruction set is very orthogonal to the registers, addressing modes, and data types.
- Unique special-purpose instructions and data types for different H-Seriesdevices that target particular applications. For instance, the H16 has bit-field instructions useful in multichannel on / off (relays, switches, lights, etc.) control and bitmap CRTs / printers while the H32 has multi-FPU coprocessor support.
- High code efficiency (i.e., bang perbyte). **For** instance, theH16, unlike some other **16/32-bit CPUs,** supports byte-aligned **instructions(allowing1-,3-,etc.** byte instructions where the older chips would have to use 2, or multiples thereof, bytes) and features "short-forms" for commonly used instructions (an **examplebeing special** short BEQandBNE instructions, since they are used so often, as well as a longer "universal" conditional branch form).



Notes: 1. If R15 or slack pointer is selected by an instruction when the S bit of the SR regisler is set to 1, the supervisor slack poinler (SSP) is used.

2. If the S bit of the SR register is cleared lo 0, the user stack pointer (USP) which is R15 of general-purpose register is used.

Figure 2—*The* programmers view of *the H16* **primarily consists of** 16 *32-bit registers (R0-R15,* **top** ***left).*** *A number of the other controlregisters (top right) are related to the register bank modes (global and ring, bottom).*

High-speed on-chip memory

This takes different forms: code EPROM on the **H8,** register banks / data RAM on the H16 and, a myriad of caches on the H32.

Advanced CMOS process

Very fast and surprisingly low power-less than 100 **mA** at 10 MHz, and half that when you exploit programmable low-power modes.

REGISTER BANKS

As mentioned previously, one of the precepts of the H-Series architecture is to exploit high-speed on-chip memory to speed overall performance. In the case of the H16, this takes the **formof 1K bytesof on-chip RAM.** The RAM is "fast" in two ways. First, on-

ZIP

Image Processing for ImageWise Video Digitizers

ZIP allows you to do more with video images than you ever thought possible

Live video on VGA screen

With the **ImageWise/PC** there's no need for an external monitor -- ZIP lets you use the VGA at 320 x 200 and view live video. Automatically adjust video camera for available light.

Advanced image processing functions

Process **single+multiple** images with math and logic functions, matrix convolution, equalization, outline, sharpen, and more.

Create color images

Use a video camera with separate RGB outputs, or use color filters with any video camera and ZIP creates full color images for display on VGA.

ZIP, versions for serial **ImageWise** and **ImageWise/PC**, include printer support and PCX, TIFF, and MAC file formats.

ZIP Image Processing Software $79

Now *get even more from ImageWise digitizers*

Statistics, conversion to **PIW-PIF-PCX-ASC-WKS-TIFF-EPS-GIF**, presentation software, and a VGA palette changer -- all in the ZIP Utility Pack $39.

HOGWARE COMPANY
470 BELLEVIEW
ST LOUIS MO 63119
(314) 962-7633

ZIP requires 384K RAM, MSDOS 2 or higher. Display requires EGA or VGA. ZIP trademark of HOGWARE. IMAGEWISE trademark of Circuit Cellar Inc.

Reader Service #126

chip RAM accesses take only two clocks while external memory accesses always take at least three and possibly more depending on the clock rate and whether you can afford the fastest memory chips. Second, in line with the H16's 32-bit software architecture (funneled through a 16-bit external bus), the 1K bytes of RAM are organized as 256 32-bit 'long words." Thus, for 32-bit data, on-chip accesses are at least three times faster than off-chip!

It's pretty **obvious** that computers spend most of their time accessing memory in one way or another (fetching instructions, loading/storing data, pushing/popping the stack). Thus, it isn't a major intellectual breakthrough (RISC hype notwithstanding) to realize that faster memory is better (even the most uninitiated know "wait states" are bad). Of course, the best way to make memory faster is to put it on the same chip as the CPU.

The more interesting question is what form the memory should take—instruction memory, data memory, or cache. Yes, faster memory is always good, but the best way to access it depends on the application environment. Remember, the H16 is optimized to serve demanding real-time interrupt-driven applications.

These complex applications for which the H16 is targeted—laser printers, robotics, high-speed data acquisition and process monitoring, etc.—are unlikely to fit in the 4K–16K bytes typically offered on an 8-bit single-chip MCU. So, the H16 doesn't include any on-chip code (EP)ROM at all. Many applications wouldn't fit in any feasible amount of on-chip memory and, for large programs, commodity memory chips are more economical anyway. This rules out instruction memory as a speed-up option.

The other option is cache for instructions and/or data. For general-purpose computers, the cache approach is a time-proven winner. Indeed, all the new 32-bit PC/workstation chips exploit it heavily (cache wars?). But, the gotcha is I said "general purpose" computers. In fact (despite the claims of those hoping to reposition their UNIX CPUs as "embedded controller" chips), cache is not a very good approach for "real-time" computers, and here's why.

The idea of cache is to exploit the concept of "locality" which postulates that program and data accesses are not random, but are "clustered" in both time (sequence) and space (memory address). (Note: "Locality" is also the underpinnings of virtual memory and its "working set" but so far, thankfully, I haven't heard anyone try to explain why VM is needed in controllers). A "data-processing" type program tends to proceed in a basically sequential way while certain routines and data structures comprise most of the activity. For instance, the Macintosh I'm writing this on spends most of its time in a loop waiting for me to think of something to type.

However, real-time or interrupt-driven systems are an entirely different beast. Instead of executing a single routine, external events send CPU program and data accesses all over the place from microsecond to microsecond. Where you are now predicts little about where you'll be during the next cycle.

Now, what do you think happens to cache when faced with, say, a dozen interrupt handlers (sorry, they all won't fit in the cache) and hundreds of interrupt sources (each with its own data structure) all incoming thousands of times per second in no predictable order? Not a pretty picture-it's called "cache thrash." *[Editor's Note: For a more detailed look at caching in theory and practice, see "Firmware Furnace" beginning on page 55.]*

Instead of built-in code or caches, the H16 offers three other programmable options to exploit the on-chip memory-data RAM, global register banks, and ring register banks.

To explain these, refer back to Figure 2 and we'll see how the rest of the figure, besides R0–R15, comes into play. First, let's get the conventional stuff out of the way. The *PC* is the program counter, while the *SSP* is the Supervisor Stack Pointer (the H16 features two-level-user and supervisor-protection). The Exception *(EBR),* RAM *(RBR),* and I/O *(IBR)* Base Registers all serve to allow the corresponding resource (respectively the exception table, on-chip RAM, and on-chip I/O) to be relocated within the CPU address space. The Status Register (*SR*) contains the condition codes (zero, carry, etc.) as well as interrupt mask and privilege level bits; itisonlyaccessibleinsupervisormode. The Condition Code Register (*CCR*) contains a copy of the same condition code bits as the SSR, but not the other privileged items, and is accessible in user mode.

One way to use the on-chip memory, as data RAM, is straightforward enough. The goal is to position your data, especially 32-bit variables, in the on-chip RAM. If all the data won't fit, choose those variables that are accessed most frequently. Certainly, the on-chip RAM is also the best place for the stack since accesses occur often and are usually 32-bit.

The second way to use the RAM is as global register banks. The 1K bytes of RAM can hold up to 16 banks, with each bank occupying 16x32 bits (64 bytes). You canprogrammablychoose whether 2,4,8, or 16 banks are used as shown on the bottom left of Figure 2. If fewer than 16, the remaining RAM can be used as data/stack RAM as described previously. Global banks are ideal for high-speed interrupts since a separate bank can be devoted to each interrupt handler. Then, when an interrupt occurs, instead of having to save/restore the "context" (register contents)---a process that can easily take 10-20 µs—only the workingbank need be switched. Figure 3 shows the

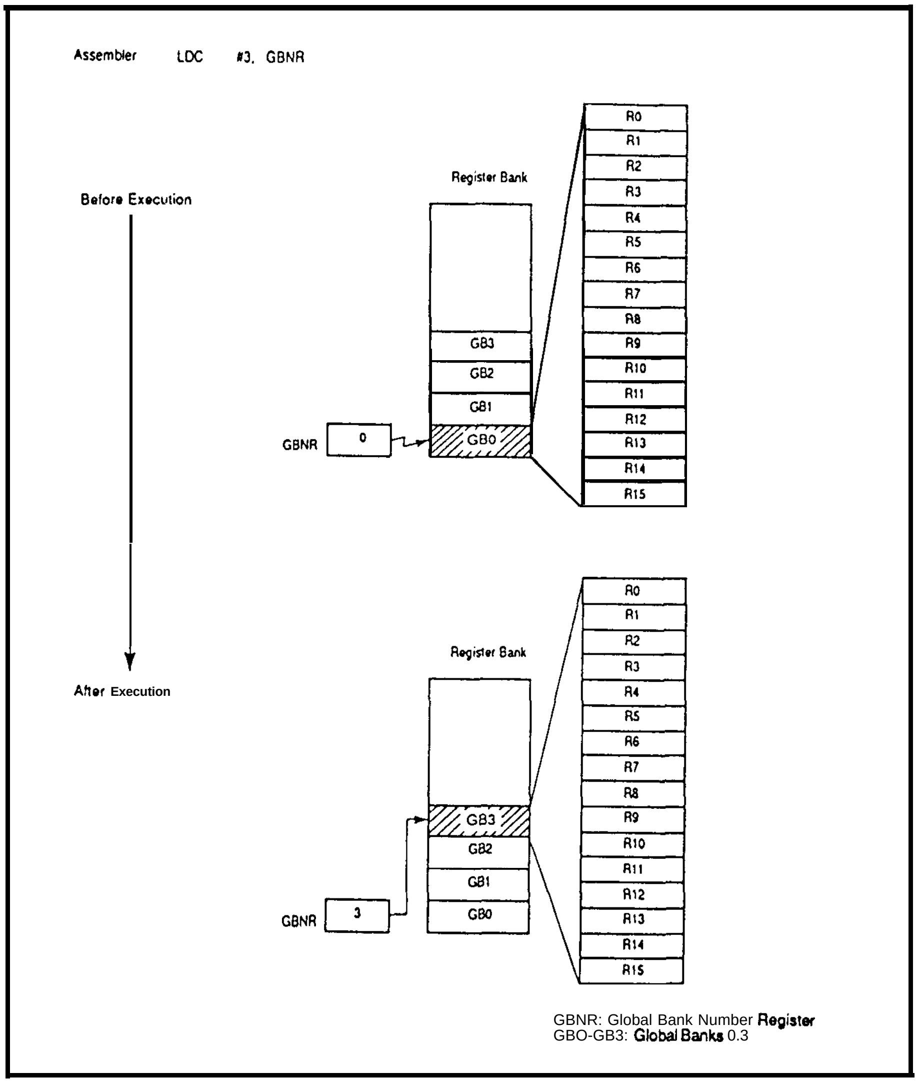
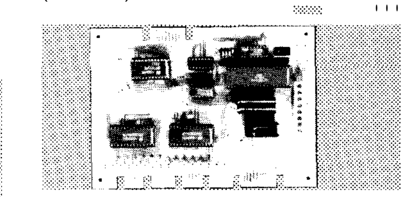


Figure 3(a)—*The* easiest way to change global register banks is to *simply load the new bank number (in this example, bank 3) in the GBNR (Global Bank Number Register). This "context switch' is an order of magnitude faster than the usual approach of moving register contents on/off the stack.*

RELAY INTERFACE

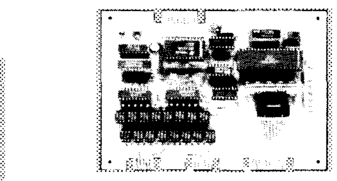
AR-16 Relay Interface .... $ 89.95
Provides software control for 16 external relays, expandable to 126 relays with EX-16 expansion cards. Plug-in relay cards. power relays and latched relays with status available. May be connected directly to serial printer port or RS-232.

EX-16 Expansion Card .... **$ 59.95**
(16 channel)



ANALOG TO
DIGITAL

ADC-8 (8 channel)' .... **$ 89.95**
ADC-16 (16 channel)' .... **$ 99.95**
Input temperature, joystick movement. voltage, pressure, energy "sage. light levels etc Voltage input level is adjustable (0 to 5 volts typical)
Connects to RS-232 or RS-422 ports



STATUS INPUT

STA-8 (8 channel)' .... **$ 99.95**
STA-16 (16 channel)* **$119.95**
STA-16-TT (touch tone input)' . . $159.95
Input on/off status of switches, relays, thermostats. security devices, smoke detectors, pressure switches and hundreds of other devices. Touch Tone input decodes all standard telephone touch tones and may be connected directly to a telephone line using an inexpensive coupler Connects to RS-232 or RS-422 ports

'Inputs are expandable to an additional 126 Status Inputs or 16 analog inputs. Add up to 112 relay outputs using EX-16 expansion cards.

- FULL TECHNICAL **SUPPORT**...provided over the telephone by our staff. A detailed technical reference manual is provided with each order including software examples in Basic and Assembly Language.
- Engineered for continuous 24 hour industrial applications.
- Use with IBM and compatibles, Tandy. Apple and most other computers with RS-232 or RS-422 ports. All standard baud rates and protocols may be used (50 to 19,200 baud) default is 9,600 baud 8 data bits, 2 stop bits, no panty. Use our 600 number to order free Information packet.
Technical Information (614) 464.4470

24 HOUR ORDER LINE (600) **842-7714**
Visa Mastercard American Express COD

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

Reader Service #120

INTRODUCING

étude™

25 MHz 8-bit

ANALOG-TO-DIGITAL CONVERTER

Based on the TRW THC1068-1 hybrid flash converter, its high signal-to-noise ratio yields excellent accuracy at the Nyquist limit.

- 4 KB of cache SRAM or to host as converted at DMA speed
- I/O or DMA data transfer
- 10 MHz full-power bandwidth
- 3.92 mV resolution
- 16 jumper selectable base addresses
- Input unipolar or bipolar
- External clock and trigger inputs
- Software source code included

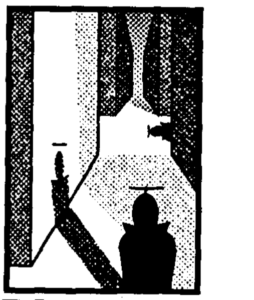
INTRODUCTORY PRICE $299

ALSO AVAILABLE AS A KIT FOR $99, INCLUDING:

- Printed Circuit board
- Software
- Manual & assembly instructions

(KIT introductory price includes PAL and 4KB of high-speed SRAM)

Requires: PC compatible ½ length 8-bit expansion slot. DOS 2.11 or greater. EGA, VGA or Hercules display needed for graphic representation of data.



Silicon Alley Inc.

P.O. BOX 59593
RENTON, WA 98058
206.255.7410

© 1989 Silicon Alley Inc. étude is a trademark of Silicon Alley Inc. Other brand or product names are trademarks or registered trademarks of their respective holders. Prices and specifications subject to change.

Reader Service #151

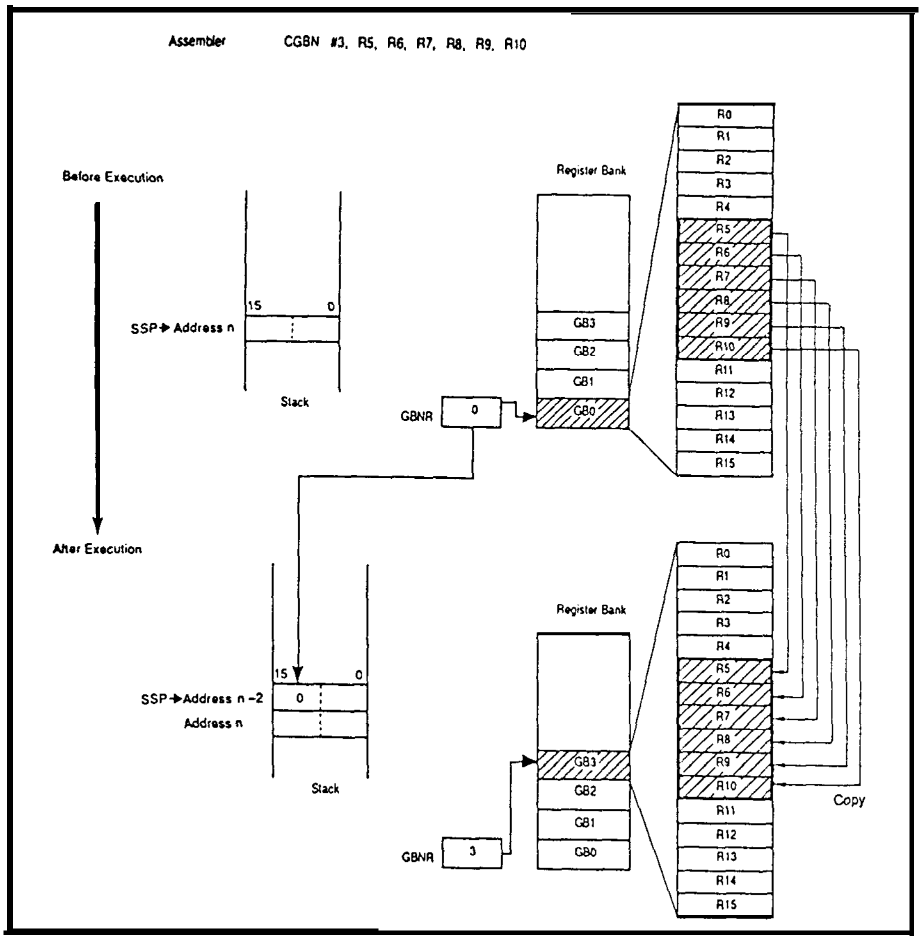


Figure 3(b)—*Another way to change global banks, using the CGBN (Change Global Bank Number) instruction, has two advantages. First, the old bank number is pushed on the stack and can be later recalled with the PGBN (Pull Global Bank Number) instruction. Second. selected registers can be copied from the old bank to the new during the switch.*

two ways a bank switch can be invoked. The first (3a) is simply to write the new bank number into the Global Bank Number Register (*GBNR*). The second, much fancier way (3b) is to use special CGBN (Change and Push Global Bank Number) and *PGBN*(Pull Global BankNumber) instructions. As expected, these use the stack to save and restore bank numbers so you can keep track of "nested" interrupts/banks. Furthermore, you can specify that the contents of any or all (or, of course, none) of the general-purpose registers be copied from the old bank to the new bank allowing you to keep track of key global values across interrupts/banks.

The final way to use the on-chip RAM is ring bank mode. As shown on thebottomrightof Figure2, thismode allocates the RAM as eight global banks (described in the previouspara-graph) and eight ring banks. Finally, we can finish up with Figure 2. The Bank Mode Register (*BMR*) selects which type of bank setup to use: global or ring. The ring bank mode relies on the remaining registers: Current Bank Number Register (*CBNR*), Valid Bank Number Register (*VBNR*), and the Bank Stack Pointer (*BSP*). Together, the ring banks and these control registers serve to implement a "register window" scheme-a concept made popular by the RISC folks. Its purpose is to reduce subroutine call overhead by keeping stuff in registers across subroutine calls rather than the typical case in which arguments and return values must be shuffled on and off the stack. The programmer hooks to the ring banks are in the form of special instructions Increment & Decrement Current Bank Number (ICBN and DCBN respectively) and two unique addressing modes: Current and Previous bank. Simply speaking,

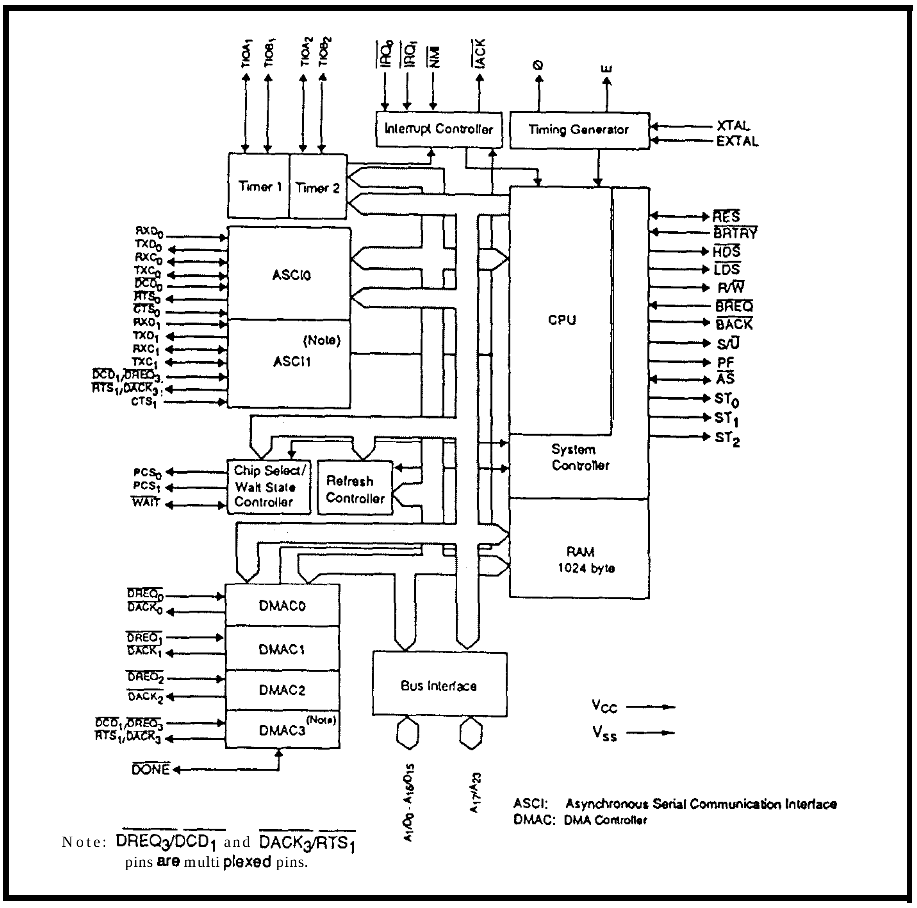


Figure 4—*The H 16 integrates three types of logic: the CPU and RAM/register banks, system logic (chip select/wait state. DRAM* **refresh,** *and interrupt controllers), and I/O functions (UARTs (ASCIs), timers, and DMA).*

in ring mode the programmer has access to three register banks simultaneously: a global bank, the “current” bank, and the “previous” bank. The global bank(s) can be used for, naturally, global variables; the current bank can hold a subroutine’s local variables; and **the** previousbank is **used** to send/return that subroutine’s arguments and results. If subroutines are nested more than eight levels deep, theoldest levels are automatically pushed (and restored when you finally come back up for air) on the bank stack (*BSP*). The ultimate goal of all this is to keep the action on-chip instead of having to go off-chip where things slow down.

We won’t say more about ring bank mode (it’s pretty heavy stuff) since the simpler global bank mode is ideal for our multitasking BASIC compiler. Each BASIC task (up to 16) is allocated its own register bank, so switching tasks is simple and fast.

