

CIRCUIT CELLAR[®] INK[®]

THE COMPUTER APPLICATIONS JOURNAL

#67 FEBRUARY 1996

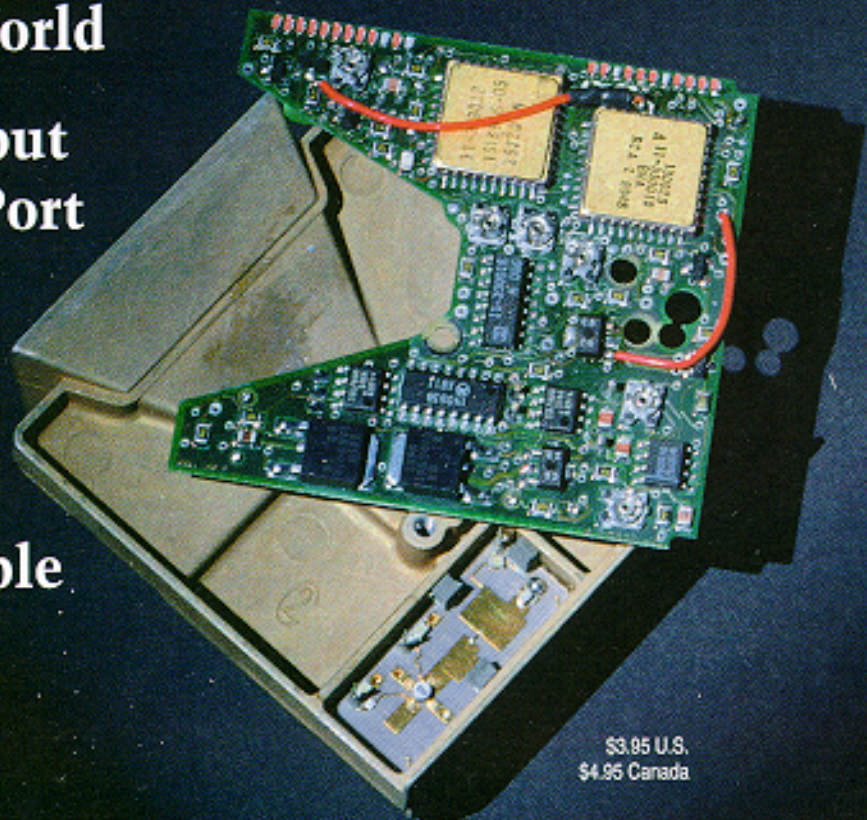
EMBEDDED APPLICATIONS

Connect the Personal PBX to the Outside World

12-bit Analog Input on Your Printer Port

Introducing the Zilog Z380

EPC: 1 Gbps Using UTP-5 Cable

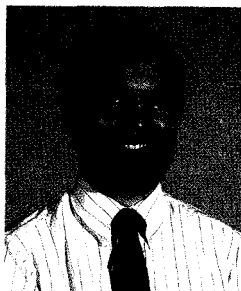


\$3.95 U.S.
\$4.95 Canada



TASK MANAGER

The Embedded INK



Without a doubt, if you're looking for an issue of *Circuit Cellar INK* dealing with embedded applications, this is the one you want.

We kick off our features this month by visiting an old friend that's been supercharged for the '90s. While the **Z80** isn't a microcontroller, it's been used in embedded applications for years. The 2180 was introduced over 10 years ago and added a host of onchip peripherals, but remained an 8-bit chip. The new 2380 brings the **Z80** core into the 16-/32-bit realm with a wider address and data bus, new instructions, and higher speeds while retaining backward compatibility at the binary level.

Next, we revisit the Personal PBX, a project first introduced about a year ago. In this installment, Richard Newman connects the PBX to the phone company and improves on some of the core circuitry. Keep an eye out for Caller ID in a future issue.

And, on the desktop scene-have you ever wanted to add a simple analog input to your PC or laptop, but didn't want to spend hundreds of dollars or didn't have an expansion slot available? **LPT:Analog!** is a neat little adapter that plugs into the parallel printer port of any PC compatible and gives you a 12-bit A/D converter. Just the thing for portable applications.

In our final feature, we follow up on last month's overview of low-to midrange PICs with a look at Microchip's high end: the **PIC17C44**. Don't throw away all that PIC code you've already written when your application demands more than the smaller PICs can offer. Simply move on up to this powerhouse processor.

Our first Embedded PC feature describes a new networking technology now available based on ATM that achieves data rates as high as 1 Gbps over standard Category 5 unshielded twisted-pair cable. The inventor himself gives the inside scoop on how this new technology can be applied.

In our other **EPC** feature, Ed Nisley, our own guru on embedding PC code, reveals some of the reference books he's found invaluable over the years. If you've wondered just how he does it, here are some of his resources.

In the **PC/104** Quarter, we size up some **PC/104** packaging options. There are off-the-shelf solutions for even the most unusual project.

Finally, Russ Reiss surveys what's available for small displays in embedded PC systems. There's something for everyone.