Timer Mode	TIOA Function	TIOB Function
Interval timer	-------------	-------------
Event counter	-------------	pulse input
One-phase pulse generator	pulse output	-------------
Two-phase pulse generator	pulse output 1	pulse output 2
Pulse width measurement	pulse input	-------------
Frequency measurement	trigger input	pulse input
S/W triggered one-shot	-------------	pulse output
H/W triggered one-shot	trigger input	pulse output

Figure 5—*Each of the H 16s two* **on-chip timers** *has two I/O lines, TIOA and TIOB,* **which** *play* **different roles depending on which timer** *mode is used. TIOA and TIOB can a/so* **be** *used as simple I/O lines if they aren’t needed for timing.*

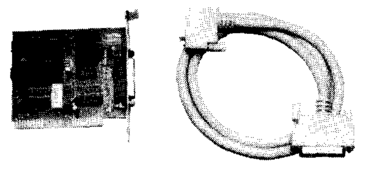
JDR Microdevices

- *30 day money back guarantee*
- *1 year warranty on all products*
- *Toll-free technical support*

New! Modular Programming System

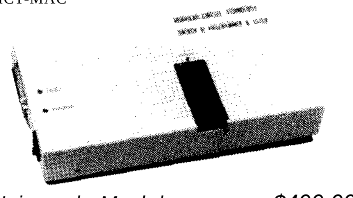
FROM MODULAR CIRCUIT TECHNOLOGY

This integrated system is ideal for developers--it easily expands as your needs grow! All the modules use a common host adaptor card so you need just one slot to program EPROMS, PROMS, PALS and more!



Host Adaptor Card *$29.95*
- A universal interface for all the programming modules
- User-selectable programmable addresses prevents addressing conflicts
- Includes a high quality molded cable

MCT-MAC



Universal Module *$499.99*
- Programs EPROMS, EEPROMS, PALS, bi-polar PROMS, 8748 & 8751 series devices.
- Programs 16V8 & 20V8 GALS (gallium arsenide) from LATTICE, NS, SGS
- Tests TTL, CMOS, Dynamic &Static RAMS
- Load disk, save disk, edit, blank check, program, auto read master, verify and compare
- Textool socket accepts .3" to .6" wide IC's from 8-40 pins

MCT-MUP

EPROM Module **$179.95**
- Programs 24-32 pin EPROMS, CMOS EPROMS and EEPROMS from 16K to 1024K
- HEX to OBJ converter
- Auto, blank check/program/verify
- VPP selectable 5, 12.5, 12.75, 13, 21& 25 volts
- Normal, Intelligent, interactive, and quick pulse programming algorithm

MCT-MEP		
MCT-MEP-4	I-EPROM Programmer	5169.95
MCT-MEP-8	8-EPROM Programmer	5259.95
MCT-MEP-16	16-EPROM Programmer	$499 95

PAL Module *$249.95*
- Programs MMI, NS, TI 20 & TI 24 pin devices
- Blank check, program, auto, read master, verify and security fuse blow

MCT-MPL

PAL Programming development software
MCT-MPL-SOFT 599.95

Order toll free 800-538-5000

MasterCard VISA MEMBER DIRECT MARKETING ASSOCIATION MMC

2233 BRANHAM LANE, SAN JOSE, CA 95124
LOCAL: (408) 559-1200 FAX (408) 559-0250 TELEX 171-110
RETAIL STORE: 1256 S. BASCOM AVE., SAN JOSE, CA
HOURS: MON.-FRI. 9-7, SUN. 12-4 (408) 947-8881

Terms: Minimum order $10.00. For shipping and handling include $2.50 for ground and $3.50 for air. Orders over 1 lb and foreign orders may require additional shipping charges—please contact our sales department for the amount. CA residents must include applicable sales tax. Prices are subject to change without notice. We are not responsible for typographical errors. We reserve the right to limit quantities and to substitute manufacturer. All merchandise subject to prior sales. A full copy of our terms is available upon request. Items pictured may only be representative.

COPYRIGHT 1989 JDR MICRODEVICES CONTINENTAL US AND CANADA

Circle No. 115 on Reader Service Card
Dealers Circle No. 116

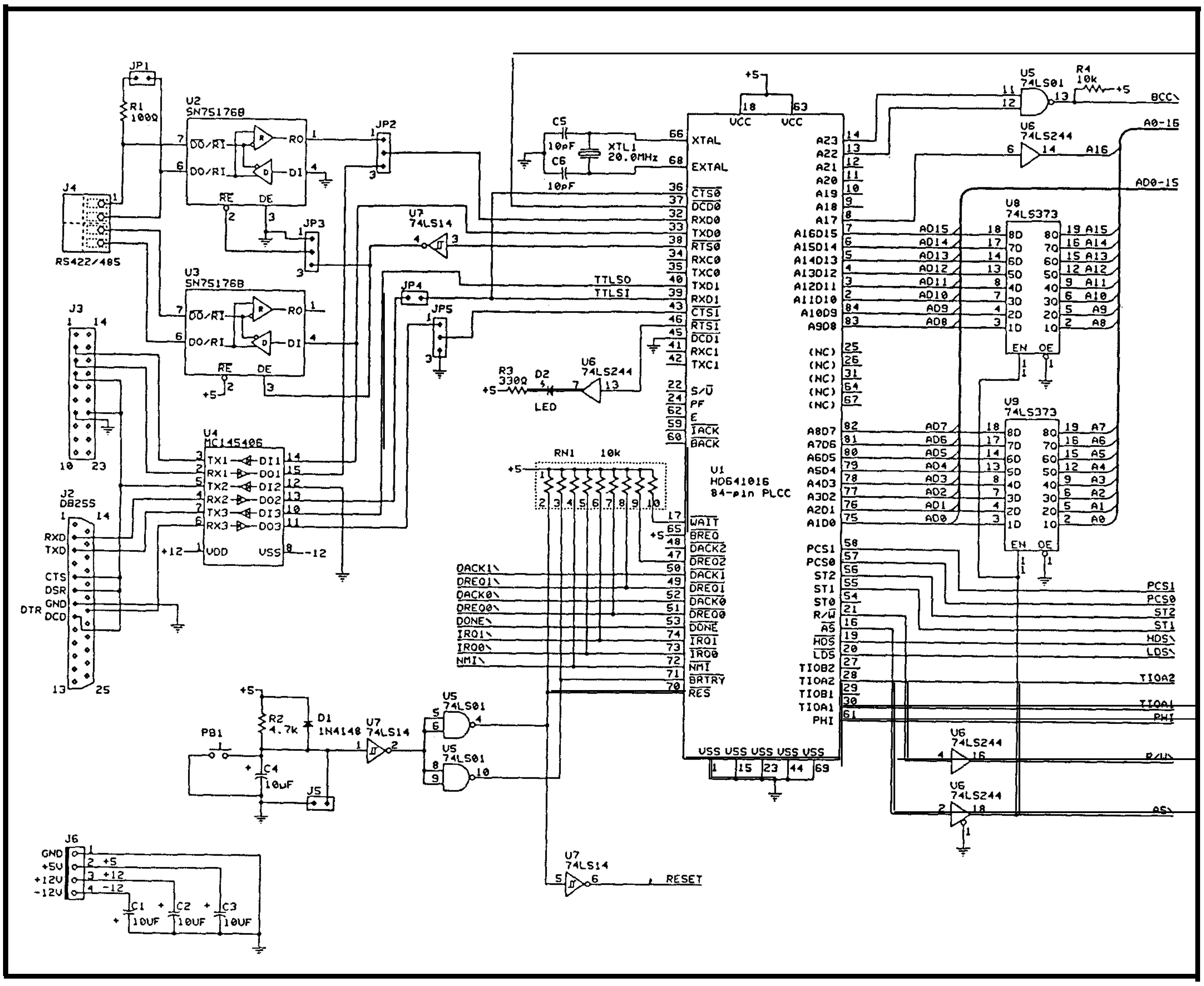


Figure 6—*The first Page of the schematic consists mainly of the H16, address latches U8 and U9 (which serve to demultiplex the processor's address/data bus), and, on the left. the serial I/O transceivers (which offer RS-232, RS-422, and RS-485 options).*

MORE THAN JUST A CPU

There was a time when the previous section would have completely described the CPU chip. But, looking at Figure 4 you can see everything said so far comprises only a small portion (CPU and RAM) of the device.

The rest of the goodies on the chip can be roughly divided into two categories: system (less affectionately known as "glue") logic and I/O.

Those familiar with the company's HD64180 8-bit chip will recognize the integration strategy:

- First, build in system functions that are required in almost every design and/or work better on-chip than off.
- Then, add I/O functions using the same criteria.
- Keep adding stuff until you run out of transistors.

Exploiting the same integration strategy, the H16 incorporates the same basic functions as the HD64180, though each H16 function is upgraded from the older chip's counterpart.

Chip Select/Wait State Controller

The H16 provides two chip-select lines (PCS0, PCS1) which encode one of four chip-select areas, each of which has programmable size (in 64K-byte increments) and location within the processor's 16M-byte address space. Each area's access privilege level (user or supervisor) can be assigned, while accessing an "undefined" address (not mapped to any chip select) causes an exception, all of which helps to harden the system against the inevitable software bugs. Each of the four areas also features programmable wait states (from 0 to 7) and an external WAIT input is provided as well.

DRAM Refresh Controller

There is no better place to put a DRAM refresh controller than on the CPU chip (with the possible exception of on the memory chip itself-more later). Besides hiding the asynchronous and high-speed timing glitches that can plague an external DRAM controller scheme, integration unifies DRAM refresh with other CPU operations such as DMA and even reset. The H16 DRAM refresh controller is improved from that of the '180 in two key ways. First, it supports DRAMs needing up to 11 bits of refresh address (including 1M- and 4M-bit chips) where the'180 only offers 8 bits (up to 256K-bit chips). Second, the H16's controller deals with the special cases of refresh when another "master" has control of the bus and when the H16 is in one of its low-power operation modes (SLEEP and STOP). The controller keeps track of deferred refreshes and can juggle the bus priority to catch up before the DRAMs start to fade.

Interrupt Controller

Being designed for real-time applications, the H16 includes an interrupt controller to field requests from a multitude of sources. The shortly-to-be-discussed on-chip I/O devices alone account for 22 interrupt sources. To manage all these interrupts effectively, the controller incorporates four-level programmable interrupt priority. NMI is always the highest priority, but the other internal and external sources can be freely assigned to the remaining three levels. Not only does this allow prioritization depending on the application, but the priority can be dynamically changed to adapt to varying loads and response time needs.

The H16 includes the same basic I/O selection as the HD64180, but, as in the case of the on-chip system logic, the functions are somewhat improved.

UARTs

The H16 includes two full-function UARTs called ASCIs (Asynchronous Serial Communication Interfaces). Like those on the '180, the ASCIs contain all the features and functions of the fanciest stand-alone UART chips-baud rate generator, programmable formats, handshake lines, and so on. Two key improvements are a) the baud rate generator is more independent of the system clock (you can derive standard baud rates at many different system clock frequencies) and b) there are a full complement of handshake lines (CTS\, DCD\, and RTS\ for each channel) and they work in a more straightforward manner than on the '180.

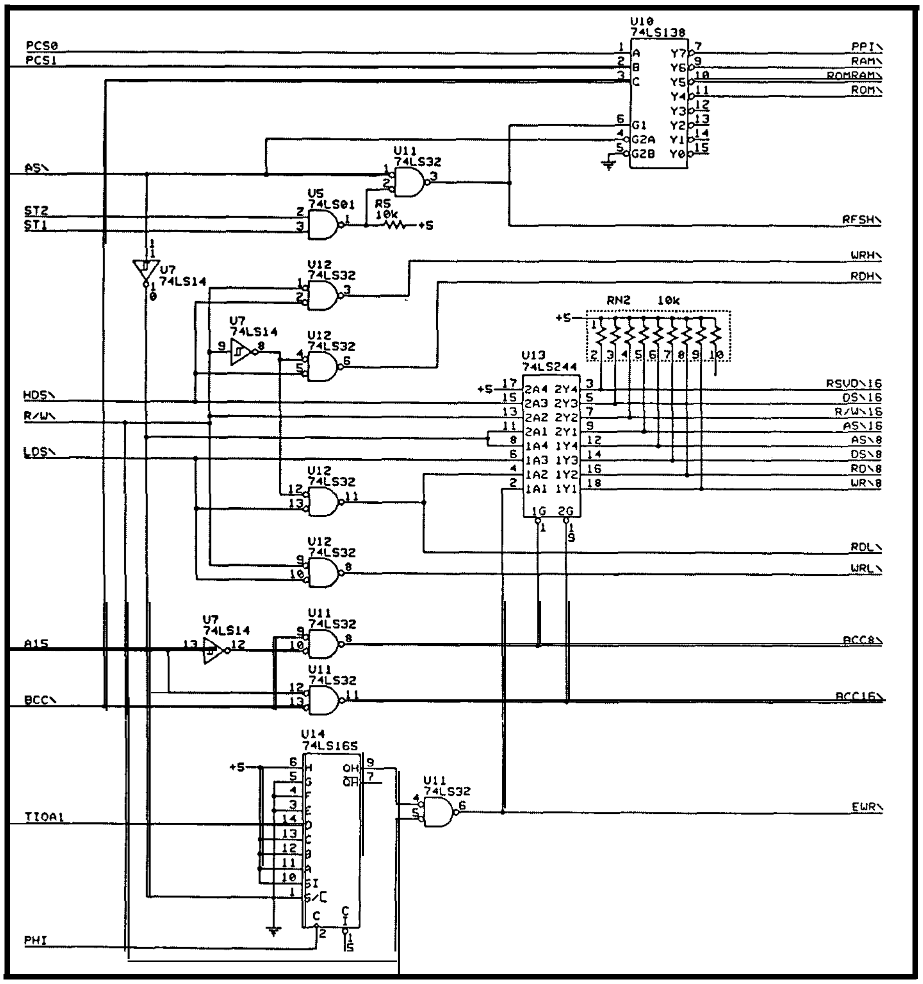


Figure 7—The second page of the schematic contains logic (U10, U11, and U12) to generate various on-board control signals. U 14 is responsible for generating an 'Early Write' (EWR\) strobe required by the 8255s and other chips with long write data hold times. U 13 buffers 8- and 16-bit control signals asserted during off-board BCC bus accesses.

Timers

The H16, like the '180, includes two 16-bit counter/timers. However, unlike the HD64180's timers, which do little more than count system clock pulses, the H16 timers each have two I/O lines (TIOA and TIOB) and a plethora of operating modes which are illustrated in Figure 5.

Notably, the timers also work with the on-chip DMAC so you can set up a table of timer values which are automatically loaded one after the other by the DMAC to generate any kind of variable waveform you desire without CPU intervention.

DMA Controller

Speaking of the DMAC, the H16 includes four DMA channels (versus the HD64180's two) which speed the transfer of data within the system. The DMAC supports transfer from and to anywhere, on-chip or off, with one exception: you can't DMA to/from the DMAC itself! The need for a DMAC might be questioned by those used to lower-end 8-bit chips, but remember that one of the main reasons to use a 16-bit chip is to accelerate data transfer. Furthermore, if DMA is needed, you'll be very glad it is included on-chip since the integration eliminates all kinds of potential headaches. For instance, the on-chip DMAC works hand-in-hand with the chip-select/wait-state controller, DRAM refresh controller, and even the user/supervisor protection scheme.

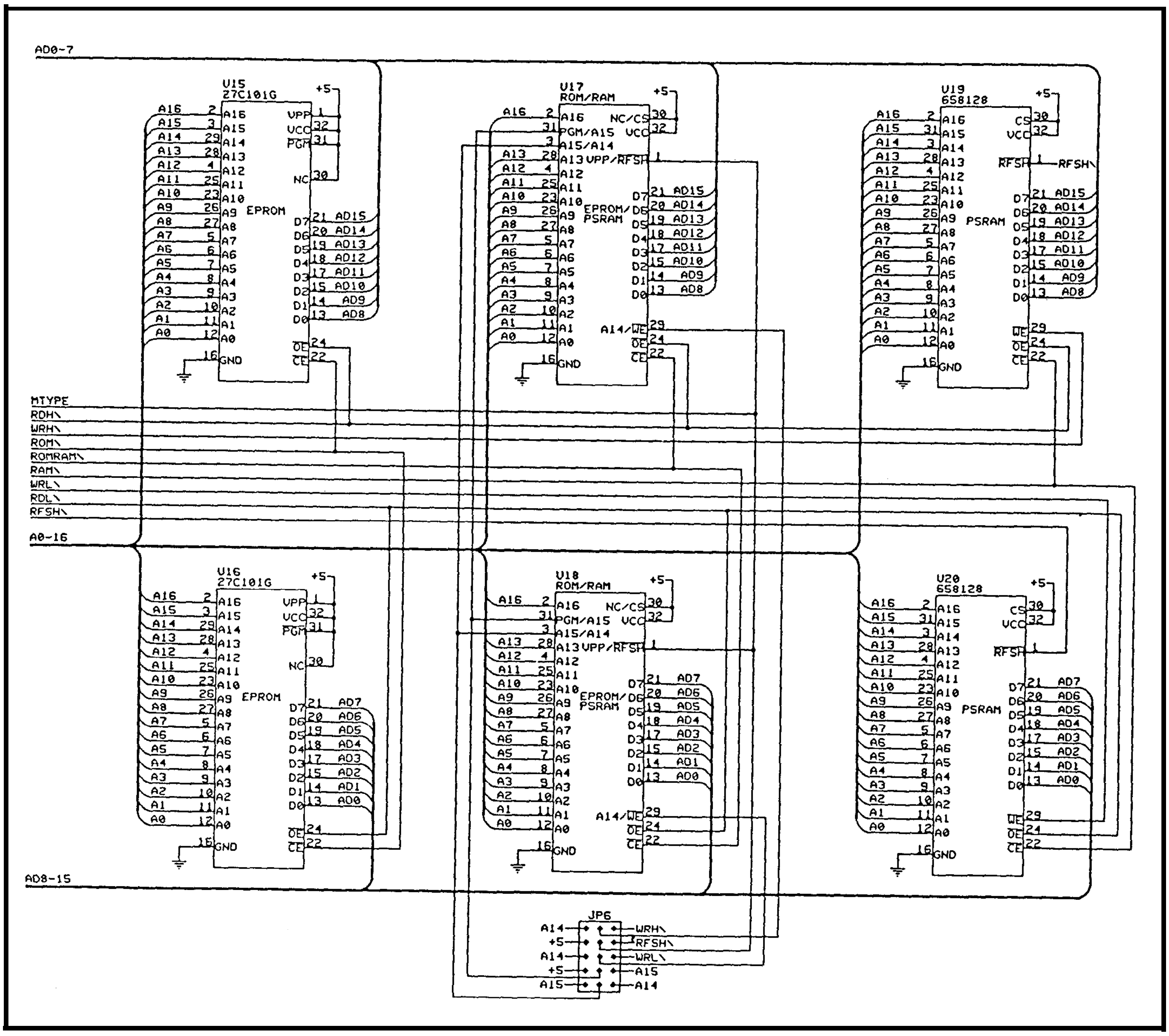


Figure 8—*The six 32-pin* **sockets which** *comprise the BCCH16's memory are each capable of holding 1M bit memory ICs for a Total of 768K* **bytes.** *Sockets U 17 and U18 are switchable between RAM and ROM with jumpers (JP6) located just below U18.*

Beyond this, the DMAC features all kinds of extras: different request (edge or level sense), bus (burst, single cycle, or cycle steal), and priority (normal or "express") modes;continue operation for repetitive block transfers (ideal for CRT refresh); and bus matching between 8- and 16-bit ports. Furthermore, unlike the '180 DMAC which features only dual-address (read source and then write destination) transfers, the H16 DMAC also offers single-address transfers (read source and write destination simultaneously) by exploiting dedicated DACK\ lines for each channel.

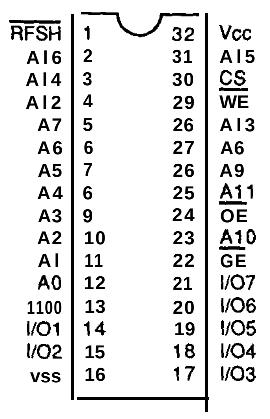
Anyway, the bells and whistles are great, but those who really need DMA will tell you that the bottom line is raw speed. At a CPU clock rate of 10 MHz, the H16 can achieve a maximum throughput of 6.66 Mbytes/ second which, I suspect, is more than fast enough for most applications.

I could go on and on about the H16—bus retry (you don't want your "mission critical" box to **choke on an** alpha particle), prefetch algorithm, trace exception, precharge wait state, and so on. Anyway, you get the idea. The H16 has everything on it but the kitchen sink (actually, I heard a rumor it is in there, just not documented). If you want to know more, wade through the 600-page(!) manual yourself.

For now, we can sum up the H16's role by referring to the first page of the schematic (Figure 6). There we see the H16 (U1) which, thoughdrawn to look like a DIP, is actually packaged in an approximately 1-sq.-inch 84-pin PLCC (Plastic Leaded Chip Carrier).

At the top left of the schematic is the logic which connects the H16's ASCIs (UARTs) to the outside world. ASCI channel 1 (RXD1, TXD1, etc.) is intended to connect to the main console (whether a terminal or PC running terminal emulator) via RS-232 transceiver U4 and DB25 connector J2. ASCI channel 0 serves double duty, either connecting to an auxiliary RS-232 device (via U4 and connector J3) or an RS-422 or RS-485 channel via transceivers U2 and U3 and screw connector J4. A similar RS-485 connection is also offered on the BCC180, BCC52CX, RTC31, and RTC52 boards which allows you to string a combination (up to 32) of **these boards** together in a mini network. Notice how the unused ASCI handshake lines serve alternate roles as I/O lines. DCDO\ the crystal connected to the top left of the H16 (20-MHz crystal = 10-MHz system clock) can be changed to a different frequency without having to modify the software. At the bottom left of the schematic is the reset circuit which kicks things off at power-up, and also allows manual reset via on-board push button PB1 or a remote switchvia J5. Resistor pack RN1 serves to pull-up active-low control inputs such as DREQ\ (DMA request) and IRQ\ (Interrupt request).

Meanwhile, the right side of the schematic is mainly composed of the



Symbol	Pin Name
A0–A16	Address Inputs
I/O0–I/O7	Data Input/Output
RFSH	Refresh
CE	Chip Enable
OE	Output Enable
WE	Write Enable
CS	Chip Select
Vcc	Power Supply
Vss	Ground

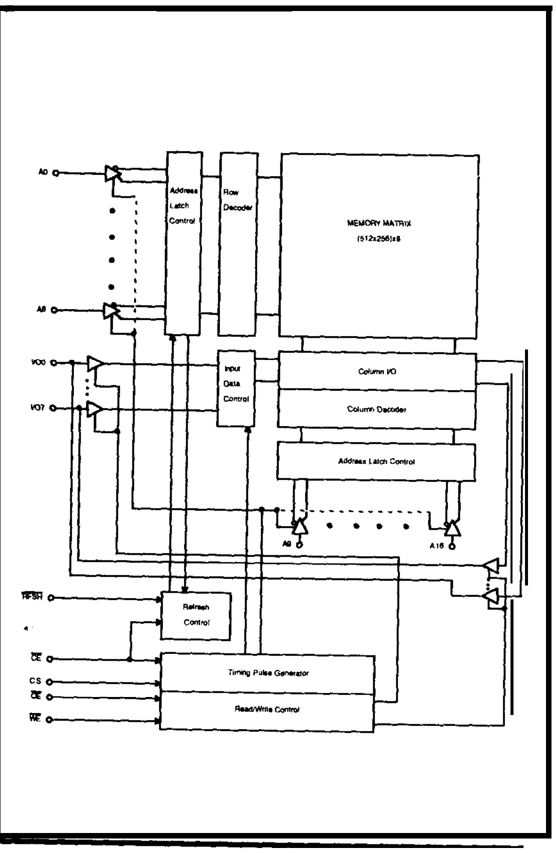
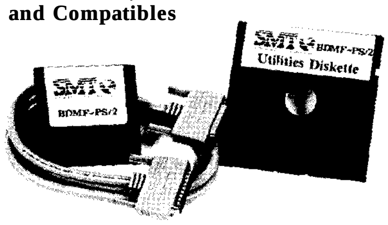


Figure 9—*From the outside. the HM658 128 1M-bit* **Pseudo-Static RAM** *(PSRAM)* **looks much** *like a conventional static RAM. Only the RFSH* **pin (pin 1) gives away the fact it's** *actually a DRAM inside,*

acts as an "MTYPE" input which we'll discuss a little later. RTS1\ is used as an output to drive an LED for boot-time diagnostics. CTS0\ is connected to channel 1 receive data input (RXD1)—a strange looking arrangement indeed-allowing the CPU to "watch" the raw serial bit stream to automatically determine the terminal baud rate when the system is booted. Notably, this auto-baud scheme is independent of the system clock, so two'LS373 latches which, clocked with Address Strobe (AS\), serve to de-multiplex the H16 address/data bus. You may have a question about the odd naming for the address/data bus (e.g., A1D0) but it makes sense when you recognize the H16 uses separate high and low data strobes (HDS\ and LDS\, pins 19 and 20 respectively) instead of an A0 address line. However, for clarity when connecting to outside chips which have an AO, off-

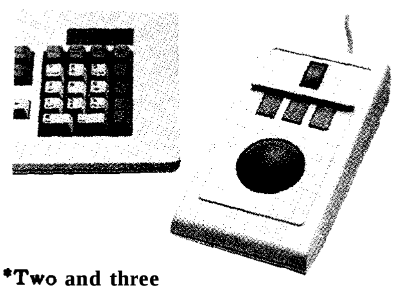
THE INTERCHANGE"

Bi-directional Data Migration Facility for IBM PS/2, AT, PC, PORTABLE and Compatibles



Features:
***Parallel port to parallel port.**
***Economical method of file transfer.**
***Bi-Directional file transfer easily achieved.**
***Supports all PS/2 systems (Models 30, 50, 60, and 80).**
***Supports IBM PC, XT, AT, Portable and 100% compatibles.**
***Supports 3 1/2 inch and 5 1/4 inch disk transfers.**
***Supports hard disk transfers.**
***Supports RAMdisk file transfers.**
***The SMT 3 Year Warranty.**
ONLY $39.95

FastTrap™
The pointing device of the future is here!



***Two and three axis pointing capability.**
***High resolution trackball for X and Y axis input.**
'High resolution fingerwheel for Z axis input.
***Use with IBM®PC's, XT's, AT's and compatibles.**
***Three input buttons.**
***Full hardware emulation of Microsoft" Mouse.**
***Standard RS-232 serial interface.**
***Includes graphics drivers and menu generator.**
***Easy installation.**
***1 year warranty.**
***Made in U.S.A.**
ONLY $149.00

LTS/C Corp.
167 North Limestone Street
Lexington, Kentucky 40507
Tel: (606) 233-4156

Orders (800) 872-7279
Data (606)252-8968 [3/12/2400 8-N-11
VISA, Mastercard, Discover Card, TeleCheck

Reader Service #135

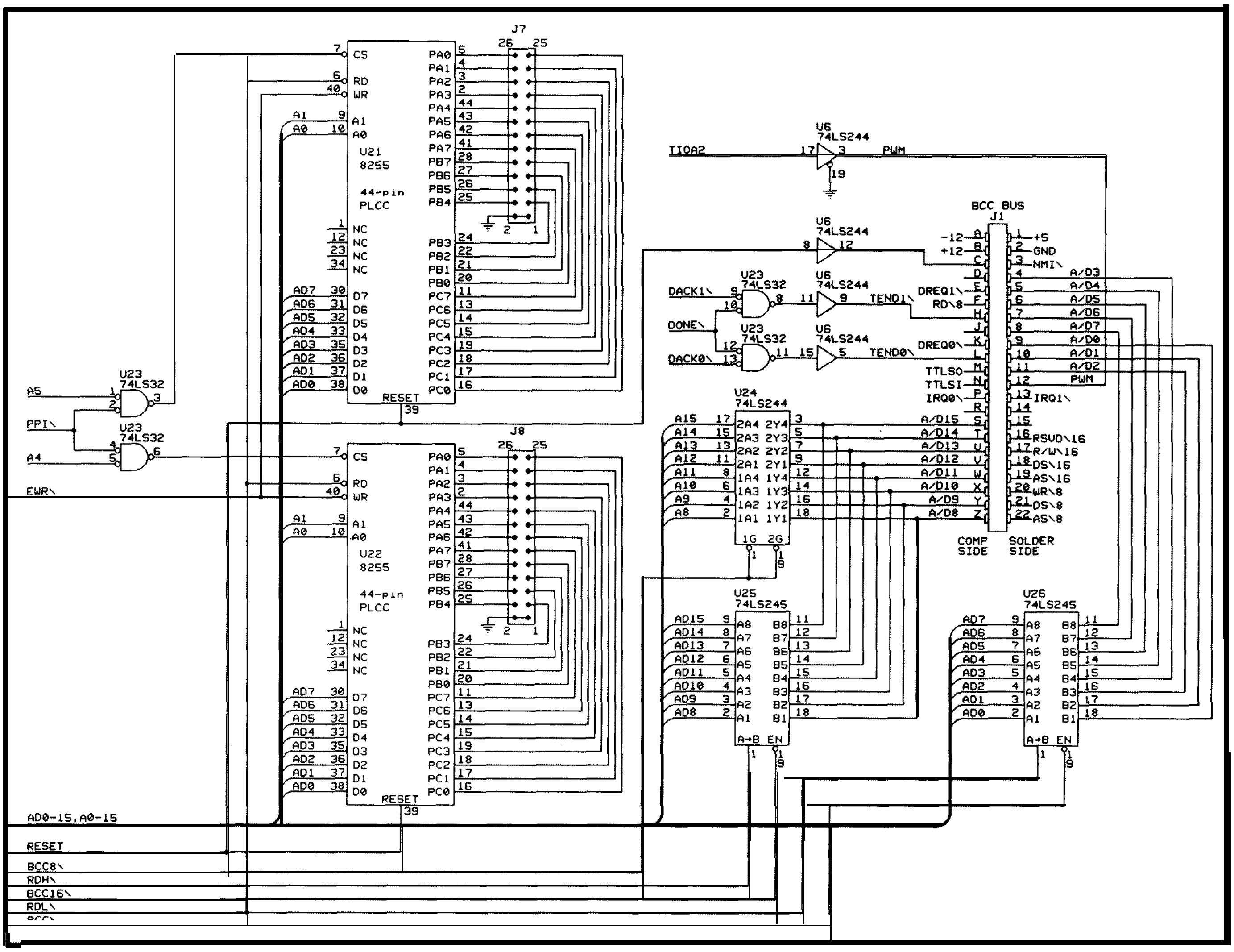


Figure 1 0—*The last* section *of the schematic finishes the design with two 8255s (48 I/O lines) and the BCC* **bus** *connection.*

chip the bus names are changed to the more conventional ADO-15 and AO-A15.

The AND gate at the top right decodes accesses to the top 4 megabytes as a signal called BCC\. This is used to signal that an access is going off the board onto the BCC bus.

Finally, at the top of Figure 7, U10 (74LS138) decodes the H16 chip select lines (PCS0 and PCS1) to generate chip selects for the other devices on the board. Note how the "C" input (most-significant bit) to the decoder is connected to the previously mentioned BCC\ signal. When BCC\ is low indicating an external bus access, the chip select outputs are shifted to the YO-Y4 pins which aren't connected to anything. This ensures that no on-board devices will be selected during an off-board access. Also, gates U5 and U11 decode H16 status lines ST1

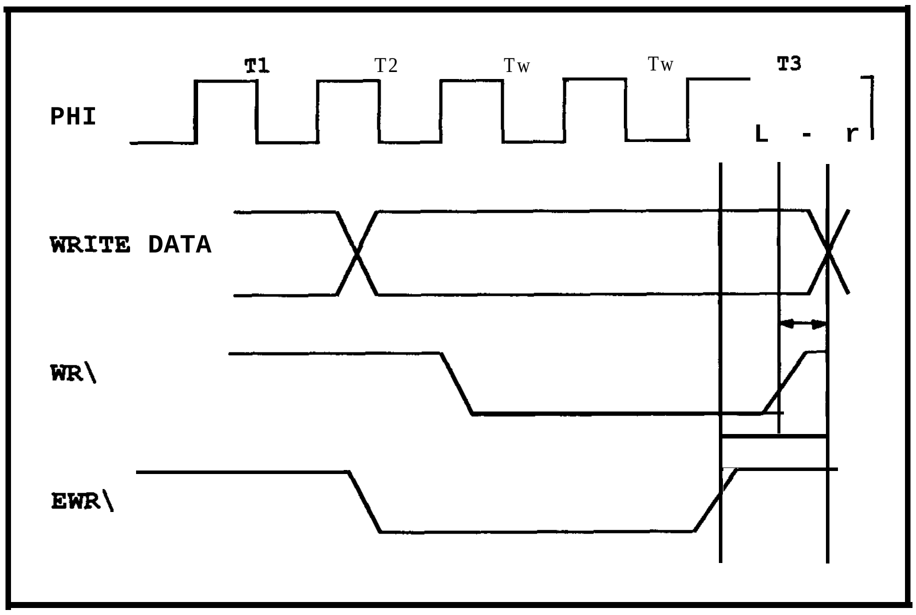


Figure 1 1 -The 8255s *require a long wife data hold time (the time for which the data remains stable after the trailing edge of WR\).* **The** *time provided by a WR\ derived from the H16 data strobes (HDS\ and LDS\) is not sufficient, so an "Early Write' (EWR\) is generated* **which easily meets the 8255** *spec.*

and ST2 to identify an H16-generated DRAM refresh cycle. The RFSH\ signal disables the decoder to prevent unwanted chip selects during a refresh cycle. Finally, OR gate U12 along with inverter U7 serve to generate separate high and low data bus read and write strobes (named RDH\, RDL\, WRH\, and WRL\ respectively) from the H16 HDS\, LDS\, and R/W\ signals. For now, ignore the rest of the figure which is related to BCC-bus interface logic. I'll get to it later.

BIG MEMORIES

Besides speed, the other reason to move from an El-bit chip is to eliminate oppressive memory size constraints—namely the infamous 64K barrier. Some older chips, such as the aforementioned HD64180 and a certain well-known family of Intel chips, make efforts in the form of "pages" or "segments" to go beyond 64K. These patches work OK in some cases (e.g., quite fine for RAM disk) but the 64K limit is still a drag when it intrudes on your programming style.

To take full advantage of the H16's linear 16M-byte space means including a bunch of memory on the BCCH16. Yet, the design goals, such as low-power operation, small form-factor, and reasonable cost, must also be met. Finally, the basic BCC concept-that the board serve as its own development system-imposes additional constraints.

The solution exploits the latest in JEDEC standard byte-wide memory ICs as shown in Figure 8. Six 32-pin sites are provided organized as three pairs (high and low data bus) of memories. Starting from the left, U15 and U16 serve as the boot/system "ROMs" (actually EPROMs) which store the BASIC compiler and run-time package. These sockets are labeled as containing 27C101G devices which are 1M-bit (128K-byte) chips, though a 27512 (64K bytes) chip will work (plugged into the lower 28 pins) as well. The middle pair of sockets, U17 and U18, serve a dual role. During BASIC program development, these sockets hold RAM chips (actually "pseudo-static" RAM chips, more in a moment). After development is finished, the compiled program ("production") code is burned into EPROMs which replace the RAMs. Switching the socket function between (Pseudo-Static)RAM and (EP)ROM requires jumper area JP6 at the bottom of the page. If you look closely, you'll see one of the jumpered lines, the RFSH\ signal, is connected to the MTYPE signal which goes back to the CPU (via an unused serial port handshake input as described earlier). By watching for activity on the RFSH\ line, the CPU can determine which way the jumpers are set. Finally, on the right side, U19 and U20 also hold pseudo-static RAM chips which serve as data RAM during development and production.

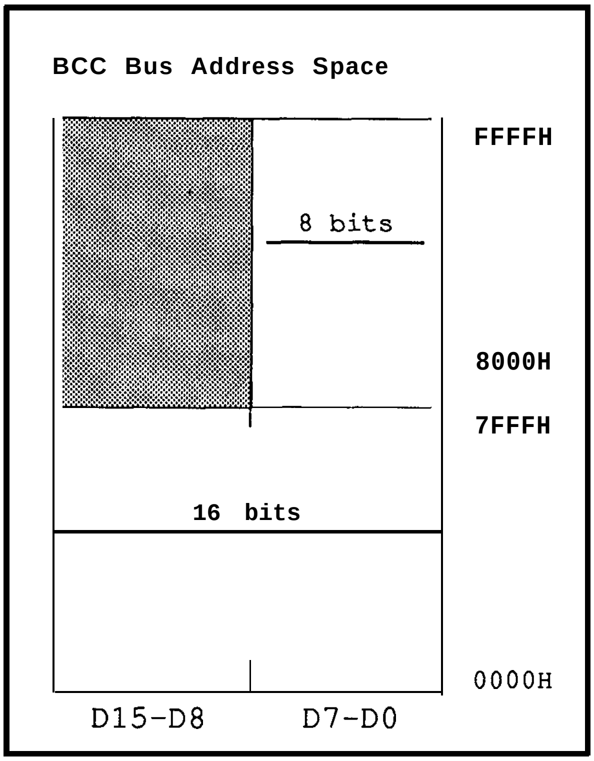


Figure 12—*To allow both 8-bit and 16-bit BCC bus access, the 64K bus address space is split into two pieces: 32K 8-bit ports and 32K 16-bit ports.*

Besides thanking the silicon wizards for "puttin' on the bits," there isn't much to say about the EPROMs which have worked pretty much the same since the days of the 2708 other than you should expect to see more and more 32-pin sockets (breaking the long-established 28-pin tradition).

The pseudo-static RAMs (PSRAMs) are another story. Basically, PSRAMs like the HM658128 (Figure 9) are designed to combine the best of both worlds-the low cost/bit of regular DRAMs with the packaging efficiency, ease of use, and EPROM compatibility of byte-wide static RAMs. Looking inside the PSRAM we see the top half of the chip is indistinguishable from an SRAM. The diagram doesn't show that the memory cells are actually DRAM type instead of the larger (four times more transistors) SRAM type. The only clue to the DRAM nature of this memory is the "Refresh Control" block at the bottom.

The PSRAM tries to make refresh as easy as possible by offering three different schemes: address refresh, automatic refresh, and self-refresh. The first, address refresh, is much like a regular 1M-bit DRAM in that the CPU, or other outside logic, is required to cycle through 512 refresh addresses every 8 ms (i.e., a new refresh address every 15 µs or so). The second method, automatic refresh, is somewhat like the first except no refresh addresses need be provided. A counter on the chip keeps track of them so all that's needed is to hit the RFSH\ (pin 1) line every 15 µs. The final method is self-refresh. As long as RFSH\ is held low, the PSRAM will refresh itself using internally generated refresh addresses (as in automatic refresh) and an internal refresh timer.

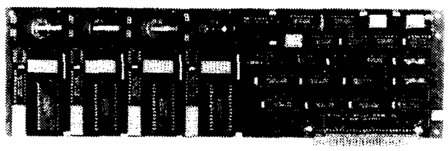
Having said all that, the obvious question is why use a PSRAM with refresh smarts built-in when, as described earlier, the H16 also includes refresh logic-who's in charge here? The answer has nothing to do with refresh. Instead, the PSRAM is used for other reasons including a) socket compatibility with EPROMs which is needed for the BCCH16 to be its own development system, b) to eliminate the **address mux TTLs required by regular DRAMs, and c) to take advantage of the most modern DRAM process without** being forced into the multimeg arena-using 1M x 1 DRAMs would dictate a minimum of two megabytes on the board and require much more power (almost 10x compared to two 128K x 8 PSRAMs) as well. Anyway, we're able to keep each chip (the PSRAM and H16) from stepping **on** each other's toes by using automatic refreshand simplyconnecting the H16-derived RFSH\ signal to the PSRAM RFSH\ pin.

PARALLEL I/O

The only other "big" chips on the board are two 8255 parallel I/O chips shown on the schematic in Figure 10. Besides offering 48 bits of I/O (since parallelI/Oisrequiredinalmostevery control application) these serve to connect an EPROM programming daughterboard for burning compiled BASIC programs into EPROMs destined for the ROM/RAM sockets (U17 **and U18). This is exactly the same** scheme as used on the BCC180, and the 8255 connectors on both the BCC180 and BCCH16 (J7 and J8) have equivalent pinouts. The gates (U23) connected to each 8255 serve to derive each device's chip select from the'138 decoded output (PPI\) and a couple of address lines. Observant readers will note that both 8255s are connected to the H16's lower data bus. In other words, they are accessed as separate 8-bit devices, instead of a single 16-bit pair. Blasphemous as it may seem (indeed, it would be easier to hook them up as a 16-bit pair) there is a good reason: compatibility with existing BCC180 BASIC programs. One of the great things about the BASIC is it includes statements (`P IOIN` and `PIOOUT`) which access the 8255s directly—no **Strange PEEKS/POKES** to clutter your program. However, these BASIC-180 statements interpret the 8255s as offering six 8-bit ports, so we follow suit with the BCCH16. Anyway, despite the 8-bit interface, you can still blast data out these ports at a megabyte or two per second. Never fear, you'll see how to implement true 16-bit I/O via the BCC bus shortly.

The only other thing to notice about the 8255 interface is the chips' write (WR\) lines are driven by a signal called EWR\ which stands for "Early WRite." It turns out that some older **chips** like the 8255 have, well, er, "interesting" timing specs. Particularly exciting is the "data hold from write" spec which dictates that the data written to the 8255 remain stable longafter theCPU completes the write cycle and moves on. An alternative to making the data do overtime. is to make the WR\ signal punch in early (Figure 11). If you flip back to page two of the schematic (Figure 7) you'll see that EWR\ is generated by U14, a parallel-in, serial-out shift register clocked by the system clock (10 MHz). This circuit synthesizes the EWR\ signal which pretty much looks like the regular WR\ signal except it occurs earlier in the bus cycle. Something that looks odd is the connection of TIOA1 (one of the H16 timer lines)

PC BIOS Development · Industrial Control · Embedded Systems · ROM Development



ROM
EEROM
SRAM
SRAM AS ROM

The Kolod Research **R^{3}OM** Card

SUPPORTS MOST 28-PIN JEDEC MEMORY DEVICES. YOU CAN USE STATIC RAMS AS ROMS TO DYNAMICALLY DEVELOP AND TEST PC ROM BASED CODE *WITHOUT* BURNING IN ROMS AND FULLY DEBUG THE ROMS YOU DO BURN

The R^{3}OM card provides all the facilities you need to develop and deliver PC based BIOS, embedded systems, industrial control software, and ROM based applications

The R^{3}OM card features include:

* 4 independent 28 pin JEDEC defined sockets: each socket's address is switch-selectable to anywhere within the 0 to 1Mbyte address range
* ability to address and configure sockets independently or consecutively for a total of upto 128K SRAM, 256K ROM, or any combination of ROM, EEROM or SRAM in between
* use only the sockets and memory you need
* jumper select battery backup for any or all of the sockets
* variable wait-state generation for each of the sockets - means you can use slower memories in high-speed systems
* software or hardware controlled write protect for each socket (for SRAMs and EEROMs)
* unique software driven 3 channel DMA fly-by test circuitry
* flexible port I/O addressing
* complete technical documentation including programming and hardware specifications with examples (source code included. of course)
* full support for PC/XT/AT/386 machines
* high quality multi-layer construction

Engineering Excellence

Kolod Research Inc.
1898 Techny Court, Northbrook, Illinois 60062-5474 U S A

$285.00 +
shipping/handling
Made in U.S.A.

(312) 291-1586 for other options, sales &technical information

Reader Service #131

68000 K-System **Factory Direct Prices**

N o iws a bus that makes it easy of to use the e ntire f
Utilizing native 68000 signals, thelow cost 68000 systems

then designer This
with it low
tonging from personal use thobugh educationalsand labdratory applicatioris up a l
control and systems development. All ofthis is accomplished aperformance a rifice
or reliability.
1/4 inch) of the K-Buthe dptimaldivision
of system functions thus simplifying system configuration. The motherboard incorporates integral card guides and compatible power connectors which minimizes packaging requirements. Both SKDOS and OS-S/68000 are fully supported allowing efficient system utilization in both single and multi-user applications,

AVAILABLE IN *KIT* FORM

Boards currently in production:

K-BUS	12 Slots, .8" centers PC type power connectors	$129.95
K-CPU-68K	10MHz 68000 CPU, 2 ROM sockets (12 or 16MHz)	$129.95
K-MEM	256K static RAM or 27256 type EPROMs (OK installed)	$ 59.95
K-ACI	2 serial ports with full modem controls (68681)	$ 99.95
K-FDC	Floppy disk controller (up to four 5 1/4 drives)	$ 99.95
K-SCSI	Full SCSI implementation using 5380 chip	$ 99.95
K-DMA	2 channel DMA controller using 68440 chip	$129.95
K-PROTO	General purpose/wirewrap board	$ 39.95
K-xxx-BE	Bare board with documentation for above	$ 39.95

Software:

SKDOS	Single user, editor, assembler, utilities, BASIC	$150.99
OS-9168000	Multi-user, editor, assembler, SCRED, utilities BASIC C, PASCAL, FORTRAN are available	s300.00

inquire about our **UniQuad** line of **68xxx** Single Board Computers.
Quantity *and package discounts available*

Terms: Check, Money Order, Vka, MasterCard-Prices include UPS ground shipment in continental U.S.

lHazelwood Computer Systems

Highway 94 at Bluffton UniQuad™ K-Kits™
Rhineland, MO 65069 . (314) 236-4372

Reader Service #124

to input D on the shifter. Once again (as in using a UART handshake line for auto-baud detection), an H16 line is being used for an unintended purpose. In this case, the timer line acts as a simple l-bit output which is used to programmably "tune" the write cycle timing in case we run into other chips with difficult timing requirements somewhere else in the system.

BCC BUS INTERFACE

The system so far is a complete computer with H16 (and all it includes-timers, UARTs, etc.), memory, and parallel I/O. All that's left is to provide a way for this computer to access boards out on the BCC bus, a 44-pin bus for control applications.