In our regular columns, Ed finishes up the Vid-Link project by describing just how he squeezed all the code into an 8-KB EPROM, Jeff converts Intel hex files into BASIC DATA statements, and Tom revisits fuzzy logic by looking at some new hardware offerings.

editor@circellar.com

CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

PUBLISHER
Daniel Rodrigues

EDITOR-IN-CHIEF
Ken Davidson

PUBLISHER'S ASSISTANT
Sue Hodge

EPC MANAGING EDITOR
Janice Marinelli

CIRCULATION MANAGER
Rose Mansella

TECHNICAL EDITOR
Carole Boster

CIRCULATION ASSISTANT
Barbara Maleski

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

CIRCULATION CONSULTANT
Gregory Spitzfaden

WEST COAST EDITOR
Tom Cantrell

BUSINESS MANAGER
Jeannette Walters

CONTRIBUTING EDITORS
Rick Lehrbaum
Russ Reiss

ADVERTISING COORDINATOR
Dan Gorsky

NEW PRODUCTS EDITOR
Harv Weiner

CIRCUIT CELLAR INKS THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S. and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. All subscription orders payable in U.S. funds only, via International postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to Circuit Cellar INK Subscriptions, P.O. Box 698, Holmes, PA 19043.9613 or call (800) 269.6301. POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 698, Holmes, PA 19043.9613.

ART DIRECTOR
Lisa Ferty

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover photography by Barbara Swenson
PRINTED IN THE UNITED STATES

For information on authorized reprints of articles, contact Jeannette Walters (660) 6752199.

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC
Barbara Best
908) 741.7744
Fax: (908) 741-6823

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

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

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

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 6 bits, no parity, 1 stop bit, (860) 871-1988; 2400/9600 bps Cower HST, (860) 871-0549 World Wide Web: <http://www.circellar.com/>

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

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

Entire contents copyright © 1996 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar *INK* is a registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc is prohibited.

- 1 2** Updating a Classic: the 280380 Microprocessor
by Monte Dalrymple
- 1 8** Connect the Personal PBX to the Real World
by Richard Newman
- 2 6** LPT:Analog!
A 12-bit A/D Converter Printer Port Adapter
by David Prutchi
- 3 4** PIC17C44 High-End Microcontroller Architectural Overview
by Ron Cates
- 7 2** Firmware Furnace
Vid-Link Characters
Part 2: Bits to Dots
Ed Nisley
- 7 8** From the Bench
Intel Hex to BASIC Data Statement Translator
Jeff Bachiochi
- 8 6** Silicon Update
Fuzzy Buster
Tom Can trell

EMBEDDED PC

See **pages 41-71** for Our Special **Bonus Section**

INSIDE ISSUE 67

- 2** Task Manager
Ken Davidson
The Embedded INK
- 6** Reader I/O
Letters to the Editor
- 8** New Product News
edited by Harv Weiner

- 91** ConnectTime
Excerpts from
the Circuit Cellar BBS
conducted by
Ken Davidson
- 104** Priority Interrupt
It Just Frosts My Chops
- 81** Advertiser's Index

READER I/O

LEARNING HOW TO CRUISE?

The Internet is a hot topic. Everybody's getting on line—individuals, companies, organizations, and researchers. But, once you're online, it's not always easy to locate information or identify the best Web sites. So for the newbies, I'd like to suggest a few sites.

You should start with an FAQ (Frequently Asked Question), which are Internet reports on topics ranging from microcontrollers to artificial intelligence, C++, and PCMCIA. The best way to find a FAQ is to FTP to MIT using <ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/news/answers>. Ohio State maintains a similar list of FAQs on the Web. Just type in <http://www.cis.ohio-state.edu/hypertext/faq/usenet/FAW-List.html>. Alternatively, you could browse Usenet groups such as news.answers, sci.answers, and comp.answers.

Your second best source of information is Usenet. The only problem is finding what newsgroups exist. FTP to <ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/news/answers/active-newsgroups>. This list includes all known groups with a description. Even better is Dejanews, which provides a free service to Usenet via the WWW. Just type <http://www.dejanews.com>. **Mailing lists** (<http://www.neosoft.com/internet/pam1>) offer discussions on specific topics.

The Web is the third best place to look. Cera Research maintains an overview of Web search engines at <http://www.cera.com/srcwww.htm>. **Search engines survey the Web and answer search questions.**

Archie is the easiest way to obtain source code. Access Archie via <http://www.man.net/Astra/AA.html>. It identifies locations of files on things like Motorola's 68HC11, C++, and assembly language programming.

It's also easy to publish on the Web. Individuals and companies publish hot lists devoted to specific topics. You can get in touch with these using Lycos or Yahoo.

This should get you started. Good luck and happy cruising.

Jason McDonald
Fremont, CA
jasonm@violet.berkeley.edu

FIRE THOSE ENGINES

I enjoyed Ed Lansinger's "Developing an Engine Control System" (*INK* 62-64) very much. I've only one minor nit to pick.

Although Ed's two coil ignition system is fine for drag racing, for a street application, the increased plug wear over a four-coil system would be unacceptable. There's a need for a feedback of resistance at the spark plugs

during various RPMs and load, which should also provide cylinder pressure calculations.

This change would further refine the ignition curve, boost control of the turbocharger, and could potentially help control the electrohydraulic actuation of the intake and exhaust valves so the camshaft could be done away with altogether. It could lead to controlling the AR (Active Radical) Combustion we now call four-cycle engines.

Jim Chaney
Airdrie, AB, Canada

COME ON EDITORS!

I really enjoyed Do-While Jones' "Digital Filter Alchemy" (*INK* 61), except I needed definitions for the symbols in a list. For example, that squiggly Greek letter-damping factor, yes?—and the subscripts on the omegas—what frequencies do they refer to!