As you'll see, the basic bus interface is pretty simple (the 44-pin limit helps discourage "creeping elegance"). The major challenge is to find a way to remain completely compatible with existing 8-bit I/O boards while at simultaneously supporting thedevelopmentof high-performance 16-bit I/O boards (likely candidates are fast A/D and parallel I/O). Here's the solution and it turns out to be surprisingly easy.

The existing 8-bit bus spec uses a total of sixteen lines for address and data: Multiplexed address/data lines AD0-7 and non-muxed addresses A8-A15. Thus, the obvious place to start is to decide that 16-bit accesses will simply use the same lines for the H16s muxed address/data bus ADO-AD15 (actually A1D0-A16D15, but I explained about renaming thesealready didn't I?). Now, how to decide which typeof buscycle—8 or l&bit-should be run?

The answer is embodied in the remaining logic on page two of the schematic (Figure 7). First, gates in U11 and U7 generate BCC8\ and BCC16\ signals (which identify 8 and 16-bit cycles respectively) based on two inputs-BCC\, which decodes a BCC-bus access in general and A15, which partitions the BCC-bus space into 8 and 16-bit halves. It turns out that all existing BCC-bus I/O boards feature jumper address selection which allows them to be positioned at high addresses (8000H-FFFFH) so we simply adopt the convention that accesses to this range are 8-bit while those from 0000H-8000H are 16-bit. In essence, what was once a 64K byte BCC-bus space hasbecome a 64K word space of which 96K bytes are usable—32K 8-bit ports and 32K 16-bit ports (Figure 12). This is all hooked into the BASIC **INP** and **OUT** statements-if you **INP** from **/OUT** to a port address between 8000H-FFFFH 8-bits will be moved while 16-bits will be moved for accesses to ports OOOOH-8000H.

Having derived BCC8\ and BCC16\ signals, they are used to selectively enable the respective bus control lines driven by U13. Half the chip drives the existing 8-bit control lines (AS\8, DS\8, RD\8 and WR\8) while the other half implements the new 16-bit bus lines (AS\16, DS\16, R/W\16 and a spare—RSVD\16). Notice that since this design only enables U13 during actual BCC-bus cycles, pull-ups (RN2) are required to prevent possible I/O board glitches when the BCC bus is "idle."

These bus control outputs run over to the last page of the schematic (Figure 10) where they hook directly to the BCC bus connector. All that's left to

ATTENTION COMPUTER STORES

Put Circuit Cellar INK to work for you.

Circuit Cellar INK readers are engineers, programmers, consultants, and serious computer technologists. Bring them into your store by selling Circuit Cellar INK.

Circuit Cellar INK's Direct Dealer Sales Program lets you increase your store traffic-increase your sales-increase your profits with minimal risk and up-front investment.

Increased Sales = Increased PROFIT

For information on how your business can become part of the growing Circuit Cellar INK success story, write or call: Circuit Cellar INK
Dealer Sales Program
4 Park Street
Vernon, CT 06066
(203) 875-2199

do is actually enable either the 8- or 16-bit address/data appropriately (with U24, 25, and 26), again using the BCC8\ and BCC16\ signals to decide.

The other gates perform miscellaneous functions. Starting from the top, U6 buffers one of the H16 timer "PWM" (Pulse-Width Modulator) found on the BCC52, and U6 also drives RESET to initializes I/O boards to a known state. Like the BCC180, the BCCH16 implements BCC-bus connections for two DMA channels using DREQ0/1\ (connected to the corresponding H16 inputs and TEND0/1\ (Transfer END-derived from the H16 DACK0/1\ and DONE\ lines). The remaining BCC-bus lines are three interrupt inputs (IRQ0\, IRQ1\, and NMI\), an alternate connection (instead of RS-232) for the async serial port console (TTLSO/SI) and, of course, power (+5V and ±12V) and ground. There are even five lines left unused.

Though it doesn't have a lot of pins or the high price/chip count of the big buses, don't sell this "lean and mean" bus short. You can move 16-bit data at multimegabytes per second and still talk to existing 8-bit boards.

DESIGN REVIEW

It's a good idea after all is said and done to go back to the original design goals for a reality check. You'll remember these goals, outlined at the beginning of the article, revolved around performance, cost, form factor, power consumption, and compatibility.

With a fast 16-/32-bit CPU, up to 768K bytes of on-board memory, and plenty of timers, UARTs, and parallel I/O, the BCCH16 arguably meets the performance goal. Whether the cost is "reasonable" depends on your application (and budget) but it is certainly quite competitive with most (and much less expensive than some) offerings in the same performance range.

As for form factor and power consumption, the BCCH16 performs admirably. The design is packed into a mere 25 chips and the circuit board is the same size as the BCC180. Power consumption, even during full-speed (10-MHz) operation, is in the neighborhood of 1 watt (!)—the exact value depends a lot on whether the board is populated with CMOS versions of the EPROMs, 8255s, and TTL. It can be further reduced by exploiting the H16's low-power modes.

The BCCH16 design is hardware compatible with existing 8-bit BCC I/O boards and has 16-bit I/O capability to boot. Furthermore, conscious decisions have been made (such as organizing the 8255s for 8-bit access and the 8-/16-bit BCC bus scheme) to facilitate BASIC-180 software compatibility. The rest is up to the compiler writers.+

Tom Cantrell holds a B.A. in economics and an M.B.A. from UCLA. He owns and operates Microfuture Inc., and has been in Silicon Valley for 10 years involved in chip, board, and system design and marketing.

Assembled and tested BCCH16 boards are available from Micromint Inc. Call for price and availability. (800) 635-3355, (203) 871-6170

IRS

210 Very Useful
211 Moderately Useful
212 Not Useful

TETHERLINE™

The possibilities are *Infinite.*

Tetherline: a simple, economical system of control and report for Home and Industry.

Tetherboxes are small, die-cast aluminum shielded units, each controlling and/or reporting up to seven devices on an RS-485 Tetherline.

A **T-to-C** (Tetherline to computer) is a PC compatible plug-in short card which passes both Tetherline and X-10 Powerhouse* control and report information to and from the main program through dual-ported ram without interrupts. *X-10 Powerhouse is a trademark of X-10 (USA) Inc

In the background, the T-to-C already handles the FE-485 communications protocol, with error correction and collision detection <u>and</u> X-10 envelope generation for both the one and two-way powerline interfaces, fully opto-isolated.

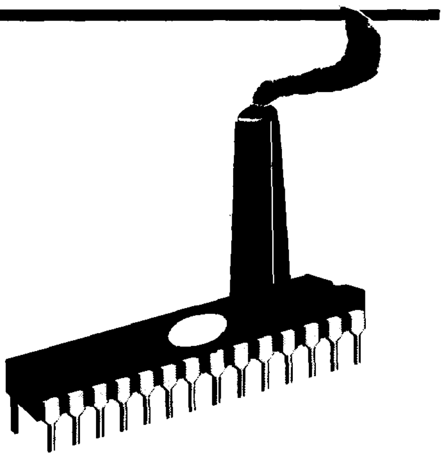
Use any language for your main program or get custom control software from us.

INFINCO

Jim Kurma
(607) **798-9700**

222 Main Street
Binghamton, **NY 13905**

Reader Service #127



Cache Craziness

by Ed Nisley

If you believe the ads and reviews, the recent proliferation of caches in high-end PCs is the best thing since the IC. It seems that no matter what programs you run, no matter what your job, things go better with cache.. .

The recent history of microprocessor **systems** is studded with "breakthroughs" that first appeared in the mainframe world decades ago. This season's PC innovation seems to be cached main memory. Contrary to the advertising excitement, a cache is simply one way to match a fast central processor to relatively slow memory.

Because a cache seems to give **you** something for nothing, the usual caveat applies: something that looks too good to be true probably is. In this column I will explore the cache fundamentals sometimes lost in the glare of publicity.

Before the cache, however, came the wait state.

LYING IN WAIT

The venerable wait state is the simplest way to saddle a fast CPU with slow memory. In fact, it forms the basis for all other methods! If the memory cannot respond by the time the CPU is ready, a circuit freezes the CPU until the data arrives. The duration of the wait is measured in ticks of the CPU clock.

For example, a 33-MHz 80386 (the current heart throbof the Crystal Crazy set)requiresvalid data twoclockcycles (60.6 ns) after it issues a request, a time scale that is slightly beyond the capabilities of affordable DRAM. Each wait state adds 30.3 ns to the access time. Figure 1 sketches the number of wait states required at various memory speeds for a '386; note that the speeds shown are for the entire memory system and not just the RAM chips themselves.

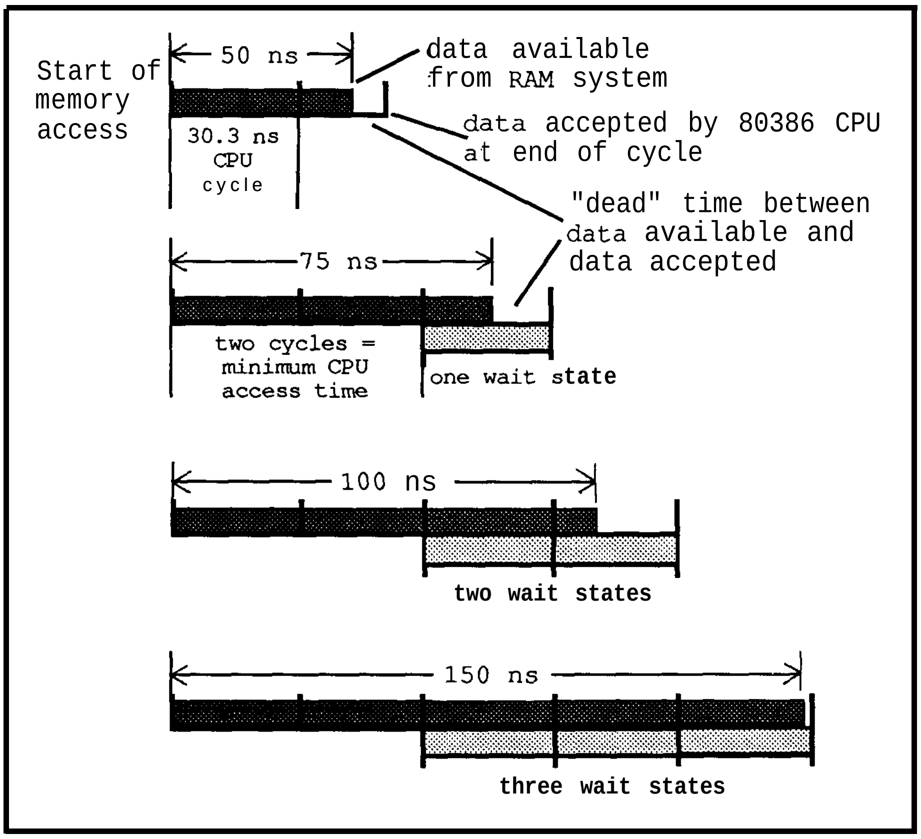

Start of memory access
50 ns
data available from RAM system
data accepted by 80386 CPU at end of cycle
30.3 ns CPU cycle
"dead" time between data available and data accepted
75 ns
two cycles = minimum CPU access time
one wait state
100 ns
two wait states
150 ns
three wait states

Figure 1 — *Waitstatesrequiredfor variousmain memoryspeeds on a33-MHz80386 system. DRAM access times are always faster than overall memory system times!*

Fortunately, '386 instructions do not need data at a continuous 16.5-MHz rate! Although a single wait state degrades performance by 50% on each memory access (requiring three cycles instead of two), the effect is less than half that for a typical sequence of instructions, because the instructions perform some useful work on-chip between memory accesses. Obviously, you can sandbag the results either way by carefully choosing the instruction sequence used for testing.

But wait states, no matter how necessary, do not make good advertising copy, so staticcolumn DRAM or page-mode DRAM appeared. Instead of one or two wait states, all memory accesses after the first within a given range of addresses require no wait states. That first access, however, INTROL

CROSS
DEVELOPMENT
SYSTEMS

• INTROL-C Cross-Compilers
. INTROL-Modula-2 Cross-Compilers
• INTROL-Macro Cross-Assemblers
Provide cost and time efficiency in development and debugging of embedded microprocessor systems

All compiler systems include:
Compiler . Cross-assembler . Support utilities . Runtime library, including multi-tasking executive . tinker . One year maintenance . User's manual, etc.

TARGETS SUPPORTED:
6301/03 • 6801/03 • 6804 • 6805 • 6809 • 68HC11 . 68000/08/10/12 • 32000/32/81/82 . 68020/030/881/851

AVAILABLE FOR FOLLOWING HOSTS:
VAX & MicroVAX; Apollo; SUN; Hewlett-Packard; Gould PowerNode; Macintosh; IBM-PC, XT, AT and compatibles

INTROL CROSS-DMLOPMEM SYSTEMS are proven, accepted, and will save you time, money, effort with your development. All INTROL products are backed by full technical support. CALL or WRITE for facts NOW:

INTROL
CORPORATION

647 W. Virginia St., Milwaukee, WI 53204
414/276-2937 FAX: 414/276-7026
Quality Software Since 1979

Reader Service #129

incurs two wait states. Write accesses are typically slightly slower than **reads,** so they may have a single wait state within the current page and one or two on a different page.

Of course, the number of wait states depends on the relative speeds of the CPU and memory. Page-mode RAM that delivers one-wait-state performance for a particular CPU could run with no waits on a slower CPU or two waitsonafaster processor.

While a CPU may rip through a perfectly tuned chunk of code with zero wait states and stumble over another section with one wait state per access, a bit of perfectly valid arithmeticallowsaclaim of about 0 7 wait states for most instruction sequences. Entering "zero waits" in the feature comparison tables justifies page-mode RAM if all else fails, which may explain some of the ad copy.

There are two troubles with page-mode RAM: the special DRAM chips do not benefit from the awesome economies of scale accruing to standardDRAMsand,byandlarge,itisn't thatmuchofanimprovement formost code. For example, a nontrivial PC program has code and data segments separated by hundreds of kilobytes, so the normal access pattern doesn't benefit from page-mode behavior.

The ideal solution would be something like an adaptive page-mode RAM with fast access to the data and code a program normally uses, while being a lot slower for anything else. While that ideal hasn't been reached yet, a cache comes close.. .at least for some applications.

CACHE BASICS

A main-memory cache is a fast, relatively small memory between the CPU and main memory, with some rather complex control logic monitoring the CPU memory signals. Figure 2 shows one common system configuration; there are at least as many more designs as there are clever designers. Don't confuse this type of cache with thesoftwarecachesusedtoboosthard diskperformance; theyarecompletely separate animals.

The overall operation of a cache is surprisingly simple. When the CPU reads data, the cache control logic checks to see if the data is in the cache. If so, a "cache hit" occurs and the CPU gets the data from the cache without delay. If not, a "cache miss" occurs and the CPU is frozen in wait states until the data arrives from main memory. Generally, the cache logic stores the new data into the cache at the same time the CPU gets it.

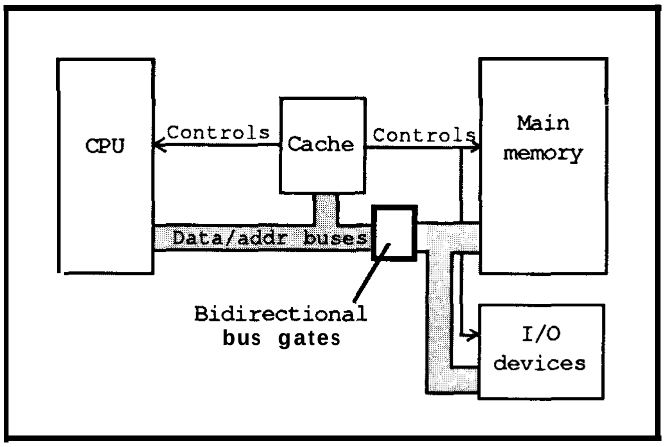


Figure 2—*Block diagram for a typical main memory Cache system. Note that all data buses are the some size.*

Because there are two buses shown in Figure 2, the CPU can get data and instructions from the cache while the I/O devices update main memory through the DMA controller. This allows higher performance because two things can be occurring at once, but introduces more cache control complexity: what happens when an I/O device updates a data value that is also in the cache? As you will see, caches are simple until you start to think about the details!

Incidentally, if you have followed the EISA drama so far, Figure 2 should look familiar. Substitute "proprietary memory bus" for the CPU-to-cache bus and "AT I/O expansion bus" for the memory-to-I/Obus and you have the essentials of an EISA system block diagram. The EISA **memory** bus runs

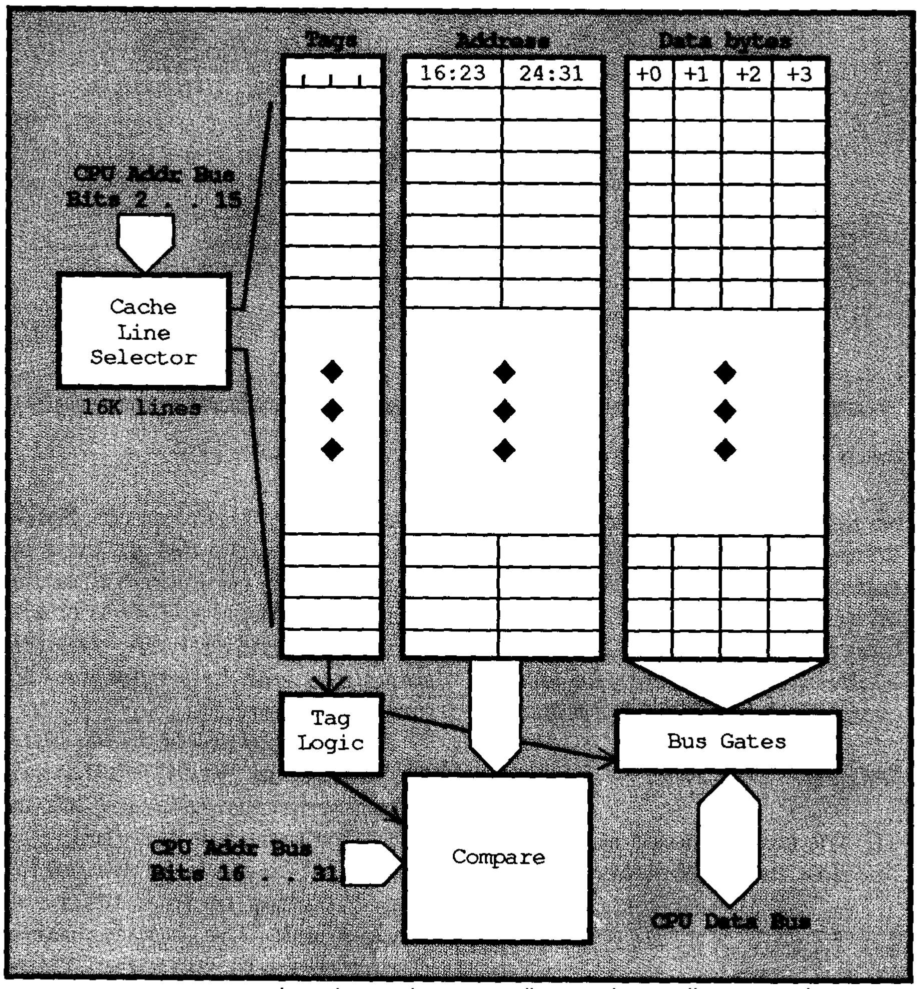


Figure J-Cache *data paths. The cache has16K lines and uses direct mapping to one of 64K main memory locations. Each cache line contains four data bytes.*

at full speed, while the I/O bus is throttled down for compatibility with the I/O devices. I will leave it to you to infer the performance effects of putting memory on the "AT I/O bus" section.

HITTING THE CACHE

Because the cache memory is fast, any data found in it can be returned to the CPU with no delay. But, because it is small, there must be some data in main memory that isn't in the cache. The cache control logic's job is to guess what data the CPU will ask for and try to have it in the cache memory, ready for use.

Themost important figureof merit for a cache is the "hit ratio" which measures the success of the cache logic's guesses. The hit ratio is simply the number of zero-wait accesses divided by the total number of accesses. For example, if the CPU made 100 accesses and 80 had zero wait states, the hit ratio would be 0.8 or 80%. Obviously, the higher the hit ratio, the better.

The dark side of the hit ratio is the number of accesses that are not in the cache. For example, that 80% hit ratio sounds pretty good unless each cache miss requires five wait states. In that case, the CPU "sees" memory with an average of one wait state. Calculated out, 100 accesses x 0.2 miss ratio x 5 waits/miss = 100 waits, or one per access!

Al though five waits per miss may seem high, Figure 1 shows that even a peppy 100-ns memory system takes four cycles for each access. Ordinarily only the last two cycles count as wait states, but if the cache control logic detects a miss it must start an access from scratch during the second cycle; all four main memory cycles count as wait states. The overall access will take six cycles!

Although you might think that the control logic could start simultaneous accesses in both the cache and the main memory, **then** cancel the main memory access after a cache hit, the situation isn't that simple. The main memory logic usually cannot cancel an access, so the circuitry would be tied up for four cycles anyway; a second access would have to wait for the memory to recover from the first, canceled,access. Also, recall that DMA accesses may be running on the other bus, so any additional CPU accesses will clobber I/O performance. Ah, tradeoffs!

Takealookat theadsforPCswith main memory caches. Try to find the number of wait states per cache miss, but don't be surprised if it doesn't appear anywhere. For extra credit, deduce the number of waits from the main memory speed if you can find that number. Interesting, no?

RUNNING THE LINE

A good-sized PC cache has 64K bytes of fast RAM. Current IBM PS/2s are limited to 16 MB of main memory because the DMA controller generates only 24 address bits, but future hardware will probably remove this limitation and allow up to 4 GB of RAM (talk about sticker shock!). Simpledivision tells you that thereare 64K bytes in main memory for each cache byte!

It makes sense to divide the cache into "lines" that can be filled by one main **memory access** ra ther than worry about individual bytes. For the 80386, the natural cache line is four bytes wide. The number of lines in thecache is simply the total cache size divided by the line size; in this case the cache has 16K lines.

The first byte in each line has a main memory address with two low-order zeros. The 80386 CPU can access four-byte words starting at any address, so parts of the result can occupy two lines. I'll ignore the obvious complications, but you should note that the cache must tuck the right bytes into the right parts of the right lines regardless of what the CPU is actually doing.

Figure 3 shows a 64-KB cache divided into 16K four-byte lines. In addition to the four data bytes in each line, the cache also stores the high-order two bytes of the main memory address used to access the data and several control bits, known as "tags" in the vernacular. The control logic uses the stored address and tags to identify cache hits.

When the CPU accesses main memory, it issues a 32-bit memory address. The cache control logic uses bits 2-15 of that address to select one of the cache lines. The data in that line will be a cache hit only if the address stored with the data matches bits 16–31 of the current main memory address and the tags are set appropriately. If everything clicks, the bus buffers gate the data onto the CPU data bus within 60.6 ns of the original request!

While the cache RAMs storing the data bytes must be fast enough to return the data bytes with no wait states, the tag and address RAMs must be faster still. Comparing the addresses requires several levels of logic gates, and bus buffers have some turn-on delays, so those RAM chips may have access times below 20 ns. There is obviously a premium for fast and clever logic design in this application.

```
          MOV    CX,0          ; 64K loop count
          MOV    DS,1000h      ; source segment 1
          MOV    BX,2000h      ; source segment 2
          MOV    DX,3000h      ; destination segment
          MOV    SI,0          ; byte offset
again:    MOV    AL,DS:[SI]    ; get one source byte
          MOV    ES,BX         ; get another
          ADD    AL,ES:[SI]
          MOV    ES,DX         ; drop result byte
          MOV    ES:[SI],AL
          INC    SI            ; step offset pointer
          LOOP   again         ;  and repeat
```

isting 1 *-A program loop* **that** *tangles with the cache and loses big.*

TAGGED LOGIC

The tag bits stored in each cache line describe the contents of the line. While the meanings of specific bits depend on the cache design, there are some functions common to all caches. The cache control logic is responsible for updating and analyzing these bits, sometimes with help from the CPU and its firmware.

Most important of all tag bits is the "In-use" (or "Valid") bit that indicates whether that cache line has any data. The cache is initially empty, so there must be a way to prevent hits from occurring on lines filled with random junk. The cache is disabled during system startup so the CPU can clear all the tags bits. Once the cache is activated, the control logic sets the In-use bit when it loads the line from main memory.

Up to this point I have glossed over what happens when the CPU writes data into main memory. There are basically three ways to handle this situation: write the data into both the cache and main memory simultaneously, write it into the cache and promise to update main memory later, and write it into main memory while invalidating the matching cache line. All three methods are used in various computers because there is no clear-cut winner.

Simultaneous updates ensure that both copies of the data are the same, but require access to the main memory bus during the entire write cycle. Updating main memory after an into-cache write is very fast, but requires clever logic to ensure that I/O devices do not get the "stale" data from main memory. Invalidating the cache entry is easy, but means that the next access

The DA/M™

A Low Cost Data Acquisition System

- 8-A/D Channels, 8-Digital I/O Channels, l-Counter/Timer.
- Runs on 12 to 24 VDC.
- 15 Systems or 255 points per RS232/RS485 Communications Port.
- Connect to Sensors that output 0-4.59V, 0-5V, 0-10V or 4-20Ma or add optional onboard amplifiers for lower signal levels.

DA/M 100-O	DA/M System	$200.00
DA/M 100-l	RS232-RS485 Converter Cable	$20.00
DA/M 100-2	ROM/RAM Piggy Back Card	$50.00
DA/M 100-3	RS232-RS485 Converter Unit	$100.00
DA/M 100-5	Screw Terminal Prototype Card	$100.00
DA/M 100-7	Isolator/Relay Card	$160.00
DA/M 100-S	Opto-22 Interface Card	$140.00
DA/M 100-9	RAM/ROM/Real Time Clock Card	$180.00

DA/M

DATA ACQUISITION AND MANAGEMENT CORPORATION LTD.
#140 17303 102 Avenue
Edmonton. Alberta, Canada T5S 1J8

Phone 1-403-486-3534
Fax 1-403-406-3535

Reader Service #119

to that data will always be a cache miss.

Ah, more tradeoffs!

The second method requires a "Dirty" bit to mark cache entries that must be copied back into main memory; the cache control logic (and sometimes the CPU firmware) uses that bit to schedule updates. Some implementations use a FIFO holding the new data and addresses so the main memory can perform updates when it's not doing anything else.

In the cache I've been describing so far there is only one possible spot for each main memory byte: in the line selected by address bits 2-15. Such a cache is called "direct mapped" for this reason. Another cache design, called a "set-associative" cache, has several possible lines for a given address; typical set-associative caches allow two or four locations for each main memory byte.

You can think of a set-associative cache as having two (or four) tag, address, and data RAMs. Each CPU access activates all of the cache RAMs in parallel; only the line holding a matching address will be a hit and the others will be misses. In these designs, the cache control logic must have a way to select one line from the set to hold new data. Although some caches pick a line at random (honest!), the Least Recently Used algorithm is more common.

For set-associative caches, an "LRU" bit is set whenever the line is accessed. Periodically, the cache control logic examines the LRU bits and invalidates any line which hasn't been accessed since the last examination. New data always goes into an invalid line unless all lines in the set are valid (they are all "recently used"); in that case the logic can pick any line.

Top-dollar mainframe caches have an additional bit that disables that line. This "Error" bit is set when the system detects an error in the cache RAM. PC-sized caches solve this problem in two ways: they don't have parity check bits (so diagnosing a failed cache RAM is difficult) and you replace the entire chip. If you had to replace the entire system board to fix a single-bit error, you would clamor for better cache error checking, too!

STRIDING INTO TROUBLE

Now that you know all about how a cache works, we can explore how it affects program performance. Despite some advertising claims, there are some things that do not go better with cache!

The ideal situation occurs when all of the instructions and data are in the cache. Conversely, if every memory reference is a cache miss, the system will run slower (sometimes much slower) than it would without the cache. Because the cache hardware design is a given in a particular machine, your program and operating system behavior determine the cache hit ratio.

Because the cache shown in Figure 3 has four bytes in each line, the data in the third statement of this program fragment will always have a cache hit (assuming no interrupts get

SA-20 EPROM PROGRAMMER

Stand-Alone 8 Gang Programmer $750

Programs 2716, 2732, 2764, 27256, 27512., 27C010, 27C020, 27C040	Quick Pulse Algorithm	Menu-Driven Operation
	Binary, Intel Hex and Motorola S Formats	A9 Intelligent Identifier
Parallel I/O or RS232		Full 1 year warranty

Call Toll-Free I-800-248-5487

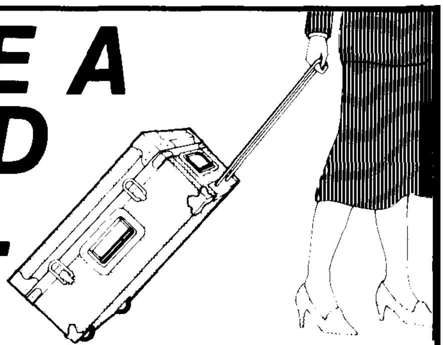
Coindata, Inc.

3026 Owen Dr., Suite 106, Antioch, TN 37013

In TN (615) 793-9142 FAX (615) 793-9177

Reader Service #111

TAKE A LOAD OFF...



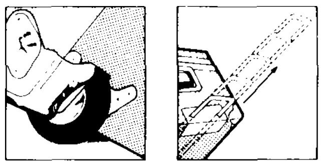
A rugged CABBAGE CASE? lined with plenty ot foam for your equipment can

TAKE A LOAD OFF YOUR MIND

when you've got to travel.

TAKE A LOAD OFF YOUR BACK

with our exclusive tilt-wheels and extension handle option.



UNLOAD ON US!

Call or write to tell us about your shipping or carrying problems

WE HAVE SOLUTIONS!

CABBAGE CASES

CABBAGE CASES, INC.
1166-C STEELWOOD ROAD
COLUMBUS, OHIO 43212-1356
(614) 466-2495 (600) 888-2495

Reader Service #108

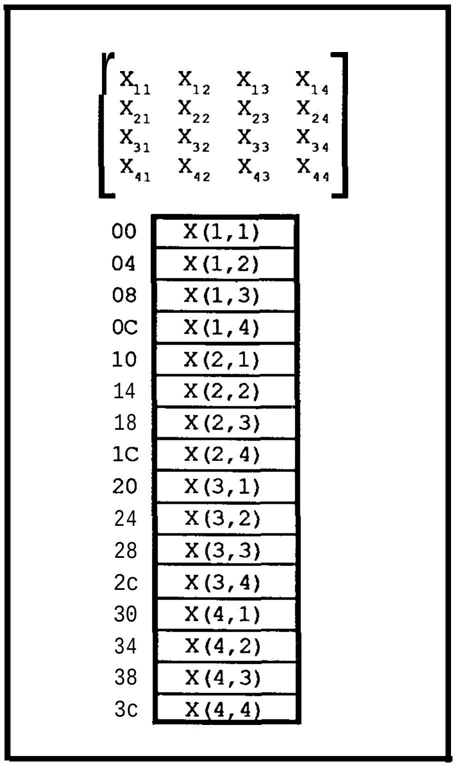


Figure 4-Array ofshortreals *and the corresponding* **storage** *layout.*

control between the second and third lines!):

```
MOV     SI, 1000h
MOV     AX, [SI]
MOV     BX, [SI+2]
```

While the natural length of a cache line is the processor's memory bus width, longer cache lines can improve the hit ratio by fetching more data on each cache miss. Because most programs use data and instructions more or less sequentially, a miss is (usually) the start of a series of references to successively higher addresses. If that is true for your programs, longer lines will improve the hit ratio by preloading more of the information your code is about to use.

However, longer lines imply a wider interfacebetween thecache and main memory, or several accesses to fill a single cache line on every miss. If that interface must be shared with the I/O system, some complications arise in getting the right data on the right part of the bus; it is rare to see a four-byte I/O device! Adding a separate bus for the cache implies that there will be three system buses and a dual-ported main memory, so you can see that the choices are not at all simple.

Loops and caches seem to be made for each other, because after the first pass all the instructions should be in the cache and the data may benefit from preloading. Usually that is the case, but the code in Listing 1 shows what can happen to the unwary programmer working on an image processing application. In this example, the code is adding the contents of two 64K-byte buffers and storing the result in a third. All three storage references use the same **cache** line, so every access is a **miss** despite the fact that the cache is busily preloading three bytes for every one actually used!

That loop also has the interesting side effect of flushing everything else out of the cache. Regardless of the state of the cache when the loop starts, it will be full of image data at the end, with the possible exception of a few loop instructions that may sneak back in after the loop data addresses run through their line.

A similar problem arises with engineering and scientific calculations on large arrays. Figure 4 shows an array of short (four-byte) real numbers and the corresponding storage layout. Each array element will occupy one cache line and most compilers will align the elements on a four-byte boundary, so each cache line holds one element.

While the small array in Figure 4 fits into the cache with no trouble, a much larger array may flush the cache every time it's accessed. Even if one array fits, the first pass through it will be all misses. Worse, if the **code** makes only one reference to it, those misses are wasted effort and slow the program down while the cache recovers. This sort of situation occurs often enough that some workstations can disable their cache under program control.

Multitaskingenvironmentsintroduce another complication, Suppose your program is running a hot, tight loop from the cache. At that point, the operating system switches to another task. The new code just happens to run a loop that flushes your code and data out of the cache. Your code will return the favor when it gets control on the next time slice, and all those cache misses start adding up.

The process of flushing and reloading the cache is called "thrashing" and can cause significant performancedegradation. Unfortunate there isn't much you can do about it because your program is at the mercy of all the other processes using CPU time. This effect does not show up under DOS, because the only "multitasking" is the occasional interrupt which uses only a few dozen instructions. How caches will perform under OS/2 is open for discussion and experimentation.

Incidentally, the term "thrashing" also applies to virtual memory systems subject to similar contention. In that case, however, the performance decrease is even worse because the system must hold the flushed information in a disk file!

PRESCRIPTIONS

The truth about caches is somewhere between the hype you will see in ads and the somewhat pathologic cases described above. It behooves you to find out as much as you can about your machine's cache and program accordingly: avoid stepping on your own feet whenever possible!

If you know that your machine has a cache, but the manufacturer's literature **does** not tell you much about it, it will be worthwhile to write some test programs to exercise the cache control logic and find out how it behaves under stress. The results of those tests will give you a good idea how to structure your code to best advantage. ✣

Ed Nisley is a member of the Circuit Cellar INK engineering staff and enjoys making gizmos do strange *and wondrous* things. *He is, by turns, a beekeeper, bicyclist, Registered Professional Engineer, and amateur raconteur.*

IRS

213 Very Useful
214 Moderately Useful
215 Not Useful

FROM THE BENCH

Gentlemen, Start Your Engines

by Jeff Bachiochi

The automotive industry perpetually seduces us with glimpses into the future. A future in which computers will infiltrate the American dream. The automobile has been the American dream ever since its introduction over 80 years ago.

When we dream of how a computer controller might be used in an automobile we conjure up thoughts of video displays and keyboards, dash mounted and within easy reach. Not only will they aid us in planning the best route to our destination, but also serve as chauffeur and autopilot.

Not many of us realize that the march toward realizing this dream has already begun. The first applications of solid-state devices in our vehicles were not computer controllers but substitutes for mechanical parts which wore out and had to be replaced. Electronic ignition replaced the mechanical points in the distributor with a Hall-effect sensor and magnet for noncontact dwell-angle timing of the ignition's firing. Windshield wipers of the past were limited to one of a few speeds, often times creatingtheneed toconstantlyturn themonandoffinlight rain. Today's wiper control, which offers adjustment of the off time between wipes, is regulated by an RC time constant.

The next generation of integration is truly computer controlled and is already in production. Full electronic control of the ignition and fuel injection system is not just a dream, but is being played out in real life by manufacturers such as Audi. Let's take a look under the hood of an Audi 5000S equipped with 2.3-liter engine using the CIS-E III Engine Control System.

SHARING THE BURDEN GETS THE JOB DONE

The CIS-E III Engine Control System is made up of two computer controllers: the Ignition Control Unit and the Fuel Injection Control Unit. A serial link between the two units allows the exchange of accumulated data. Each unit receives input from a variety of sensors associated with its job function. All inputs and outputs are checked for valid data by self-diagnostic troubleshooting which stores faults within system memory. These faults can be displayed as specific on/off patterns of an indicator light in the instrument cluster.

THE SPARK OF LIFE

The Ignition Control Unit, or ICU, is responsible for triggering the ignition coil. It receives data directly from the knock sensor, distributor, coolant temperature sensor, throttle valve switches, and altitude sensor.

The knock sensor is a piezoelectric crystal mounted next to cylinder #3. Vibrations in the engine are transferred to the crystal which creates a small voltage. By monitoring this voltage the ICU can sense ignition knock and retard the ignition timing to prevent it.

The distributor has no need for centrifugal or vacuum advance. The ICU can calculate and adjust the timing based on engine speed. The engine speed and crankshaft position is picked up from a trigger wheel with five teeth, one for each of the five cylinders. A Hall sensor sends a voltage pulse to the ICU indicating the appropriate firing point for each cylinder as the engine turns the distributor shaft past the sensor. The total pulses per minute divided by five (number of cylinders) equals the crankshaft speed in RPMs.

The coolant temperature sensor consists of two independent NTC (negative temperature coefficient) resistors.

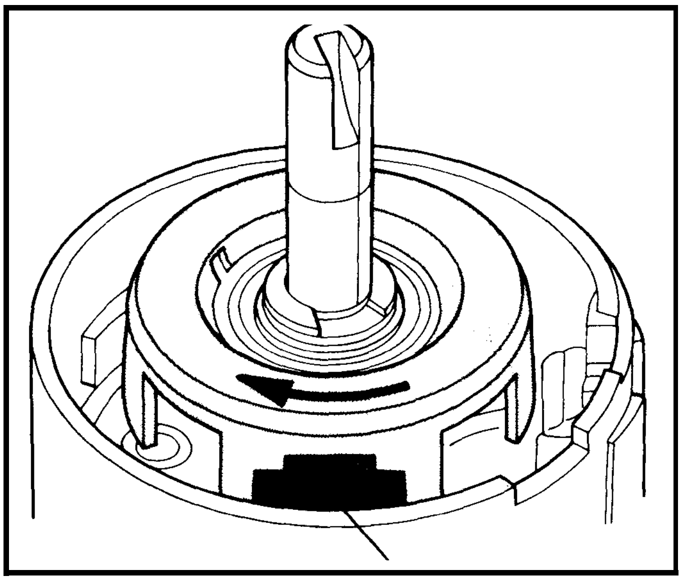


Figure 1 -Distributor's *Hall-effect sensor.*

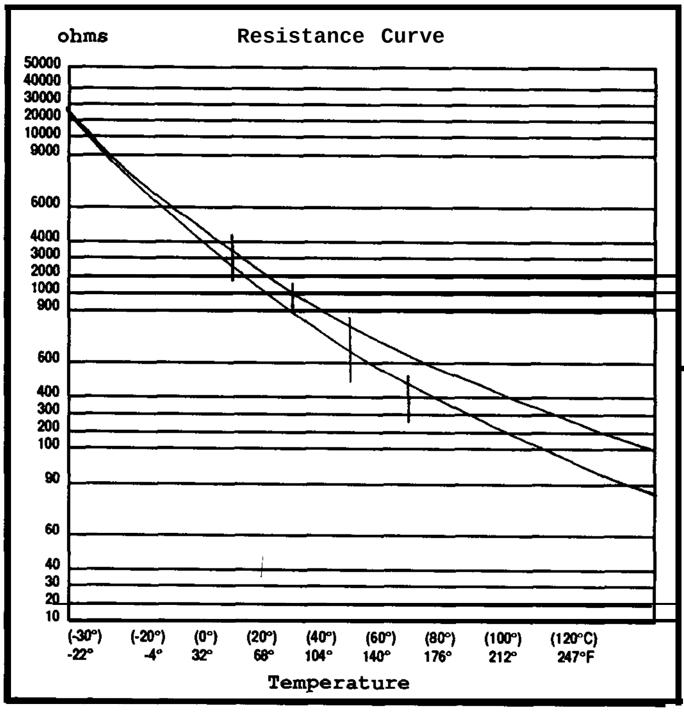


Figure 2—*NTC resistor resistance versus temperature.*

One NTC resistor provides output for each of the control units since engine temperature is necessary for calculating both the correct ignition timing and fuel mixture.

Throttle switches, both idle and full throttle, are simple mechanical switches which detect the throttle position. The idle switch indicates when the throttle is closed and affects the idle stabilizer value, deceleration fuel shut-off, and ignition timing map. The full-throttle switch indicates that the throttle is in the last 10 degrees of rotation and affects the full-throttle enrichment and ignition timing map.

The altitude sensor is a barometric chamber which will expand or contract with a change in atmospheric pressure.

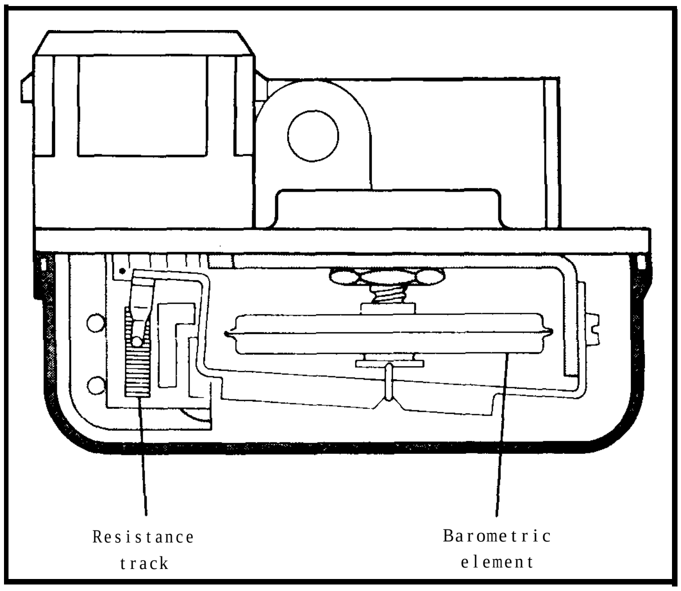


Figure 3—*Altitude sensor.*

This movement operates a variable resistor and sends a voltage to both control units for high-altitude adjustments.

The ignition coil is driven from a Darlington power stage which switches the primary of the ignition coil on and off. The secondary of the ignition coil produces a stepped-up voltage which is routed through the distributor to the appropriate cylinder's spark plug. The ICU triggers the power stage to provide the correct ignition timing based on engine speed and engine load. A look-up table stored in the ICU holds precalculated ignition timing advancements based on RPM and load.

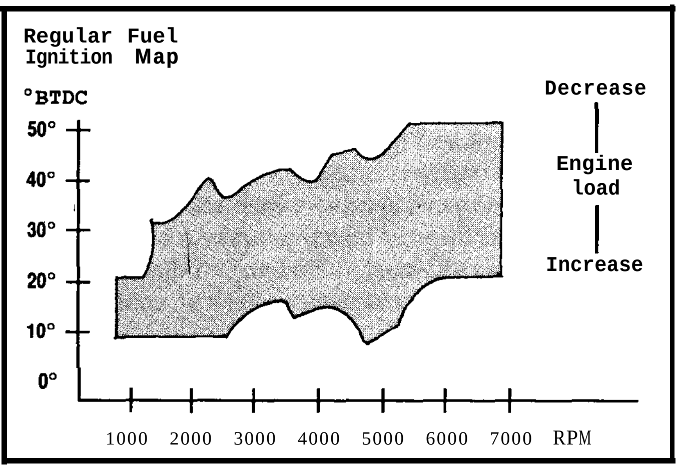


Figure 4—*Ignition map.*

Actually, there are two ignition maps programmed into the ICU: one for regular fuel and one for premium grade. The ICU uses the regular fuel ignition map when the engine coolant temperature is less than 149°F or the knock sensor indicates repeated ignition knocking. Knocking is regulated on a cylinder-by-cylinder basis, since the ICU can control the ignition timing of each cylinder independently. If a knock in one cylinder is detected, its ignition timing will be retarded. The timing will be retarded 3.4 degrees at a time (maximum of 12 degrees) until the knocking stops. The timing will then be advanced by 0.54-degree steps to bring it back to the normal look-up value.

THE HIGH-OCTANE DIET

The Fuel Injection Control Unit, or FICU, controls the cold-start valve, idle stabilizer valve, differential-pressure regulator, and carbon canister shut-off valve. It receives input data from the oxygen sensor, coolant temperature sensor, throttle valve switches, altitude sensor, and air sensor potentiometer.

The oxygen sensor located in the exhaust system monitors the oxygen content of the exhaust gas. The sensor has its own heating element which helps compensate for large temperature changes as in initial engine warm-up. Data from the oxygen sensor assists in determining the correct air/fuel mixture.

The air sensor potentiometer is a variable resistor which is positioned by an air sensor plate. This plate is

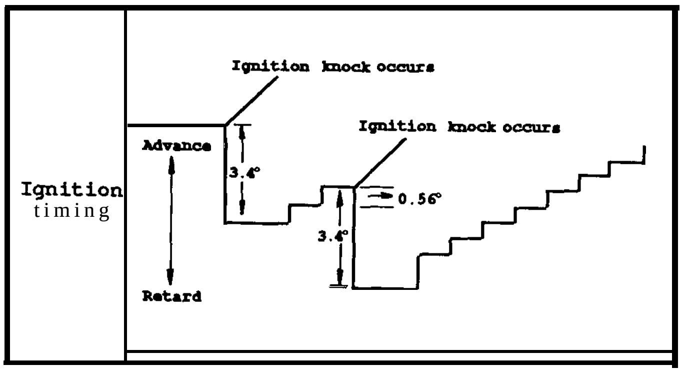


Figure 5—*Ignition knock regulation.*

moved by the flow of air and helps determine cold-acceleration enrichment and ignition timing.