I recognize that many readers are electrical engineers whose professors defined terms conventionally. But, those symbol definitions were the first things I forgot.

Come on, editors, lend me a hand.

Sayre Rodman
73210.540@compuserve.com

Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.

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

Contact our editorial offices at (860) 875-2199.

Fax: All faxes may be sent to (860) 872-2204.

BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300–14.4k bps, 8N1).

Internet: Letters to the editor may be sent to editor@circellar.com. Send new subscription orders, renewals, and address changes to subscribe@circellar.com. Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article.

For more information, send E-mail to info@circellar.com.

WWW: Point your browser to <http://www.circellar.com/>.

FTP: Files are available at <ftp://ftp.circellar.com/pub/circellar/>.

NEW PRODUCT NEWS

Edited by Harv Weiner

DEVICENET ADAPTER

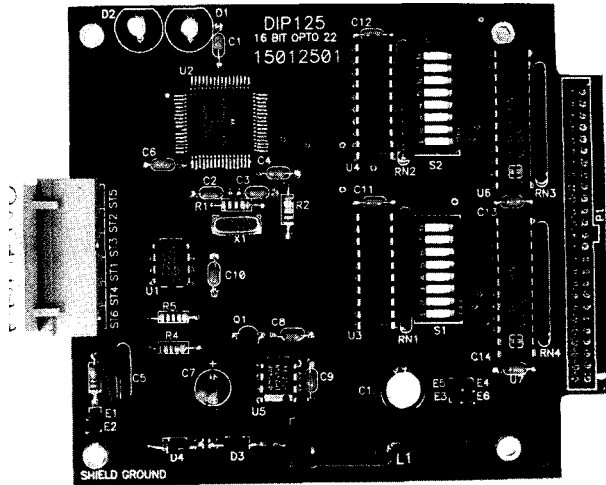
The DIP125 DeviceNet adapter enables the integration of low-cost OPTO-22 compatible plug-in I/O modules with industry-standard DeviceNet-compatible networks. The DIP125 is compatible with OPTO-22 PB4, PB8, and PB16 digital backplanes, so a wide range of digital interfaces can be connected to the network. To the master control node, the device appears as 16 bidirectional I/O points.

The DeviceNet network uses the CAN (Controller Area Network) standard to provide high-speed, robust communications between distributed I/O control nodes. Supported by a growing number of programmable controller and industrial control vendors, it has been applied to a wide range of industrial control applications.

The DIP125 control node provides 16 bidirectional I/O points on the network, enabling the user to select between input or output solid-state interface units. I/O is updated using a single DeviceNet P0 L L command. The unit may be powered from the DeviceNet 24-VDC bus power or from a local 5-V supply.

The unit operates at the standard DeviceNet speeds of 125, 250, and 500 kbps. The CAN protocol provides extensive error containment, and the DeviceNet application layer enables easy integration into systems using standard software tools.

The DIP125 sells for \$195.



D.I.P. Inc.

P.O. Box 9550 • Moreno Valley, CA 92552 • (909) 247-3342 • Fax: (909) 924-3359

#500

NONCONTACT DISTANCE MEASUREMENT

Senix announces a new line of low-cost, high-resolution OEM ultrasonic distance sensors for level, proximity, positioning, dimensioning, and other industrial applications. The OEM-ULTRA series measure distance from 0.25" to 10" with a maximum resolution of 0.06". Various interfaces are available, including 4-20-mA current loop, 0-10 VDC, relays, and serial-data communications (RS-232 and RS-485).

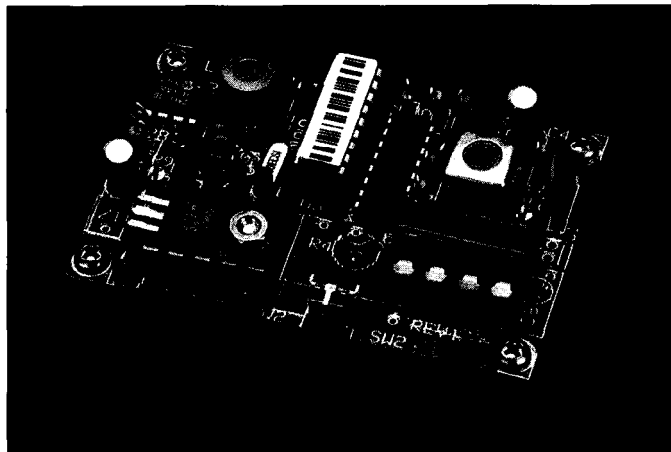
Ultrasonic sensors measure distance to materials without contact. They are insensitive to the material's color and other optical charac-

teristics. A sound pulse above the audio range is transmitted by the sensor, and the time of the echo reflection is measured. The distance to the target is then calculated based on the speed of sound. The OEM-ULTRA is especially de-

signed for high-volume applications in the automation, pulp and paper, chemical, textile, polymers, machine control, and printing industries.

OEM-ULTRA sensors are circuit-board configured. They can be mounted in the

user's equipment and connected with an attached transducer. Sensors can be packaged to meet specific mechanical requirements. Units range from simple, fixed-function proximity sensors to complete, packaged multisensor systems. Since the sensors are microprocessor based, OEM control functions can often be integrated into the sensor at little or no additional cost, eliminating the need for other expensive hardware.