The cold-start valve is a solenoid-controlled fuel injector. The solenoid opens a valve which allows additional fuel to be injected directly into the intake manifold. The FICU permits the cold-start valve to open during engine cranking for **up** to **11** seconds dependent on coolant temperature.

The idle stabilizer valve lets air bypass the throttle valve. This bypass path is constricted by a rotating valve that is **opened by energizing a motor which rotates the valve, counteracting a return** spring. The motor's DC voltage is PWM (pulse-width modulated) enabling the valve to partially open. The FICU adjusts the idle stabilizer valve if the idle throttle switch is on to maintain about 1000 RPM for a cold engine (down to about 720 RPM for a warm engine).

The differential-pressure regulator manages the fuel flow in the lower chamber of the fuel distributor. A current passing through the regulator's coil varies the valve's opening. This adjusts the rich/lean mixture of the fuel flow to the injectors. The FICU controls the mixture by passing a positive current through the differential pressure regulator to enrich the gas/air mixture and a negative current to reduce it. The mixture is increased during engine cranking to a level determined by coolant tempera-

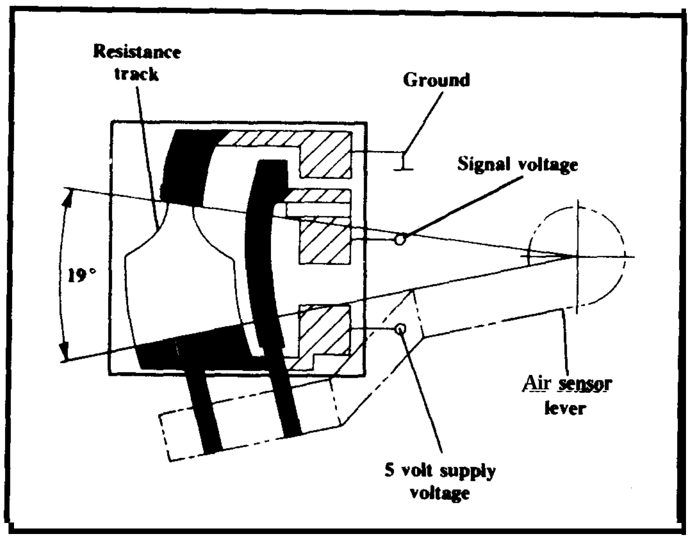


Figure 6—*Air sensor potentiometer.*

New, Improved!

SIBEC-II

The ideal solution for embedded control applications and stand-alone development.

- *Intel 8052A H BASIC CPU*
- *Serial printer output and 5, 8 bit I/O ports*
- *5 in.^{2} prototyping area*
- *Memory: 8K RAM, expandable to 128K*
- *Power requirements: 5V. DC @ 300 ma. only*
- *PROM progammer; ZIF socket for 2764 or 27128 EPROM*
- *Interrupt handling capability*
- *Built to exacting standards and warranteed*
- *Still only $228.00 including documentation (quantity 1)*

Inquire about our **PDK51** 8051-8052 product development kit for the IBM-PC/XT/AT: **$595.**
Our **BXC51** 8051/8052 BASIC compiler: **$295.**

Call now! 603-469-3232

Binary Technology, Inc.
Main Street • P 0 Box 67 • Meriden, NH 03770

Hobbyists, experimenters, R & D groups!

8031 μController Module

Our 8031 μController Module forms the core of a complete prototype or one-of-a-kind device.

We include a high quality, silk screened printed circuit board populated with an 8031 microprocessor, 11.059 MHz crystal, 74LS373 Address Latch, capacitors, communications and I/O headers and a 2764 EPROM with demonstration software. The module also contains a socket for a MAX232 level converter to provide an industry accepted serial communications port if your project requires it.

Need development software? We carry cross-assemblers, simulators and disassemblers from Pseudocode for the 8051 family.

Send $39.95 plus $3.00 S&H per μController module
Illinois Residents Add 6.25% Sales Tax

Cottage Resources
Suite 3-672, 1405 Stevenson Drive
Springfield, Illinois 62703
(217) 529-7679

Reader Service #114

CIRCUIT CELLAR INK's

ADVERTISER'S INDEX

Reader Service Number	Advertiser	Page Number
101	Ail Electronics	71
102	Alpha Products	C4
103	American Advantech	70
104	Avocet	C2
105	Berry Computers	4
106	BG Associates	30
107	Binary Technologies	64
108	Cabbage Cases	59
•	Ciarcia Design Works	79
109	Circuit Cellar	C3
110	Circuit Cellar	C3
111	Coindata	59
•	Collins Associates	79
113	Computerwise	72
114	Cottage Resources	64
117	Covox inc.	72
118	Datarx	13
119	Dycor industrial	58
120	Electronic Energy Control	43
121	Engineers Collaborative	6
122	Epoch Data	79
123	Express Circuits	38
124	Hazelwood Computer	52
125	Heath Co.	33
126	Hogware	42
127	infinico, Inc.	54
128	innotec	67
129	Introl Corp	56
115/116	JDR Microsystems	45
131	Koiod Research, inc.	52
132	Laboratory Microsystems	11
133	Lear Com Company	79
134	Logical Systems	13
135	LTS/C Corp	49
136	Markenrich	28
137	Meredith Instruments	79
138	Micromint	17
139	Micromint	61
140	Micromint	71
141	Micro Resources	6
142	Ming Engineering	38
143	NOHAU Corp	67
144	Omation	11
145	Paradigm Systems	11
146	Parallax, Inc.	25
147	PseudoCorp	38
148	Quinn-Curtis	14
149	R & D Electronics	70
150	Real Time Devices	35
•	Ryle Design	79
151	Silicon Alley	44
152	Tab Book Company	7
153	Thinking Tools	14
154	Timeline	39
•	Tinney	24
155	Unkel Software	19
156	Witts Associates	79

HAJAR ASSOCIATES
NATIONAL ADVERTISING SALES REPRESENTATIVES

NORTHEAST
Lisa D'Ambrosia
Tel: (617) 769-8950
Fax: (617) 769-8982

MID-ATLANTIC
Barbara Best
Tel: *(201) 741-7744*
Fax: (201) 741-6823

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

MID WEST
Nanette Traetow
Tel: (312) 789-3080
Fax: (312) 789-3082

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

SUBSCRIPTION PROBLEMS?

If you have problems with your subscription (delayed or missing issues, change of address, or questions on renewals). call the Circuit Cellar INK Subscriber Service Line at (914) 628-0885 or write:

Circuit Cellar INK
Subscriber Service Dept.
P.O. Box **2099**
Mahopac, NY 10541

IRS
INK Rating Service
How useful is this article?

At the end of each article and some features there are J-digit numbers by which you can rate the article or feature.

Please take the time to let us, at Circuit Cellar INK, know **how** you fell our material rates with you. Just circle the numbers on the attatched card.

ture and cranking speed. After start-up, enrichment is reduced as the coolant temperature increases. During acceleration the mixture can also increase based on the air sensor potentiometer, coolant temperature, and engine speed.

If the full-throttle switchindicates that there isa driver with a lead foot behind the wheel, the FICU will increase the fuel mixture contingent on the output of the altitude sensor. However, during deceleration the FICU sends a large negative current to the regulator cutting off fuel flow entirely to the injectors. This reduces exhaust emissions and fuel consumption.

The carbon canister shut-off solenoid blocks the purge line from the carbon canister. If the fuel vapors pass into theairintake when theengine is turned off, they may cause starting problems. The FICU will not allow the vapors to be purged until the engine is running.

SELF DIAGNOSTICS

Both the ICU and FICU constantly monitor the systems' sensors, wiring, and input signals. Either of the control units can flash the fault indicator light on the instrument cluster. The only fault indicated without accessing the fault memory is a problem with the knock sensor or fully retarded ignition timing due to ignition knock.

The fault memory stores **any** other faults and attempts to use a nominal value as opposed to the out-of-tolerance value actually found. The fault memory holds a four-digit code which indicates the probable location of the fault. The code, indicated through a flashing fault light on the instrument cluster, is accessed by inserting a spare fuse into the fuel pump relay for four seconds. The number of flashes equals the digit of the code number. Reinserting the fuse for an additional four seconds flashes the next digit. Note that this must be done after warming up the vehicle and leaving the ignition on. The ICU will report any faults first, then the FICU.

The control units can also generate signal outputs. To enable the output mode the fuse must be inserted into the fuel pump relay before the ignition is turned on. The output signals can be sequenced similar to the fault memory display. Inserting the fuse for four seconds will step

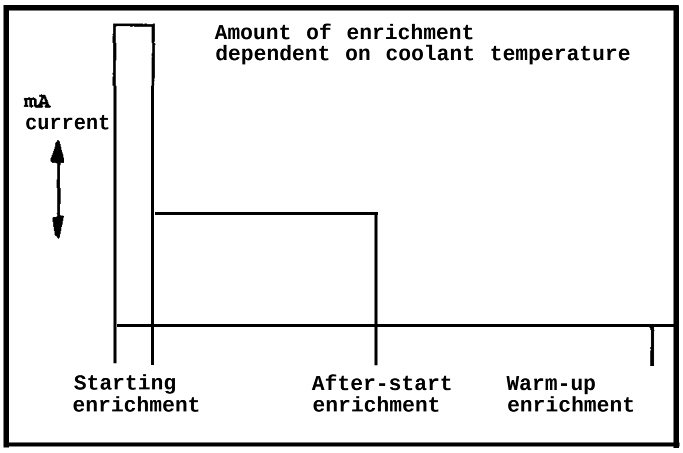


Figure 7-Mixture *enrichment.*

Code	Location of Fault
1111	Ignition control unit or fuel injection control unit
2121	Idle switch
2122	Engine speed signal or Hall sender
2123	Full-throttle switch
2141	Knock regulation
2142	Knock sensor
2223	Altitude sensor
2232	Air sensor potentiometer
2233	Reference (supply) voltage for air sensor potentiometer and altitude sensor
2312	Coolant temperature sensor
2341	Oxygen sensor control
2342	Oxygen sensor
4431	Idle stabilizer valve
4444	No faults stored in memory
0000	End of diagnosis

Figure 8—*Fault* **input codes.**

through the output codes by flashing the fault light on the instrument cluster. To enable a specific output, sequence through the codes until the one of interest is displayed, then close the full-throttle switch.

SURGERY NOT REQUIRED

Each control unit contains its own micro. Decentralizing tasks keeps each task loop short. Real-time control is handled more effectively by not over-burdening the processor. Let's take a look at one of the control units.

A peek inside the ICU will drop your jaw a few inches. Two double-sided (components on both sides, friends, not just traces) circuit boards are connected by a flexible 25-pin connector. Each board measures about 2" x 4". A Siemens processor, three additional DIP ICs, a full-size crystal, and two dozen other discrete parts are mounted on the top side of the boards. On the other side are eight ICs and over 150 (!) other discrete parts all surface mounted. The two boards are folded like a closed book and enclosed within a weather-tight plastic enclosure.

The total engine control system, as we've seen, receives input from a number of sensors: vibration: crystal; RPM: Hall-effect; temperature: NTC resistor; throttlestatus: mechanical switches; air pressure and speed: mechanically linked variable-resistive elements; and oxygen content: O_2 element. The system sends outputs to a variety of controlling devices: ignition timing: power Darlington driver;and fuel supply: solenoid valves (includingplunger-

Step	Code Displayed	Item Checked	Operating Cycle
#1	**4341**	**Differential** pressure **regulator**	10mA current flow to regulator when full-throttle switch is closed
#2	4343	Carbon canister shut-off solenoid	Clicks ON and OFF when full-throttle switch is closed
#3	4431	Idle stabilizer valve	Clicks when full-throttle switch is closed
#4	4443	Cold start valve	Clicks ON and OFF for a maximum of 10 seconds when full-throttle switch is closed

Figure 9—*Fault output codes.*

type on/off, rotary-type PWM, and bidirectional, bipolar type). Simple yet rugged sensors and valves are essential to this engine control system. Self-diagnostics close the loop by recognizing faulty parts. Even though fault diagnosticscannot indicate all problems, they will in most cases point a competent service technician in the right direction. After all, finding the actual cause of most problems takes more time than fixing it.

The next time you drive your car think about the advancements we all take for granted. You may not own an automobile that remembers your comfort specifications and adjusts the seat, steering, and side mirrors automatically. It may not be able to calculate MPG and ETA. No matter what you paid for it, I'd be willing to bet the next time the weather sprinkles on you, you'll think about intermittent windshield wipers a bit differently. Perhaps you'll even get a clearer view of things to come. ✣

Jeff Bachiochi (pronounced "BAE-key-AH-key") is a member of the Circuit Cellar INK engineering staff. His background includes work in both the electronic engineering and manufacturingfields. *In his spare time* Jeff enjoys *his family, windsurfing, and pizza.*

Innovations like these help to make today's technology morecost-effective, reliable, and easier to use. Please share your favorite ideas, chips, and circuits with others.

We will pay $25 for any **From the Bench accepted** for publication. All submissions should be typed, double-spaced, and include neatly drawn schematics or Schema configuration, library, and page files.

Indude a stamped, self-addressed envelope large enough to hold everything if you wish materials that have not been accepted to be returned.

Submit to:

From the Bench
c/o Circuit Cellar INK
P.O. Box 772
Vernon, CT 06066

IRS

2 16 Very Useful
2 17 Moderately Useful
218 Not Useful

ID160/161

Powerful PC-based Logic Analyzer for only $595

● iOO MHz model for $745

- 50 MHz Sampling Speed • Multi-level triggering
- 8K trace buffer • 32-channel capability
- Event Timer/Counter • Performance Histograms
- Hardcopy output • Disassembles 8-bit micros
- and much more ! • Satisfaction Guaranteed

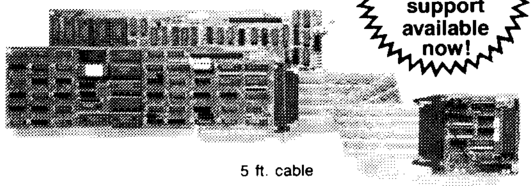
INNOTEC DESIGN, INC.
P.O.Box 3304
Cerritos, CA 90703-3304
Tel: 714-527-8540 **FAX:**714-527-1812

Reader Service #128

☆ ☆ ☆ ☆

"The *Best* 8051 *Emulator*"

NEW 8XC751 support available now!



5 ft. cable

8051

See EEM 88/89 Page D-1304

PC based emulators for the 8051 family

(8051/51FA/52/31/32/44/152/451/452/535/552/8XC751 + CMOS + more

- PC plug in boards or M-232 box
- Pull-down menus combined with Command-Driven User Interface
- Context sensitive help and On-Screen Editing of data
- 20 MHz real time emulation
- 128K emulation memory
- 48 bit wide, 16K deep trace buffer with loop counter
- Program Performance analyzer
- Powerful Macros with IF-ELSE. REPEAT-WHILE structures
- Source Level debug for PL/M-51 and C-51
- Symbolic debugging with in-line assembler and disassembler
- Execution time counter
- Trace can be viewed during emulation!

PRICES: 32K Emulator for 8031 $1790: 4K Trace $1495'

CALL OR WRITE FOR FREE DEMO DISK!

'US only **Ask about our *demo* VIDEO!**

nOHaU CORPORATION

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

"See us **at Wescon 89 — Booth 902"**

Reader Service #143

Intel's Dark Horse-The 80960

A Powerful New Controller for Performance-Critical Applications

by Tom Cantrell

Like many here in Silicon Valley, I did time in the hallowed cubicles of Intel. In fact, some of you oldsters may remember my "8088 Processor For The S-100 Bus" articles in BYTE.

The 80x86 family of processors has been good to Intel. Thereare those who would argue that thecompany should focus on new additions to this particular series of chips, and not waste time in other areas. It's to management's credit that Intel has developed some new chips (undoubtedly over the heated objections of the financial types in the company).

Let's take a look at one of these: the Intel 80960. I don't know how much attention it will get compared to the 80x86s and the new media darling 80860, but (perhaps a surprise for those of you who think I'm an incorrigible cynic) in my opinion the 80960 is a darn neat chip!

A FRESH START

The 80960 is a high-performance 32-bit CPU targeted at embedded controller applications. Gone are the segments; the chip features a linear 32-bit (4-gigabyte) address space. A block diagram of the '960 is **shown** in Figure 1.

Thankfully, the '960 doesn't have any pretensions to desktop work. The purposeful control orientation of the chip, in this era of "retro-marketed" UNIX chips, is quite refreshing.

Today, the '960 family consists of three members: the 80960KA, KB, and MC. The first two are commercial products which differ only in the presence (KB) or lack of (KA) on-chip floating point. The MC is a military version which features special architectural support for Ada as well as the typical military temp range and manufacturingflow. TheKA is quoted at $174 and the KB at $380 in 100-piece quantities-not bad for introductory prices, though limiting the chips to applications which really need performance at any price. Meanwhile, the MC goes for $2400!

There is also a CA version in the works which should be announced imminently. From what I can tell, it dispenses with the floating point (like the KA) but delivers even higher raw performance than the KA and KB thanks to internal architecture enhancements. Also, the current '960s include reserved "Special Function Registers," so perhaps we'll see some "special functions" on the CA.

Even if you can't justify a '960 in your latest control project, it's worth taking a look at to see those features that are deemed worthy by today's chip architects.

Right off the bat a look at Figure 2, which shows the '960 KB register set (the KA deletes the 80-bit floating-point registers), indicates this chip is definitely not off the old block. Gone are all those "special" registers (each with its own "unique" instructions) that have characterized previous Intel CPUs. Instead, the architecture has been given a good dose of laxative for regularity and consistency.

Make no mistake, the '960 isn't a "single chip" like the 8-bit **controllers you're** used to. Besides no on-chip memory, thechipdoesn'tintegrateanyI/O(timers,UARTs,DMACs, etc.) either. Instead, Intel assumes (correctly, I think) that for the targeted high-end applications you can easily add commodity outside memory and I/O ICs (Figure 3 shows a basic interface to outside I/O devices). The '960 silicon is purely dedicated to processing, which it does quickly indeed.

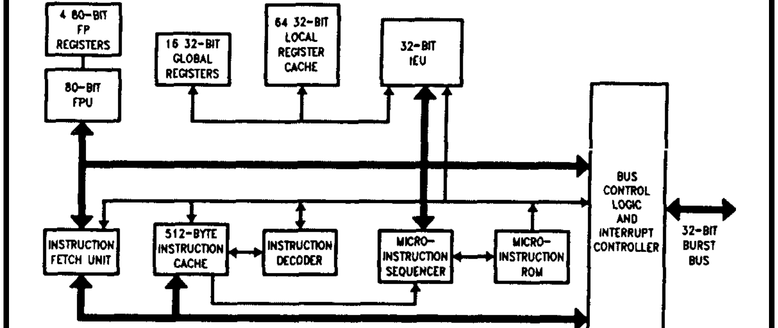
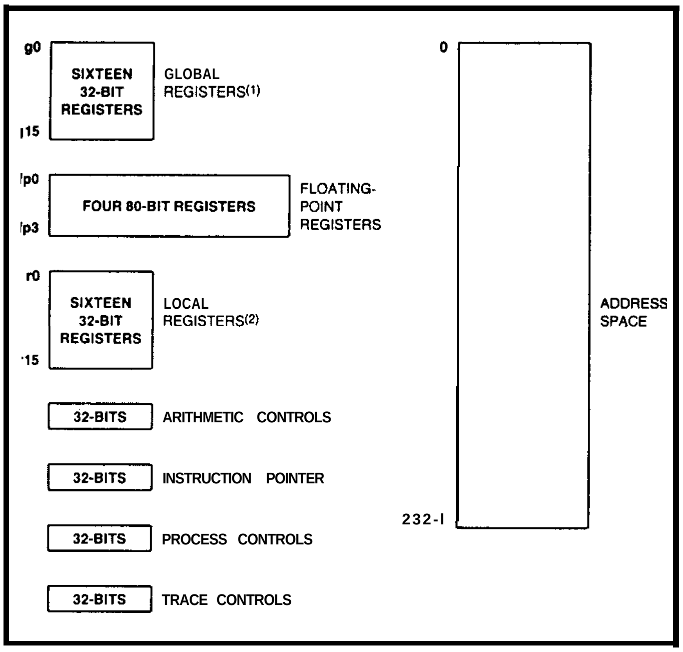

4 80-BIT FP REGISTERS
80-BIT FPU
16 32-BIT GLOBAL REGISTERS
64 32-BIT LOCAL REGISTER CACHE
32-BIT IEU
BUS CONTROL LOGIC AND INTERRUPT CONTROLLER
32-BIT BURST BUS
INSTRUCTION FETCH UNIT
512-BYTE INSTRUCTION CACHE
INSTRUCTION DECODER
MICRO-INSTRUCTION SEQUENCER
MICRO-INSTRUCTION ROM

Figure *1 -The 80960KB sports a highly parallel microarchitecture.*

WHAT'S HOT AND WHAT'S NOT

Much of the '960 promotional literature talksaboutitsRISCheritage. If you read my "RISC vs. Reality" opinion in the first issue of CIRCUIT CELLAR INK, you know where I stand on the "ISC" wars: instruction set complexity as a



Flgure 2—*The 80960* register set is *rich with general-purpose registers as well as floating-point registers.*

technical issue is vastly overblown (it is a big marketing issue though).

Indeed, the '960 does exhibit many of those features which characterize the RISCy approach including:

- Fixed-size(32-bit) instructions, many of which execute in few (1 or 2) clocks.
- Load-store architecture-instructions operate only on registers, not memory.
- Lots of registers including four local "banks" to minimize memory references across subroutine calls.
- A healthy instruction cache (512 bytes) to minimize the memory bottleneck which chronically plagues RISC.

However, in other ways the '960 deviates from RISC ideological purity (the reality part):

- More instructions and addressing modes than considered fashionable by true RISC zealots.
- Some of the complex instructions take many clocks to execute and are implemented with (horrors!) microcode.
- It is possible to write'960 assembly language; the speed features are largely in hardware and don't require an optimal optimizing compiler.

I find it interesting that the "R" word is used in some documents and carefully avoided in others. Instead, the instruction set is characterized as "simplified" (SISC?) and "versatile" (VISC?).

If fiddling with the instruction set isn't a big deal, what are the true performance features of the '960?

Parallelism -On-chip, multiple functions can proceed in parallel including regular instruction execution (integer unit), floating-point instruction execution (floating-point unit), bus access (bus interface unit), and branch processing (handled by the instruction decoder). Parallelism is the key to Intel's claim that the '960 can theoretically one-up RISC by executing more than one instruction per clock!

Scoreboarding-Doing multiple things at the same time can get confusing; it is important that the chip not get ahead of itself (in computerese, a "hazard"). To this end, the '960 implementsa big-computer feature called register "scoreboarding." The scoreboard keeps track of which registers have pending changes. For instance, when a register load instruction is encountered, the targeted register is marked in the scoreboard. Now, the next instruction execution can be started, even before the previous load completes, as long as this next instruction doesn't use the marked register. Of course, if the next instruction uses the marked register, everything must "stall" until the load completes. In general, the four functional units mentioned previously may proceed independently and completion order can differ from the order of initiation. Where RISC relies on the compiler to keep things straight, the '960 does it in hardware.