Senix Corp.
52 Maple St.
Bristol, VT 05443
(802) 453-5522
Fax: (802) 453-2549

#501

NEW PRODUCT NEWS

EMBEDDED CONTROLLER

The RPC-330 from Remote Processing is an embedded controller that includes 8 A/D converter input and 2 D/A converter output lines with 12-bit resolution, 2 quadrature encoders/counters, 34 digital lines, and 2 RS-232/485 serial ports. The operator interface consists of a keypad and LCD character/graphics port. The programming environment includes a floating-point BASIC which supports all hardware.

A built-in temperature transducer monitors ambient temperatures. This feature is useful when the RPC-330 and other devices are operating in harsh environments and temperature control is necessary. A fan or heater can be activated using one of the high-current ports.

Two operational amplifiers buffer, amplify, and filter inputs from sensors. The temperature transducer and amplifiers can be jumpered directly to an A/D converter input. The A/D converter accommodates 8 single-ended or 4 differential inputs with ranges of 0-5 V or ± 2.5 V. Inputs are overload protected to ± 15 V.

The RPC-330 operates remotely or as part of a network using the RS-485 port, and is programmed using a PC with a serial port. The RPBASIC operating system, included with the RPC-330, is an improved version of Intel BASIC-52. Besides supporting the hardware directly (using commands like **DISPLAY**, **KEYPAD**, **COUNTER**, **LINE**), serial ports are now buffered and string handling has been greatly improved. Multitasking commands speed program execution.

Two 20-MHz multimode counters interface to X1, X2, or X4 quadrature encoders or other high-frequency devices. Up and down counters interrupt the program when a preset count is reached.

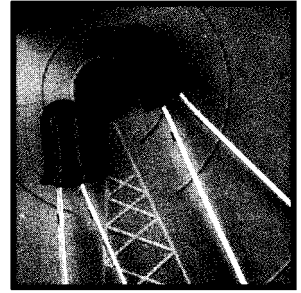
BASIC loads or reads a 24-bit number to the counter using the **COUNT** command. Using RPBASIC's multitasking feature, frequency measurements are possible.

The digital lines (one is an optically isolated input) interface to opto racks, switches, and other TTL devices. A high-current FET switches up to 2 A to control display backlighting, large relays, small motors, solenoids, heaters, and so on.

The operator interface consists of keypad and display ports. Most LCD displays interface to the display port, so a separate terminal with keypad is unnecessary, saving a serial port. RPBASIC positions the cursor and writes to the display using a single command. Graphics commands draw lines and control pixels to show level or position. BASIC automatically scans and buffers entries from the 24-key keypad port.

Power required is 5 V, and the operating range is -25°C to $+75^{\circ}\text{C}$. The RPC-330 sells for \$395 and includes a hardware manual and RPBASIC operating system. Real-time clock and battery backup module available as options.

Remote Processing
6510 W. 91 st Ave.
Westminster, CO 80030
(303) 690-1588
Fax: (303) 690-1875



SELF-SIGNALING LED

Lumex introduces the first low-battery-signaling LEDs to incorporate a CMOS chip inside a typical $T1\frac{3}{4}$ (5-mm) LED lamp. The device targets manufacturers of battery-operated end products who need a foolproof way to indicate that battery power is about to drop to zero.

Although designed to detect low power in two 1.5-V AA batteries, the LEDs also accommodate packs of 3-6 1.5-V cells by using external resistors. Applications include portable or hand-held consumer and industrial products using popular AA batteries.

The LEDs are available in red and yellow. Input voltage can be as low as 2.0 V with detection sensitivity of $2.3 + 0.1$ V and standby current of 5 μA . Light intensity at 6.5 mA (for red diffused) is 3 mcd minimum. Cost range for the LEDs is \$1.50 (single) to \$1.00 (1000+).

Lumex Opto/Components
290 E. Hellen Rd.
Palatine, IL 60067
(847) 359-2790

#502 Fax: (847) 359-2867 #503

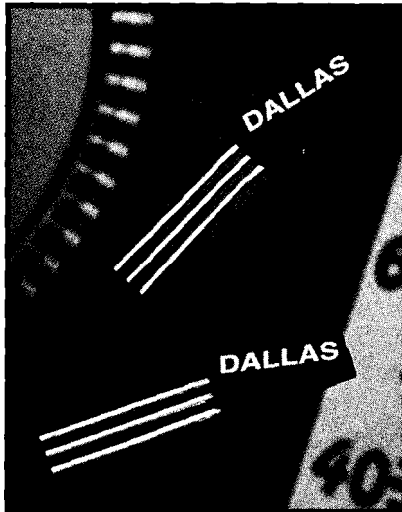
NEW PRODUCT NEWS

DIRECT-TO-DIGITAL TEMPERATURE SENSOR

Dallas Semiconductor introduces a direct-to-digital temperature sensor that can be multidropped. Multiple DS1820 **1-Wire Digital Thermometers** can be placed on a single twisted-pair wire, eliminating the complexity and reducing the expense of distributed temperature measurement.

Multidrop capability simplifies distributed temperature-sensing applications, whether the task is to measure temperature inside an electronics enclosure or a building. Multidrop capability is made possible because each DS 1820 has a unique serial number etched in silicon. Using this 64-bit lasered ROM, the Dallas proprietary 1-wire protocol identifies the temperature of a specific sensor. Thus, multiple chips residing on a 1-wire bus report back to a central processor, eliminating the need for separate wiring for each sensor. One-wire signals can travel for a distance of 300 m.