Register Sets and Cache-All "few-clock-per-instruction" (and more so "multiple-instruction-per-clock") machines suffer from the memory bottleneck. The common cry is "where can I get some cheap IO-ns DRAM s?" Until the memory folks deliver, the answer is to keep accesses on-chip. The'960 takes a two-fold tack. First, lots of on-chip registers increase the likelihood that important data can be kept at hand. Second, a 512-byte instruction cache is provided in the hope instructions can be found there, instead of in memory. Furthermore, the cache features parallel load/decode. When a miss occurs, the missed instruction is loaded into the cache and sent to the instruction decoder at the same time. Note that no data cache is provided, which is the simplest way to avoid the "cache coherency" problem associated with multiprocessor designs.

Bus *Bandwidth-A* truism in the automotive world is that there is "no substitute for cubic inches." Sure you can take a 2-liter engine and add turbos, valves, fuel injectors, and so on, but when the light turns green, give me a 454-c.i. V8 any day. Much the same goes for computers. Despite the registers and instruction cache (I remain to be convinced that the latter will be well behaved in a multi-interrupt-driven application), it is imperative that a fast CPU be able to feed its voracious instruction/data appetite. So a simple metric, like cubic inches, is bus bandwidth-namely how fast and how wide. The '960 specs well on both accounts with 16–20-MHz clocking of its 32-bit multiplexed address/data bus. Furthermore, it supports "burst" transfers (a natural match for modem DRAM column- and nibble-access modes) which boosts the rate for sequential accesses to 50+ megabytes per second at 20 MHz.

On-Chip Floating Point-If you **need floating point, the best place for it-as is** the trend with other advanced CPUs—is on-chip. The reason is that, especially as the CPU clock rate increases, the overhead of communication between the CPU and an off-chip FPU becomes a limiting factor rather than the execution of the floating-point operation itself.

Just how fast is the '960? The marketing brochure promises "100+ VAX MIPS" in the "1990s." Meanwhile, today's chip is spec'd at anywhere from 7.5 to 20 VAX MIPS depending on how contrived the benchmark assumptions are. Like the automotive world's published "horsepower" figures, take the MIPS numbers with a grain of salt. Forgetting the hype, based on what's under the hood it's safe to say the '960 is a screamer.

INTERRUPTS

A key requirement for a high-performance controller chip is the ability to field and quickly respond to a variety of interrupt sources. Thus, the '960 devotes significant resources to on-chip interrupt control logic.

The '960 offers four interrupt input lines which can be programmably specified to work in three different ways:

- As four separate interrupt inputs.
- As two separate interrupt inputs plus INTR (interrupt request) and INTA (interrupt ac-

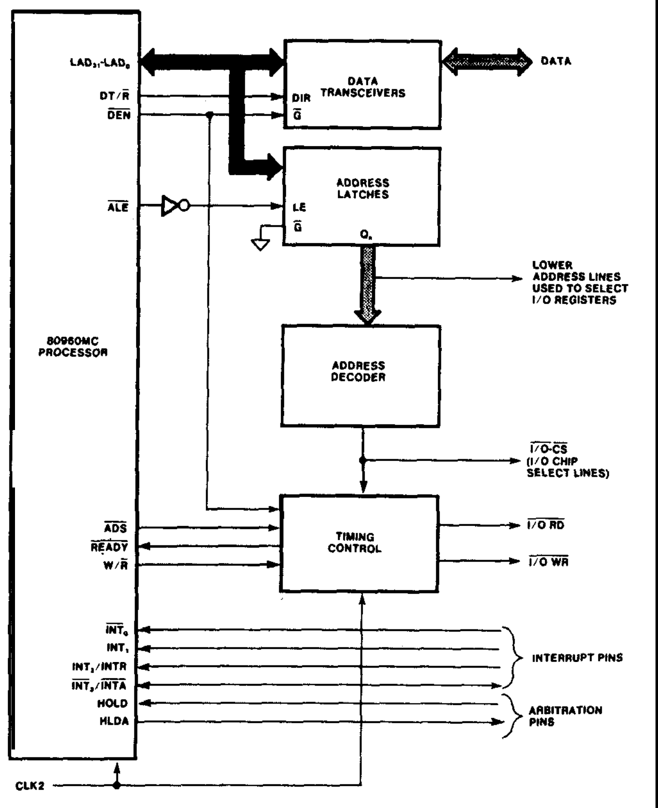
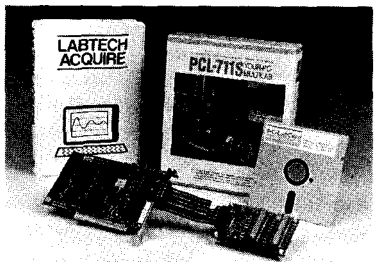


Figure 3—*Interfacing I/O* **devices** *to the 80960 is simplified by the* **use** *of a few external* circuits.

Never So Complete and Affordable...
a practical PC DA&C solution from
PC-LabCard
now only
$295



Model	PCL-711S	1-XL-812	PCL-718
Analog Input	8SE	16SE	16SE/8DI
Resolution	12 Bits	12 Bits	12 Bits
Max. Throughput	25KHz	30KHz	60KHz
DMA & Interrupt	No	Yes	Yes
Analog Output	1	2	2
Digital I/O	16/16	16/16	16/16
Counter/Timer	No	Yes	Yes
Utility Software	Yes	Yes	Yes
Screw Terminal	FREE	$55	$55
Free LT ACQUIRE	Yes	Yes	Yes
Price	$295	$375	$795

Distributors & OEM Welcome
CALL FOR FREE CATALOGS! (408) 293-6786

Advantech
American Advantech Corp.
1460 Tully Road, Suite 602, San Jose, CA 95122
TEL: (408) 293-6786 FAX: (408) 293-4697

Reader Service #103

R&D ELECTRONIC SURPLUS, INC.
Has been in business since 1946 selling NEW surplus **electronics and electromechanical parts.**

Send for our FREE 40 **page catalogue detailing:**

Batteries	LEDs
Cables	Lugs
capacitors	MOVs
Clocks	Niiads
Connectors	Power Devices
Digital Timer/Controllers	many Power Supplies
Diodes	Relays
Displays	SemiConductors galore
Enclosures	Slepper Motors & Driver ICs
Fans	Speakers
Fillers	Many switches
Heat-Shrinkable tubing	Telephones and components
Heatsinks	Transformers
Integrated Circuits	Ultrasonic Transducer Board
Lamps & Lights	
	& many more items

R&D ELECTRONICS

1224 Prospect Avenu
Playhouse Square
Cleveland, OH 44115

Telephone: (216) 621-1121 • Fax: (216) 621-8628

Reader Service #149

knowledge) lines for use with an external interrupt controller IC like the 8259A. This boosts the number of potential interrupt sources to 256.

- As three separate interrupt inputsand an IAC ("Interagent Communication") request. An IAC is similar to an interrupt, but a message is passed via memory and automatically handled by the CPU; an IAC "handler" isn't required. IACs can make the CPU stop, reinitialize, set a breakpoint, and so on.

A key factor for interrupt response is "latency'‘-the time from request to action. On a typical CPU, latency may become uncomfortably large. First, the CPU has to recognize that an interrupt signal is asserted. Next, it must complete the current instruction. Then, the "state" or "context" must be saved. Finally, the interrupt can be handled. The '960 speeds the process as much as possible by, for example, making sure slow (many-clock) instructions are "interruptable" and automatically saving the interrupted context.

DON'T BUG ME

Another modem trend highlighted on the '960 is the inclusion of some pretty powerful debugging facilities on-chip. After all, even the whizziest chip is useless if it spends most of its time (and your time) in never-never land.

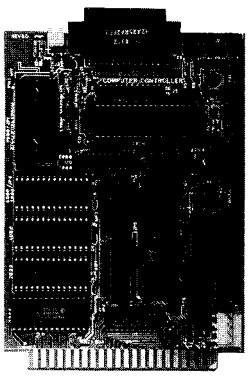
However, other than altruism, there is good reason to add debug features to the chip. See, the companies that make the chips also make the in-circuit emulators, and the emulator designers would be up a creek without some support on-chip. The main culprit are cache and associated stuff like prefetch algorithms.

In the old days, an emulator could just sit outside and watch (and/or modify) the bus traffic. But with cache and prefetching, activity on the bus says little about what the processor is actually doing. I guess in the the worst case, say a small loop which fits in the cache, the outside bus may be totally idle. Obviously, this puts a crimp in your bug-stomping style.

So, the '960 includes all kinds of debugging features, called trace controls, which, working in concert with monitors and/or emulators and/or logic analyzers, give you half a chance of seeing what's going on. The '960 supplements the traditional "instruction trace," which traps each instruction executed, with the ability to trap branches, calls, returns, prereturns (the instruction before a return), and supervisor entry/exit. It also includes breakpointing in the form of software breakpoint instructions and two address breakpoint registers.

The '960 even features automatic self-test at power-up. A *FAILURE* pin remains asserted if things don't check out. This can obviate the need for dedicated tester hardware/

BCC52 BASIC-52 COMPUTER/CONTROLLER



The BCC52 Computer/Controller is Micromint's hottest selling stand-alonesingle-board microcomputer. Itscost-effectivearchitectureneeds **only a power supply** and **terminal** to become a **complete development or end-use system, programmable in BASIC or machine language. The BCC52 uses Micromint's new 80C52-BASIC** CMOS microprocessor **which** contains a ROM-resident 8K byte floating-point **BASIC-**52 interpreter.

The **BCC52** contains **sockets** for up to 48K bytes of **RAM/EPROM**, an "intelligent" **2764/128 EPROM programmer,** 3 **parallel** ports, a **serial terminal port with auto** baud rate **selection,** a serial **printer** port, and it is bus compatible with the full line of BCC-bus expansion boards. The BCC52 bridges the gap between expensive programmable controllers and **hard-**to-justify prim-sensitive control applications.

BASIC-523 full **floating-p**oint BASIC is fast and efficient enough for the most complicated tasks, **while i**Its cost-effective design allows it to be considered for many new areas of implementation. It can be used both for development and end-use a**pplications.**

Since the **BASIC-52** is bus oriented, it supports the following Micromint **expan**sion boards in any of Micromint's card cages with optional power supplies:

BCC22 Smart terminal board	**BCC13 8-Channel 8-bit A/D converter**
BCC25 LCD display board	**BCC30 16-Channel 12-bit A/D converter**
BCC33 3-port I/O expansion board	**BCC18 Dual channel serial I/O** board
BCC40D 8-Channel optoisolated I/O expansion board	**BCC55 Prototyping board**
BCC40R 8-Channel relay output board	**BCC45 Stepper Motor board**
BCC53 Memory ad **6-port I/O exp. board**	

3CC52	BASIC -52 Controller bard	**$189.00**
3CC-SYST.5	'52 PAK' Starter System	**$449.00**

Includes:BCC52, ROM A&BUTIL., CC01, MB08, UPS10

BCC52 OEM 100 Quantity Price **$149.00**

BCC52C Lower power all-CMOS version **$199.00**

Note: The **BCC52 series** Is **available h Industrial Temperature Range, fully ested.** OEM **100 quantity price—$189.00.** Cal **for other** OEM **pricing.**

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

Micromint, Inc. -4 Park Street, Vernon, CT 06066

Reader Service #140

QUALITY PARTS • DISCOUNT PRICES • FAST SHIPPING

ALL ELECTRONICS CORP.

P.O. Box 567 • Van Nuys, CA • 91408 • 818-904-0524

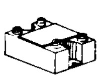
***SPECIAL* 22/44** PIN
EDGEBOARD CONNECTOR

.156" pin spacing .200" between double **rows.** Gold **contacts** • P.C. mounting • **Same** as AMP Z-530655-6
CAT# EBC-1G $1.00 each. 10 for **$8.00**
100 for **$60.00**

6 **Mhz or 10 Mhz** CRYSTALS
(minimun order 100 **pieces** of one value)
HU **18/U Style**
6 **Mhz CAT# CRY-618S**
10 Mhz **CAT# CRY-1018S**
100 **pieces** for $75.00 • 500 **pieces for $300.00 • 1000 pieces** for S500.W

WIDE BAND AMPLIFIER
NED# UPC1651G
1200 Mhz **@** 3db. Gain: 19 db **@** f-500 hz
5 **volt** operation. Small **package** 4 mm **dia x 2.5mm thich. CAT#** UPC-1651 2 **for** $1.00
10 **for $4.50 • 100 for $35.00**

SPECIAL **10** AMP
SOLID STATE RELAY



ELECTROL **# S2161**
Control: 5.5 **-10Vdc**
(will operate 3 - 32 Vdc)
Lord: 10 **Amps @** 240 **Vac**
2 **1/4" x** 1 **3/4" x 7/8" CAT# SSRLY-10B**
$19.60 ea • 10 for $85.00 • 100 for $500.00

10 AMP 200 PIV
FULL WAVE BRIDGE
Small size: 5/8" square **CAT# FWB-1020**
$1.00 each • 10 for $9.00 • 100 for $80.00

PIEZO WARNING DEVICE
Murata Erie# PKB8-4A0
High **pitched** audible alarm. **Operates on 3 - 20 Vdc @ 20 ma.** 1' **high** X **7/8" dia.** P.C. board **mount.**
CATX PBZ-64 $1.75 each
100 for **$160.00 500** for **$700.00**

OPTO-ISOLATOR
Sigma # 301T1-12B1. Signal applied **to** the **input is** coupled by **means d light to isolated** photo **conductive** cell. High **reliablility** switching. 12 **voltinput. CAT#** **OP-301**
$1.50 each . 10 **for** 513.00

ORDER TOLL FREE I-800-826-5432

FAX (616) 781-2653 **Terms: Phone orders must be charged to Visa MasterCard or Discover Minimum order $10.00 • Add $3.00 per order** la **shipping/handling • CA residents add sales tax • Quantities Limited • Items pictured may differ from actual product • Mail orders may be paid by check • NO C O.D.**

FOR A FREE 60 page CATALOG
Circle No. 101 on Reader Service Card

software (nontrivial for a chip this fast and complicated) and help solve problems in the field.

CHECK IT OUT

Normally the cost and complexity of a chip like the '960 would make casual experimentation prohibitive. After all, the chip itself costs hundreds of dollars and you've probably got better things to do than taking a wire-wrap gun to its 132-pin package (you won't find a socket at Radio Shack either).

However, all is not lost. To support the independent developers, Intel offers the "QT960" Design Kit which is a complete 80960KB-based single-board computer. The board includes a resident debug monitor and communicates via an 82510 (successor to the venerable 8251) UART with support for XMODEM downloading of host-developed code. For memory, the QT960 contains sockets which support a mix of EPROM and/or flash (5V-only) EPROM and SRAM. For fast transfers and timing operations, the board features an 82830 DMA/timer chip as well.

To support your prototyping efforts, the board offers seventeen square inches of prototyping area. Furthermore, schematics, monitor source code, and PLD equations are included. All in all, the QT960 Design Kit is just what the engineer needs to get a '960 design off to a flying start.

Best of all, it only costs $960 (interesting coincidence)! Intel also offers a version with more and faster memory chips for $1960, which should make their financial types a little happier.

To get hold of a QT960, contact your local Intel distributor. (To find the distributor closest to you, call 800/874-6835). For more information on the 80960, call 800/548-4725 or write:

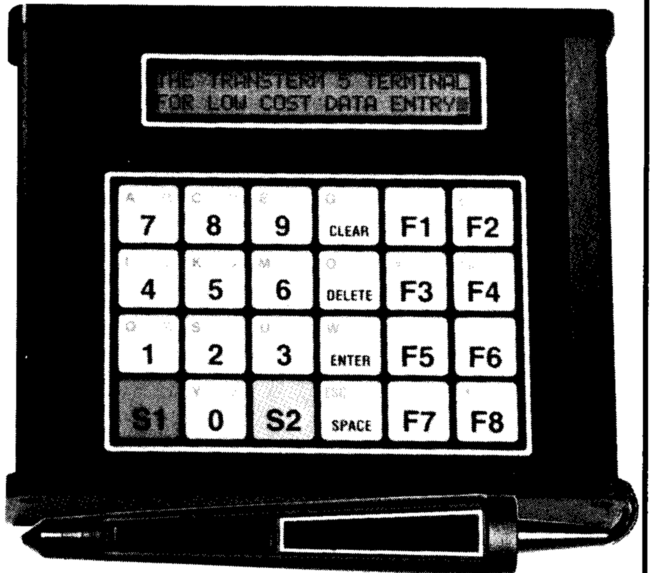
Intel Corp.
Mailstop GR1-58
P.O. Box 58065
Santa Clara, CA 95052-9979

Tom Cantrell holds a B.A. in economics and an M.B.A. from UCLA. He owns Microfuture, Inc., and has been in Silicon Valley for 10 years involved in chip, board, and system design and marketing.

IRS

219 Very Useful
220 Moderately Useful
221 Not Useful

$249.TERMINAL



Featuring
- Standard RS-232 Serial Asynchronous ASCII Communications
- 48 Character LCD Display (2 Lines of 24 each)
- 24 Key Membrane Keyboard with embossed graphics.
- Ten key numeric array plus 8 programmable function keys.
- Four-wire multidrop protocol mode.
- Keyboard selectable SET-UP features—baud rates, parity, etc.
- Size (5.625" W × 6.9" D × 1.75" H), Weight 1.25 lbs.
- 5 × 7 Dot Matrix font with underline cursor
- Displays 96 Character ASCII Set (upper and lower case)

Options—backlighting for display, RS-422 I/O, 20 Ma current loop I/O,

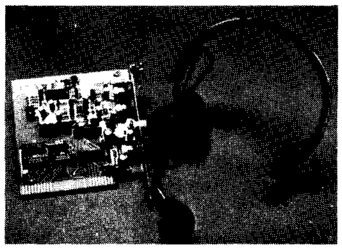
COMPUTERWISE, INC.

302 N. Winchester • Olathe, KS 66062 • 913-829-0600 • 800-255-3739

VOICE MASTER KEY®
VOICE RECOGNITION SYSTEM
FOR PC/COMPATIBLES & TANDY 1000 SERIES
A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING. . .The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response. **a complete, superior speech and sound development tool.**



SOFTWARE CONVERSION CODES. The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED. .**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY $149.95
COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS SOFTWARE ONLY—REQUIRES TANDY BRAND ELECTRET MICROPHONE.

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add $5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

COVOX INC.
675 Conger Street. Eugene. OR 97402
TEL: 503-342-1271 • FAX 503 342-1283

Reader Service #117

Reader Service #113

SOFTWARE by DESIGN

Algorithms for Trigonometric Functions

by Jack Ganssle

Most real programs are forced to use floating-point numbers to handle high-precision data over a wide range. Even in embedded systems floating point is often needed to convert raw data into some useful form. Ed Nisley discussed the difficult problem of converting from floating point to integers in Firmware Furnace in CIRCUIT CELLAR INK #10. A complementary problem is implementing transcendental functions in floating point.

C and other high-level languages usually include all of the standard math functions built-in to the library. However, assembly language programmers, already saddled with the tedium of generating detailed machine code, must also develop their own approximation algorithms. Sometimes C programmers face the same issue, particularly if they delete the math library to save space.

The very nature of trig functions implies that no exact solutions can exist in a digital computer. All trig routines use some sort of approximation to solve for an input value.

Hundreds of different algorithms exist. Each involves a tradeoff between precision and computation time. To further complicate matters, many algorithms are designed only for limited ranges of input data; calling the function with a value outside of the range will often give wildly inaccurate results.

How do you decide which algorithm to use? If you know the data will be confined to a small range, you can easily optimize the algorithm to cope with that data. Or, if speed is not an issue but precision is, then slow but precise routines can be employed. No one compromise is ideal for all applications. The algorithms I'll present represent a fair tradeoff for many applications. Note that some machines have hardware support for parts of some of these functions (e.g., the 8087); with careful planning you can partition the problem in such a fashion that the hardware handles what it can, and the software takes up the slack.

Most transcendental functions are based on some sort of series expansion. The obvious choice is the Taylor series, but in practice it converges to useful answers much too slowly. Many other series can be used, but the most common is the Chebyshev.

These algorithms are shown in a "pseudo language" so they can easily be implemented in any computer dialect. They'll work with any floating-point format.

You can simplify these algorithms quite a bit. For clarity's sake they're shown decomposed as a series of equations. In practice, often several steps can be reduced to one. Or, you can take advantage of the nature of floating-point numbers to reduce computation time.

For example, whenever a value must be raised to an even power (e.g., squared), increment the number's exponent digit. With most floating-point formats, adding one to the exponent digit doubles the value; it is equivalent to shifting the value left one bit. Similarly, don't compute the same value repeatedly; do it once and use it as required. Finally, *never* compute a constant; if you need n/2, precompute the constant's value and build it into the code as a fixed constant.

THE TANGENT

The tangent is almost always computed using the relation *tan(a)=sin(a)/cos(a). The sin* and cos functions are described later.

```
;*If the input argument is X and the result A:

STEP1: IF X<0 THEN BEGIN
              X=absolute value of X
              CALL  STEP2
              CALL  STEP3
              CALL  STEP4
              RETURN-A
                     (since ATAN(X)=-ATAN(-X))

STEP2: IF X>1 THEN BEGIN
              X-l/X
              CALL  STEP3
              CALL  STEP4
              RETURN -A+pi/2
                     (since ATAN(X)=π/2-ATAN(1/

STEP3: IF X>TAN(pi/12) THEN BEGIN
              X=(X * SQRT(3) - 1)/
                           (SQRT(3) + X)
              CALL  STEP4
              A-A + pi/6

STEP4: T1=  0.99999999971301
       T2=  -0.3333331319373
       T3= 0.199977320113
       T4=  -0.14195746243
       T5= 0.963034789
       A = X * (T1 + T2*X**2 +
               T3*X**4 + T4*X**6 + T5*X**8)
       RETURN
```

Figure 1 *-An arctangent routine.*

The arctangent (inverse tangent) is much more difficult. Worse, it is used as part of the approximation process for many other functions. Since any approximation can be no more precise than its weakest link, it is essential that the arctangent (or, *atan*) is as accurate as possible.

Figure 1 shows the pseudo-language description of an *atan* routine. This routine gives answers with about 9.5 decimal digits of precision. A considerable amount of exponentiation is involved-be sure to avoid it as described earlier!

The *atan* function returns an argument in radians.

SINE AND COSINE

The sine and cosine functions can be calculated as shown in Figure 2. This results in about 6 decimal digits of accuracy.

These approximation are valid only for the range of 0 to 45 degrees (the argument is in degrees). For the range 45 to 90 degrees, remember that *sin(x)=cos(90-x).*

Beyond the 90-degree mark, you must segment the function into quadrants. Both sine and cosine are periodic, but their signs change as a function of the quadrant.