The DS 1820 provides 9-bit temperature readings. To communicate information, only one wire plus ground must be connected from a central microprocessor to a DS1820, enabling remote measurements, simplifying analog circuitry, and reducing the need for shielded cable. This feature is useful for applications such as HVAC environmental controls; sensing temperature inside buildings, equipment, or machinery; and process monitoring and control.



The DS1820 measures temperatures from -55°C to $+125^{\circ}\text{C}$ in 0.5°C increments. It converts temperatures to a digital word in 200 ms (typical) and requires no standby power. Power for reading, writing, and performing temperature conversions can be derived from the data line

itself, so there's no need for an external power source. The device stores energy in an internal capacitor when the signal line is high and continues to operate off this power source during the low times of the 1-wire line until it returns high to replenish the capacitor supply.

To ensure that temperatures stay within the required range, the DS-1820 features a user-definable, non-volatile temperature-alarm setting. An alarm search command identifies and addresses devices whose temperature is outside the programmed limits so the user can adjust the temperature.

The DS1820 1-Wire Digital Thermometer is available in a 3-lead device or as a 16-pin SSOP for surface-mount applications. The DS1820 costs \$2.77 in OEM quantities.

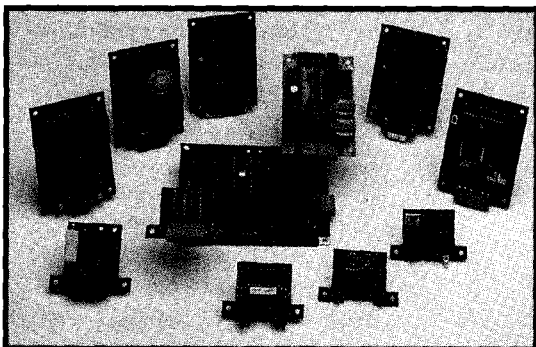
Dallas Semiconductor
4401 S. **Beltwood** Pkwy. • Dallas, TX 75244-3219
(214) 450-0448 • Fax: (214) 450-0470

#504

GATEWAY DEVICES FOR EMBEDDED CONTROL SYSTEMS

parvus has introduced a new **Gateway family** of low-cost, easy-to-use Gateway and shared memory devices. Developers of embedded control systems can interconnect control-system networks which use incompatible communication protocols or add networking capability to existing platforms without hardware modification and with minimal software overhead. Initial offerings of the Gateway product family support for CAN, parvNET, and Echelon **LON-Works** networks. Components include control modules, network media drivers, customizable gateways, I/O modules, micronodes, interface modules, and prototyping products.

The shared memory devices create a window within conventional memory planes which enables data sharing or communication with disparate computers, intelligent nodes, and networks. parvus supports its shared memory devices with an array of network I/O and interface modules.



parvus Corp.
1214 Wilmington Ave., Ste. 100
Salt Lake City, UT 84152-1045
(801) 483-1533 • Fax: (801) 483-1523

#505

FEAT'URES

12 Updating a Classic:
the 280380
Microprocessor

18 Connect the Personal
PBX to the Real World

26 LPT:*Analog!*

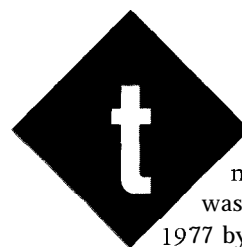
34 PIC17C44 High-End
Microcontroller
Architectural Overview

FEATURE ARTICLE

Monte Dalrymple

Updating a Classic: the 280380 Microprocessor

After reviewing the Z80's popularity, Monte introduces the Z380, a new chip offering a full complement of 16-bit data manipulations and enough 32-bit instructions to handle the new 32-bit addresses.



The original Z80 microprocessor was introduced in 1977 by Zilog. It quickly became one of the most popular 8-bit microprocessors ever. In fact, it was in the running for the original IBM PC. However, the lack of a clear upward migration path kept the Z80 and its derivatives in the realm of embedded control, with little glamour but huge volumes.

In the 1980s, the Z180 was released with on-chip peripherals and a faster CPU. It was followed by the Z280, which attempted to bring minicomputer power to the architecture. While the Z180 was hugely successful, the Z280 was only marginally so. Today, you still find a Z80 in many modems and a Z180 in most inkjet printers.

Despite these advances, the Z80 architecture has been limited to 8 bits for data and 16 bits for an address size. True, the Z180 and Z280 add memory management units (MMUS) to expand the physical address space, but they both limit the logical address space (the amount of memory actually accessible by a program at one time) to 16 bits.

The Z380 microprocessor was developed to meet both of these constraints in addition to boosting the performance of the CPU itself.

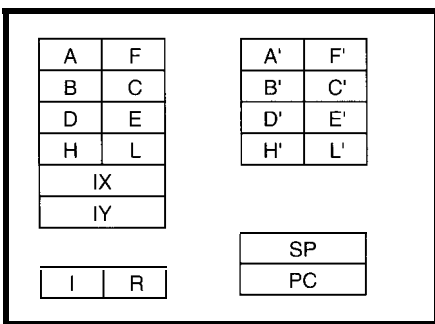


Figure 1—The register set of the original Z80 contains an alternate bank that is ideal for fast interrupt handling, and 16-bit index registers that are dedicated to the indexed addressing mode.

The Z80 is available with a maximum speed of 20 MHz, which works out to 7.3 MIPS maximum. In contrast, the 2180 runs at 33 MHz or 8.3 MIPS maximum. Of course, these maximums are never achievable because most instructions require more than the minimum number of clocks to execute. As well, the memory interface at these speeds is difficult at best.

The Z380 is currently available in 10-MHz (3 V and 5 MIPS) and 16-MHz (5 V and 8 MIPS) versions. With the Z380, it's easier to get close to these maximum numbers because of the improved execution unit and optimized bus interface.

All previous versions of the Z80 architecture have been excellent at handling 8-bit data and had rudimentary 16-bit data manipulation instructions for 16-bit addresses. However, many of today's embedded control projects demand both a larger address space and wider data.

The Z380's instruction set remains 100% binary compatible with the Z80 and 2180, but adds a full complement of 16-bit data-manipulation instructions. It also includes rudimentary 32-bit data-manipulation instructions to handle the new 32-bit addresses.

The address space of the Z380 is a full 32 bits, linearly accessible. All internal data paths are 32 bits wide,

while the external data path is limited to 16 bits to keep packaging costs down.

Binary compatibility is still a big selling point, despite the press about high-level language programming. Many users have huge investments in proprietary code written in assembly language. The Z380 enables them to run this code without change.

For users writing new code, several additions to the instruction set include:

- relative jumps and calls with 1-, 2-, or 3-byte offsets
- stack relative instructions
- 32-bit linear address space

These features make the Z380 a competitive processor.

REGISTER SETS

The original register set of the Z80 architecture is shown in Figure 1. The accumulator (A) is the destination of byte data-manipulation instructions. The other registers can be used for addresses or data, with the HL register pair as the destination for 16-bit data-manipulation instructions.

The alternate register bank offers fast context switching or a scratchpad in small RAMless systems. This alternate bank was one of the major advantages of the original Z80. However, there is no alternate version of the IX and IY index registers in the original Z80 architecture.

The Z80 also has a 16-bit Stack Pointer (SP) and a 16-bit Program Counter (PC). The Interrupt (I) register creates an interrupt vector, and the Refresh (R) register provides a refresh address for previous-generation DRAM.

Figure 2 pictures the register set of the Z380. It is a superset of the original Z80 register set with the addition of alternate index registers. All of the registers (except for A) are expanded to a full 32 bits. Like the original Z80,

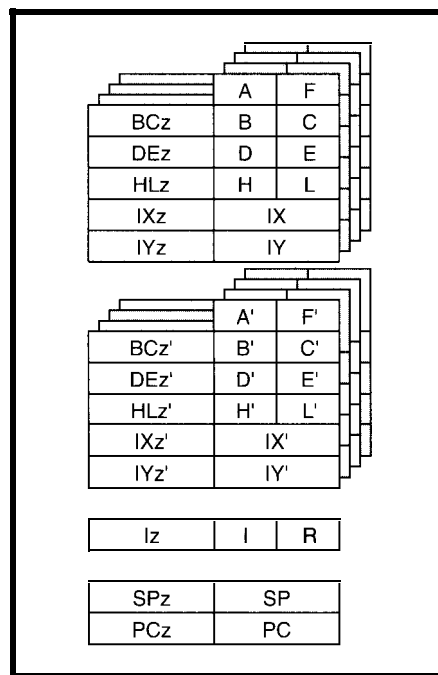


Figure 2—in the Z380, almost everything is widened to 32 bits, and now there are alternate index registers, plus four copies of the full register set.

the Z380 uses the HL register pair and its extension as the destination for 16- and 32-bit data-manipulation operations. The SP and PC registers are also widened to 32 bits.

But, this expansion of the original register set is only the beginning. The Z380 supports 4 copies of each register set, and the architecture itself supports 128 copies!

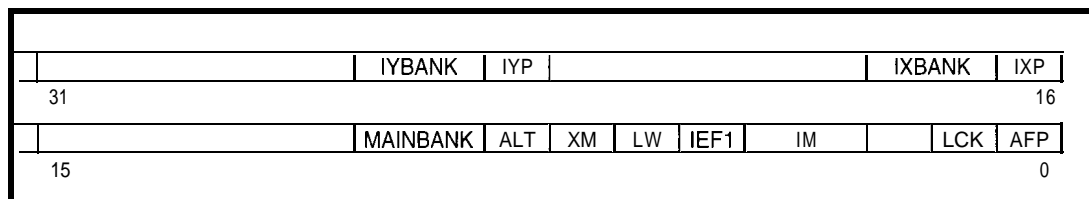
Switching between register sets is controlled by the Select Register (SR), which can also be read to determine where the program is operating. The SR can be pushed to and popped from the stack as part of a context switch.

Other bits in the SR tell the program the current operating mode of the processor (more on this later), the interrupt mode, and the current state of the Interrupt Enable flag (IEF1). Figure 3 shows the SR's organization.

INSTRUCTION SET

The expansion of the register set is only part of the solution to the limita-

Figure 3—The 32-bit Select Register (SR) controls the various modes of the Z380. Notice that the bank fields can be expanded to seven bits if necessary to meet future requirements.



tions of the Z80 architecture. The other part is the instruction set itself. A number of approaches were used in creating the Z380's instruction set, depending on the relative importance or frequency of the desired result.

For example, the original Z80 instruction set has a jump-relative instruction with an 8-bit displacement. The 2380 was deemed to require the option of larger displacements as well as a relative address version of the CAL L instruction.

Because these types of instructions occur frequently in programs, separate opcodes were assigned for 16- and 24-bit jump-relative instructions. New opcodes were assigned for the 8-, 16-, and 24-bit relative-address CAL Ls.