The inverse functions (*arcsine* and *arccosine*) are computed using the algorithms shown in Figure 3. If no error is introduced in the calculation, these routines are exact. Of course, the square root and arctangent routines will be the main source of error. The results are in radians.

```
; The input argument is X (degrees) and the
;   output is A
; X ranges from 0 to 45 degrees


X=X/100

        SINE(X)=1.7453293*X* (1-0.50758*X*X*
                (1-0.149*X*X))

        COSINE(X)=1-1.523087*X*X*
                  (1-0.25382*X*X*(1-0.1*X*X))
```

Figure 2—*Sine and cosine routines.*

HOW CLOSE IS CLOSE?

Typical errors are included with the algorithm descriptions, but these predictions assume that no errors will be introduced in the algorithms' implementation.

Probably the largest source of errors is the precision of your floating-point library. Thirty-two-bit IEEE format (that used in the IBM PC) carriesabout six to **seven** decimal digits of precision—quite a bit less than these algorithms are designed for. Indeed, the coefficients of the equations cannot **even be accurately represented** using as many as 32 bits.

Each additional math operation (addition, multiplication, etc.) used during the computation of the algorithms will introduce error.

```
; X is the input argument and A is the result.

ARCCOSINE ROUTINE:

        IF X= 0 THEN RETURN A= pi/2
        IF X= 1 THEN RETURN A= 0
        IF X=-l THEN RETURN A- pi
        IF 0 < X < 1 THEN RETURN A= ATAN(SQRT
               (1-X**2)/X)
        IF -1< X < 0 THEN RETURN A= ATAN
               (SQRT(1-X**2)/X) + pi

ARCSINE ROUTINE

        IFX- 0 THEN RETURN A= 0
        IF X= 1 THEN RETURN A= pi/2
        IF X=-l THEN RETURN A= -pi/2
        IF 0 < ABS(X) < 1 THEN RETURN A=
               ATAN(X/SQRT(1-X**2))

(ABS is the absolute value function)
```

Figure 3—*Arcsine and arccosine routines*

The 8087carries an 80-bit floating-point number, good for some 18 decimal digits of precision. Although more accurate than these algorithms, you must be careful to not lose precision by careless rounding or by thoughtlessly discarding resolution (most double-precision packages only carry 64 bits of the possible 80). The reference listed at the end of this article describes algorithms with 20 or more digits of precision. Of course, these techniques are measurably slower!

It makes good sense to prototype the algorithms to measure their accuracy, particularly if your data is limited to some small range. However, don't compare the results to that generated by your compiler's internal trig functions! They are also based on some unknown approximation, and may be considerably less precise than you'd expect. Use a good set of tables or a known high-precision software package. Be wary of double-precision software packages-some give the same level of accuracy as the single-precision routines.

FINALLY...

If you need a particularly fast or precise approach to solving these transcendental functions (or others), refer to Computer Approximations by J.F. Hart (John Wiley & Sons, 1968). This reference is the Baedeker of math algorithms; it examines most common math functions in excruciating detail. The book does assume some knowledge of higher math. ✣

Jack Ganssle is president of Softaid, a vendor of microprocessor development tools. When not busypushing electrons around, he sails upand down the East Coast on his 35-foot sloop.

IRS

222 Very Useful
223 Moderately Useful
224 Not Useful

CONNECTIME *Excerpts* from *the Circuit Cellar BBS*

THE CIRCUIT CELLAR BBS
300/1200/2400 bps
24 hours/7 days a week
(203) 87 1-1988 — 4 incoming lines
Vernon, Connecticut

Conducted **by Ken Davidson**

The message base of the Circuit Cellar BBS in now available on disk. See page 78 for details.

In this installment of ConnecTime, we'll deal with keypad, telephone line, and motor control interfacing. But first, proper power supply decoupling is crucial in any TTL circuit for reliable operation. Placement of bypass capacitors can be tricky at best, especially on multilayer boards, as we find out in the first discussion.

Msg#:22565
From: BOB PADDOCK To: ALL USERS

I hope you can give me a definitive answer to a question about the proper way to connect decoupling capacitors on a multilayer PC board.

Let's say my circuit is a 74AC273 and 0.33µF decoupling cap on a four-layer PCB (two layers for signals, one layer for ground, one layer for Vcc).

Each output of the 273 can sink or source 24 mA, so if all eight outputs change state while fully loaded, we have a current of about 200 mA for ground and Vcc. (For this discussion, we'll assume the part isn't going to melt from exceeding its package dissipation rating.)

There are several ways that the 273 and 0.33µF cap can be connected to Vcc and ground:

1) Connect the pins of the 273 to the interlayers (GND & Vcc) at the pins of the 273. Connect the cap to the interlayers. There are no traces between the 273 pins and the cap, just the interlayer connection.

According to Intel appnote AP-125 "Designing Microcontroller Systems for Electrically Noisy Environments," the 273 currents will take the shortest path to the highest nearby voltage source (the cap, hopefully). This way takes up the least PCB area (you don't need traces between the 273 and the cap).

2) Connect the pins of the 273 to the interlayers at the pins of the 273. Run traces from the cap to the 273 pins. We now have inductance between the cap and the 273.

3) Connect the cap to the interlayers. Run leads from the 273 to the cap. Inductance between the cap and the 273.

4) Connect both parts to the interlayers and run traces between the parts. Seems redundant, and you have parallel inductance.

RCA appnote "Printed Circuit Board Design Using AC/ACT Logic Devices" has this to say:

"DO NOT decouple the Vcc and ground plane by tying Cd [decoupling cap] between planes near an IC.

"DO connect Cd leads right at the IC Vcc and ground terminals. Preferred placement of the decoupling cap is diagonal to the Vcc and ground pin-under the IC or on the bottom of the PCB. [Easy to say, tough to do.]

"For surface mount, AVOID extra inductance of through-board connections to Cd. It should be remembered that an ampere or more of high-frequency transient current courses between Vcc/GND pins and the Cd body. [We're not using surface mount parts on this project, but we still have the current transients.]"

They don't say at what point to connect to the interlayers (at the cap or at the 273). Probably should do as case 2 above.

It seems that Intel and RCA disagree on how to hook up a decoupling cap. I feel that what RCA says makes the most senses, but on a very tight PCB using case 1 above makes layout so much easier.

So what do you think is the right thing to do?

Msg#:22614
From: ED NISLEY To: BOB PADDOCK

Well, the point of having separate power and ground planes is to reduce the inductance between the power supply point and the ICs. By increasing the copper area you reduce the inductance, and by having two parallel planes you increase the distributed capacitance. Decoupling capacitors should be relatively small (100 nF or so), have good high-frequency characteristics, and be distributed relatively evenly over the board.

If you have a good power and ground plane setup, the caps can be located physically near the ICs, but connected directly to the planes. The transient currents will come from the distributed capacitance through the planes, which will have much lower inductance than traces connecting the ICs and caps. Because the caps are associated with the planes rather than the ICs, you don't need redundant traces connecting them.

Putting a ferrite bead at the point where the power supply enters the board, with a 10-µF tantalum cap just inboard of that, reduces the hash crossing the board connector through the power supply. The remaining caps on the board are all relatively small decoupling caps spread evenly over the area: The rule of thumb is one cap per IC; although you can reduce that somewhat for SSI gates, you should not scrimp around bus drivers!

I'm not sure what difference surface mounting makes; the inductance of the vias is probably less than the inductance of the PC traces or the leads on standard caps. In any event, more smaller caps will give better results than fewer bigger ones simply because they'll improve the distributed capacitance.

Measuring the actual power supply noise is an art unto itself; **you** need good equipment and impeccable techniques. A friend of mine found this out the hard way when he was measuring switching power supply noise: he had to build a special fixture to get the scope probe and ground connections to the right points without the inductance of ordinary leads!

Real-time controllers often require some form of operator input, and a keypad is a popular choice for the user interface. The complexity of the circuit used to interface the keypad depends heavily on what else the controller is doing, as the next conversation shows.

Msg#:21846
From: ROBERT BARBAGALLO To: JEFF BACHIOCHI

I am in the process of building an analog multiplexer for a lighting control system using the 8051. I have written and tested both input and output routines and they work great. Where I'm having trouble is with the keyboard routine. I would like to use 4x 4matrix keyboard to cut costs, but I am unwilling to debounce in software, so I chose the MC14419 to take care of the problem. The soft problem is that the multiplexer can accept an input from 1 to 192 (decimal) which reflects dimmer output number, and from 1 to 96 (decimal) which reflects input channel number. There is also a clear key and an enter key. When I read the keyboard port, it is in BCD and I can't figure out how to separate all this. I've never done any keyboard routines; could you advise me on a good book or some sample listings that may help me.

Msg#:21876
From: JEFF BACHIOCHI To: ROBERT BARBAGALLO

If you haven't already seen Ed's Firmware Furnace from the January/February 1988 (#1) issue of Circuit Cellar INK, shame on you. There, Ed explains the task of using minimal hardware (74LS240) for keypad-type input. It is a polled system with no hardware debounce. This design was used in the Infrared Master Controller ("Ciarcia's Circuit Cellar," BYTE, February 1987).

If you want to use the Motorola device, it is simple to use. A four-by-four cross-matrix l-of-16 input gives BCD output and strobe (needs a clock-80 kHz max.). An interrupt upon strobe routine could read the outputs which, if the high four bits were tied low (D4–D7) on, say, port 1 of your 8051, would read decimal O-9. Note that the keys A-Fdo not produce a strobe. They do produce an output, but contact bounce can exist on the output which would normally be cleaned by the strobe. Since you are in control of what "legal" input looks like, set up some rules.

For example, dictate that input will consist of three numeric digits (0–9, 0–9, 0–9) followed by a mode digit (A-F). Each digit entry will cause a strobe. Take the first digit times 100, add the second digit times 10, and add the third. Now look for a mode digit (A-E) by polling. Determine the function and whether or not the value is legal.

If you must, you can use only decimal digits; the first or last digit could indicate the mode (level, channel, clear).

Other alternatives are available. Check out the 74C922 (INS8242N), HD100165, or cascading two 74LS148s fall are available from Jameco and others). The MC14490 (six debouncers) could be used to debounce a 4 x 4 matrix on port 1 which is split into four output bits and four input bits (similar to Ed's circuit). Whatever you choose, it will depend on the rest of your program. If the controller is not doing much more than responding to keyboard input, you can get away with less expensive hardware. Otherwise you may need to forego polling and use a strobed interrupt routine.

Projects which need to connect to the telephone network are always popular, but special precautions must be taken before they can be legally connected to the nearest phone jack. The next message thread deals with whether a DAA (Data Access Arrangement) is the best way to handle those precautions.

Msg#:21313
From: EDWARD ARCHIBALD To: ALL USERS

I'm building a project which is essentially an 8031-based phone answering machine. I have completed all of the basic hardware and firmware and am currently using a device that is manufactured by Cermetek (CH1813) which provides a DAA for my device. It also provides ring detect, off-hook detect, audio coupling to the phone line, and a means for forcing the line off-hook and on-hook using a TTL-level signal. However, this thing is expensive and there are number of other things that I would like to do. I don't really have the experience to know how to proceed. Any tips on these things would be appreciated. Also, any tips that indicate the reasons why any of these things could not be done, or would be expensive to do, would be very useful.

Are there any books available whose subject is exclusively telephone electronics and includes sample circuits?

1) First off, I would like to duplicate the functions that the Cermetek device provides at a lower cost (< $26.00)

2) Let's say that I have a telephone hooked up to my phone answering device that has a ringer that cannot be turned off. How can I inhibit ringing on this phone while still being able to detect things like the phone going off-hook? I would like to be able to

control this with the microprocessor so ideally a good answer would include a TTL-level interface.

3) I would like to be able to use the DTMF keypad of a phone that is connected to my device without causing the phone switching equipment to take notice. Is this possible? Maybe there is a sequence of tones that can be dialed to disable further interpretation?

4) How can I detect that a calling party has hung up before the telco finally disables the connection? How about when the telco finally drops the connection?

5) Let‘s say that I have two phone-type devices hooked to my device. I would like to be able to discriminate which of the two devices has gone off-hook.

Msg#:21321
From: NATHAN ENGLE To: EDWARD ARCHIBALD

I'm sure that there are some books about the telephone network, but since I work at AT&T I usually find it easier to just pick somebody's brain. The parts required for a phone line interface are cheap: just a transformer, a switch-hook relay, some zener diodes, and various resistors and caps.

The big cost in building it yourself is that you'll have to register your device with the FCC (as per Part 68 of the FCC regs). There are a LOT of restrictions on what sorts of things you're allowed to do: maximum amounts of current you're allowed to pull, maximum signal levels you're allowed to produce, minimum on-hook impedance, and so on.

To tell if someone on your circuit is off-hook you can use a sort of comparator setup. Line voltage difference is about 50V between tip and ring when you're on-hook, but when someone goes off-hook the line voltage drops down to about 6V.

You can hang additional phones in parallel on a phone line so that more than one person can listen and speak, but there's a definite limit tothenumberthatyoucandobecausetheextrasetswilltend to distort the signal a little bit.

Individually your questions don't ask for much, but taken as a whole I think it could get complicated, especially the FCC stuff. They're in the process of "simplifying" the FCC regulations again and nobody really knows 100% of how everything is going to settle down. There's going to be a five-year grace period during which everyone is supposed to learn the new regulations (including the FCC people who will have to enforce them); I know that our department is intentionally waiting to let the dust settle before we try to get new approvals on any of our products.

Msg#:21445
From: EDWARD ARCHIBALD To: NATHAN ENGLE

Thanks for the info. Now is the time that I wish I knew someone whose brain I could pick for the answers to these questions. By the way, do you know where I can find the details of the Part 68 regs? This is beginning to sound like it is significantly out of my league, but I could probably learn a whole lot by trying.

Msg#:21328
From: KEN DAVIDSON To: EDWARD ARCHIBALD

A $26 DAA is already cheap. Just a few years ago you'd have been hard pressed to find any for under $100. As Nathan said, you could probably build one for less as far as parts go, but do you really want to spend a few thousand dollars to get it certified so it is legal?

Radio Shack sells (sold?) a book called "Understanding Telephone Electronics" that provides a good introduction to most telephone basics. There aren't specific examples in terms of circuits, though, and not many of the questions you pose are answered.

Msg#:21446
From: EDWARD ARCHIBALD To: KEN DAVIDSON

$26.00 is cheap for a DAA? That's good information. I wonder what the folks who build answering machines that retail for $50.00 do? I don't think my wife is going to like this, but our answering machine is about to become a volunteer for the sake of science.

Msg#:21452
From: KEN DAVIDSON To: EDWARD ARCHIBALD

The guys who make cheap phones design the circuit and have it tested once at a cost of a few thousand dollars. Then they make several hundred thousand units, and thecost amortized over the entire production run becomes a few cents per unit. DAAs are produced in much lower quantities and have more features than a typical CO interface in a phone will have, so they are more expensive.

Msg#:21457
From: NATHAN ENGLE To: EDWARD ARCHIBALD

I can get an address for you, but I really can't recommend diving into Part 68 as an intro to telephony. It's a little dry.

Ken's right about DAA modules at $26 per; they're not really meant for something you plan to make a million of. If you've really got something that big up your sleeve then it would be well worth your while to design and register yourself. However, for projects of the sort that 99% of experimenters do, $26 is a very reasonable price.

One alternative to chopping up an answering machine would be to use one of those cheapo import phones. They have similar electronics as far as the phone line is concerned, they cost a lot less than answering machines, and cutting them up is pure pleasure (I hate those phones). :-)

Msg#:21593
From: DAVID LAWSON To: EDWARD ARCHIBALD

If you need, I've done several phone systems and did telephone interfaces from very cheap ($7) to very expensive ($100). I've also been present at the Part 68 testing and know what they were looking for two years ago, so I might be of some help.

I haven't found a good way to monitor the line to see if someone picks up a phone locally. On detecting **when** someone hangs up remotely (the called party) most COs (I think; at least mine does) will reverse the line polarity momentarily when the calling party hangs up. The way I sense for remote hangup is my CO will reassert dial tone 10 seconds after the remote party hangs up. I also have a fail safe that will disconnect me after 60 seconds. My unit won't stay on-line long because it doesn't have a lot to say; it just calls a paging transmitter.

Msg#:21769
From: EDWARD ARCHIBALD To: DAVID LAWSON

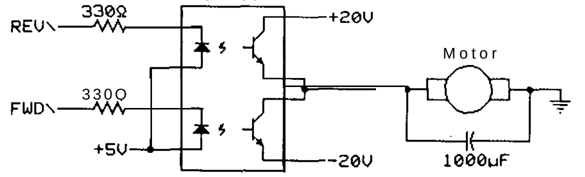
Thanks for the reply Dave. Since I posted, I've run across a few schematics for phone interfaces, and they look pretty complicated to my novice eyes. The main characteristic in all of them is the use of optoisolators. Anyway, thanks again.

Our last interfacing discussion deals with the best method for controlling motors used for track switching in model railroads.

Msg#:22159
From: WILSON SNYDER To: STEVE CIARCIA

I am looking for a suggestion on driving switch-machine motors for a college model railroad. These 30-mA motors select which of two tracks are routed to a third. The input of this circuit needs to be two digital lines: one to drive the motor forward and one to drive reverse (active low). Protection is not needed to prevent both from being low at once, with the exception of a couple of nanoseconds when the signals are switching. The other input is a 18VAC wire to power the motor (I usediodes to derive plus and minus voltages). Also, 5V is available. The output is a single wire: a positive voltage to turn the motor one way, a negative voltage for the other, and open to throw neither way. This output drives the motor (the other side of it is grounded), and a nonpolarized 1000-µF cap across the motor.

I tried an opto-Darlington, the NEC 2502-4, but it dies most likely due to the capacitor in-rush current (resistor limit it?). The circuit I tried is:



A nice design might use one of the Signetics power drivers, but one of the big problems I have had is getting a design that is cheap enough. I am going to need about a thousand of thesecontrollers, and anything much more than $1.50 (per motor) for the components would break the bank. Finally, I don't mind ganging up these controllers to provide, say, eight controllers on one board. This may allow the use of 8-wide parts such as drivers, resistor packs, etc.

Msg#:22254
From: STEVE CIARCIA To: WILSON SNYDER

Optoisolators usually aren't good for powering loads directly. Take your isolator outputs and drive a couple pass transistors and it will probably work.

A neater solution is to use a power op-amp like the Sprague UNL3751 Z. Such anop-amp has a 3.5-amp output current at 28V. Simply apply the drive voltages (±14V unless we add external components to allow ±20V) to the op-amp and run it like a logic comparator or amplifier. Negative voltage in, +14V out; positive in, –14V out; or OV in, OV out. These chips cost about $1.50 in hundreds. Contact a Sprague distributor and request samples.

The Circuit Cellar BBS runs on a 10-MHz Micromint OEM-286 IBM PC/AT-compatible computer using the multiline version of The Bread Board System (TBBS 2.1M) and currently has four modems connected. We invite you to call and exchange ideas with other Circuit Cellar readers. It is available 24 hours a day and can be reachedat (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, and either 300, 1200, or 2400 bps.

IRS

231 Very Useful
232 Moderately Useful
233 Not Useful

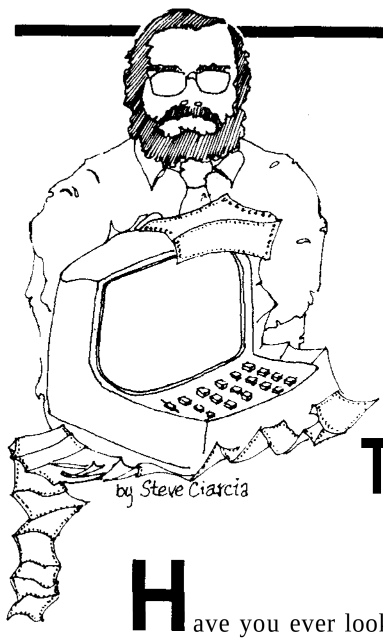
SOFTWARE and BBS AVAILABLE on DISK

Software on Disk
Software for the articles in this issue of Circuit Cellar INK may be downloaded free of charge from the Circuit Cellar BBS. For those unable to download files, they are also available on one 360K, 5.25" IBM PC-format disk for only $12.

Circuit Cellar BBS on Disk
Every month, hundreds of information-filled messages are posted on the Circuit Cellar BBS by people from all walks of life. For those who can't log on as often as they'd like, the text of the public message areas is available on disk in two-month installments. Each installment comes on three 360K, 5.25" IBM PC-format disks and costs just $15. The installment for this issue of INK (October/November 1989) includes all public messages posted during July and August, 1989.

To order either Software on Disk or Circuit Cellar BBS on Disk, send check or money order to:
Circuit Cellar INK — Software (or BBS) on Disk
P.O. Box 772, Vernon, CT 06066

or use your MasterCard or Visa and call (203) 875-2199. Be sure to specify the issue number of each disk you order.



STEVE'S OWN INK

Those Dazzling 32-Bit Chips

Have you ever looked into a car mirror and seen the sun? If you're like most people, you'll snatch your gaze elsewhere, but not before being dazzled. At first, the intense white light overwhelms anything else in your vision. For long minutes after, everything you see has the image of the sun burned into it. Even if you close your eyes, the amazing brilliance is projected on the back of your eyelids.

When I listen to engineers talk about the glories of 32-bit processors and controllers, I think about being dazzled. It seems like everyone who looks at these fast new computational wonders sees them wherever they look for months afterward. Even worse, some of them seem to have been temporarily blinded to common sense by the brilliance of what they've beheld. I'll be the last person you hear saying that we need to return to the "basics" of the 4004, but I will say that shoe-horning a fire-breathing, muscle-bound super-processor into every project that comes along is a short-cut to sloppy engineering.

The trouble is not so much the 32-bit processors and controllers themselves, but with the attitude of many of the engineers who turn to them on a routine basis. When you're working with machine intelligence that makes the "hard stuff" easy, it's tempting to turn to the same solution over and over again. Most of the new-generation 32-bit processors will let you get away with it because they just do so much. Of course, in most of the applications that come along most of the capabilities of the processor will be wasted, but this is dazzling technology. The picture of the hot new processor shows up on every new project that comes along. If you believe enough of the press releases you read concerning the processor, before long you've started substituting sheer power for thoughtful, professional design work.

I guess, when you get right down to it, this is my biggest problem with the biggest microprocessors and microcontrollers. I've always felt that the *design* is more important than the *hardware.* Getting inside a problem, understanding all the alternatives, and then deciding on the best approach for a solution get me excited. By the time theactual hardwarerollsout, it'salmost (and only*almost*)an anticlimax. I know that there are situations which demand narrow constraints on your final design. I just see a huge difference between working with limits placed by clients, supervisors, or existing conditions, and limits established by an engineer's personal set of blinders. One set adds to the challenge (and sometimes, the fun) of engineering; the other will lead inevitably to inflexibility, mediocrity and, worst of all, boredom.

Thirty-two-bit microcontrollers have their place. If I were designing the control system for a commercial nuclear reactor or an advanced avionics system, I would probably go with 32 bits as the minimum acceptable system. If, on the other hand, I'm working on distributed building control or industrial data acquisition, my experience tells me that 8 bits is the place to start looking for reliable, cost-effective solutions. I really don't care how powerful the new generation processors and controllers are: The one between my ears blows them all away, and it's the one I'll rely on to provide the best solutions to application problems.

Steve

Steve Ciarcia