But, such an approach was not feasible for all the instructions that use direct addressing or contain immediate data because there are so many. So, the concept of a decoder directive was introduced.

Simply put, a decoder directive is an escape sequence that tells the instruction decoder to expect more than the usual number of bytes of direct address or data in the instruction.

For example, the DD I R I B (decoder directive immediate byte) prefix tells the decoder to fetch one additional byte beyond what it would normally expect with the instruction immediately following.

Likewise, the DD I R I W [decoder directive immediate word) prefix tells the decoder to fetch two additional bytes with the instruction immediately following. In this way, wider data and addresses can be used with the existing opcodes.

Decoder directives also expand the instruction set to handle 16- and 32-bit

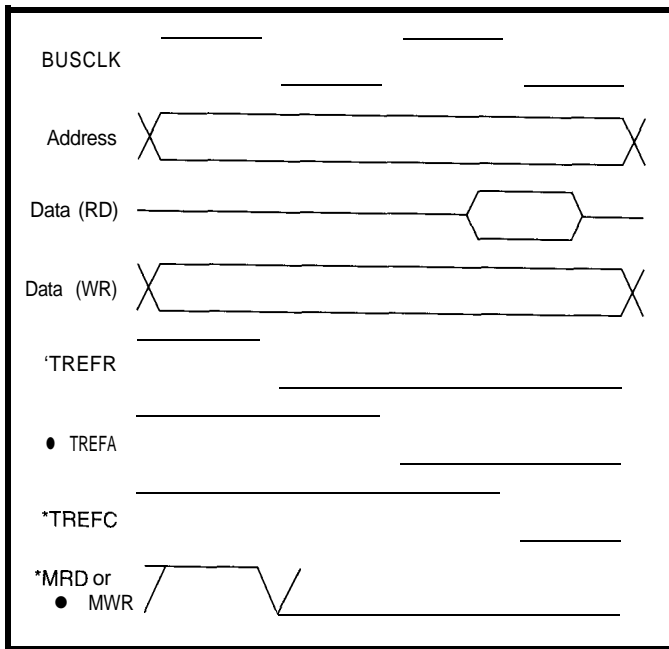


Figure 4—Memory reads and writes take two clock cycles. Three different timing signals are available, making DRAM interfacing simple.

data. The 2380 can operate in either word mode or long-word mode, as selected by a bit in the SR. In word mode, all 16-bit data-movement and data-manipulation instructions operate on 16-bit data. In long-word mode, all 16-bit data-movement and data-manipulation instructions actually operate on 32-bit data.

Decoder directives enable shorter or longer data and addresses to be manipulated as required by the application. Of course, since the Z80 had only

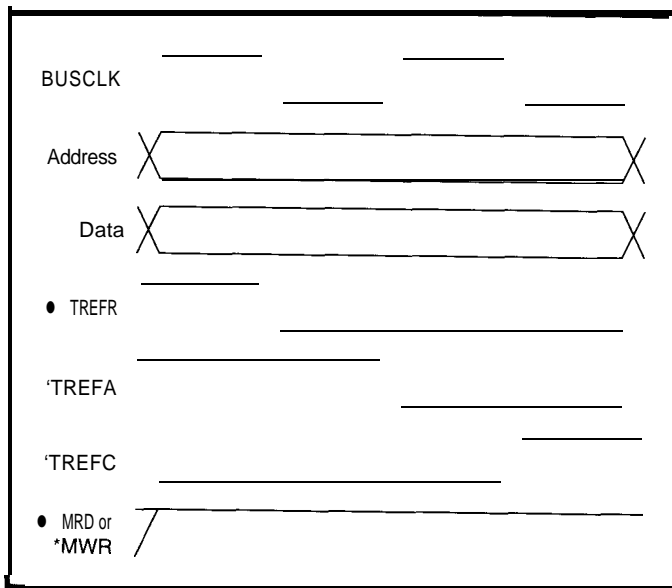


Figure 5—Modern DRAMs do a refresh when the order of the RAS and CAS is reversed. The Z380 reverses the order of the *TREFR and *TREFC when doing a refresh cycle.

rudimentary 16-bit data-manipulation instructions, new opcodes were required to add a normal complement of 16-bit data-manipulation instructions.

WHICH MODE TO USE?

This information is fine for the data processed by the CPU, but what about the CPU's program address space? This is where the final mode bit in the SR comes into play.

The 2380 can operate in either native mode (the default at powerup or reset) or extended mode. This choice of mode is controlled by the X bit in the SR and is set by a special instruction.

In native mode, the PC only increments across 16 bits, and the SP increments and decrements across 16 bits. In this mode, the two registers are just as if they are only 16 bits wide, and the data pushed to and popped from the stack is also 16 bits wide.

Native mode is completely compatible with the 280. Yet, it still offers access to memory addresses larger than 16 bits for data. Although 32-bit addresses can be manipulated by the program, the address space for code is only 16 bits wide.

Extended mode is another story. Here, the real power of the 2380 is available to the user. In this mode, the PC, SP, and stack operations (at least calls and interrupts) are of full 32-bit width.

Because of the difference in the information on the stack during a subroutine call or an interrupt service routine (ISR), this mode is not strictly compatible with the Z80. Although this incompatibility is an unavoidable byproduct of the architecture's expansion, it is easily handled when parameters are passed on the stack.

Notably, the selection of extended mode is one way only. That is, an instruction lets you enter extended mode, but the only way to go back to native mode is through reset. This restriction prevents users from worse-than-death fates should they return to native mode accidentally or while return addresses are stored on the stack.

THE VECTORED INTERRUPT

One of the key advantages of the Z80 over its contemporaries is the vectored-interrupt capability. This capability enables the I register to be used in conjunction with an 8-bit vector that is returned to the CPU by the interrupting device during a special interrupt-acknowledge cycle.

The concatenation of the vector with the contents of the I register is used as an index into a memory-located table which contains the starting address of the appropriate ISR.

The 2380, as already shown, has an expanded I register, so the ISR index table can be located anywhere in the 32-bit address space. For more peripherals (or more interrupt-type encoding), the 2380 adds another interrupt mode to the three already present in the Z80. This new mode lets the interrupting device return a 16-bit interrupt vector. In all other respects, the operation of the interrupts is the same as in the Z80.

PERIPHERALS

While the 2380 is only a microprocessor, not a microcontroller, it does contain a couple of useful peripherals. The first is a refresh controller for DRAMs. This controller can be programmed for different refresh intervals and can accumulate up to 255 missed refresh cycles if the 2380 is not in control of the bus all the time.

The refresh controller does not provide a refresh address. Rather, it uses the CAS-before-RAS refresh that's built into all modern DRAMs. In this refresh mode, the DRAM selects the address to refresh, and the refresh cycle is triggered by a special sequence of control signals to the DRAM.

As you'll see in a moment, the 2380 bus interface provides for a sim-

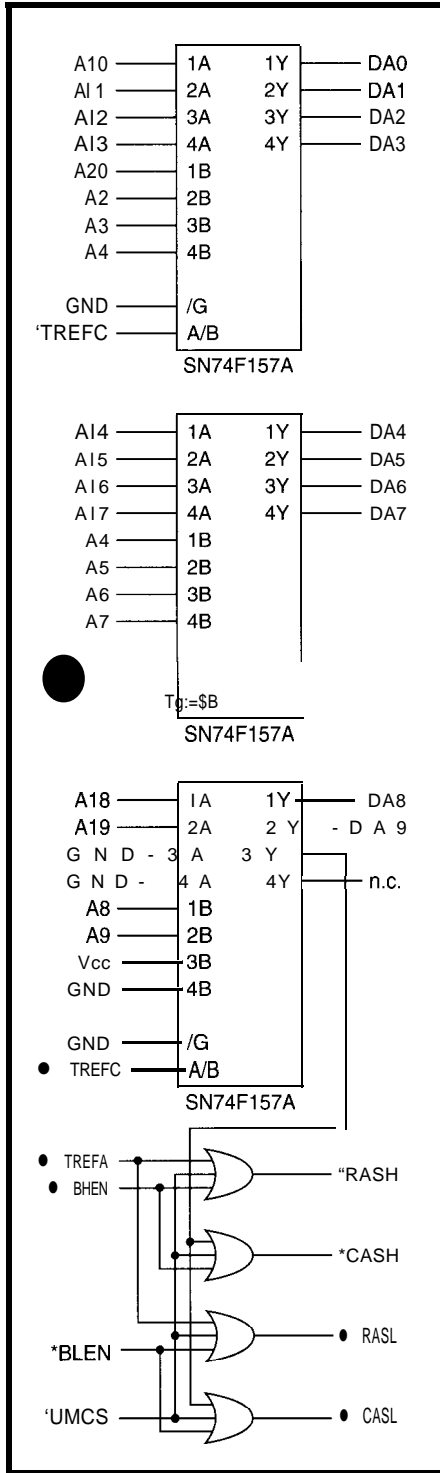


Figure 6—Interfacing a Z380 to a pair of DRAM SIMMs is simple. All you need are muxes for the address and four gates to control RAS and CAS for the two bytes.

ple interface to DRAMs and includes the correct signal sequencing to trigger this special refresh cycle.

The second peripheral included in the 2380 is a chip-select and wait-state generator. Six separate chip selects are available with a wait-state generator for each. There are also wait-state

generators for interrupt-acknowledge, return-from-interrupt, and I/O cycles.

Each chip select can be programmed to be active or inactive during refresh cycles. The chip selects can be programmed as a lower, an upper, and four contiguous midrange selects, or as lower, upper, and everything-in-between selects.

Typically, the lower chip select—which is always enabled after reset—controls ROM. The upper chip select controls either SRAM or DRAM for the stack. And, the midrange selects are for SRAM, DRAM, or ROM.

BUS INTERFACE

Another unique feature of the 2380 is its bus interface. It has the normal 32-bit address bus and 16-bit data bus, but the control signals for these buses are unique.

There is one set of control signals optimized for the memory interface and a completely separate set optimized for the I/O interface. The two sets of interface signals operate at different speeds, with the I/O interface speed being under program control! This dual arrangement dramatically reduces the amount of glue logic required to build a system out of the 2380.

Figure 4 shows the timing and signals for the memory interface. In addition to the normal *MRD (Memory Read) and *MWR (Memory Write), there are three separate timing reference signals active at different times during a memory cycle.

To gain complete control over the exact timing of the three control signals, the *WAIT signal is sampled, not once, but three times during the cycle. In addition, the MSIZE (Memory Size) signal is sampled during every memory transaction to determine whether the memory being accessed is 8 or 16 bits wide. Thus, a byte-wide boot ROM can be used with 16-bit primary system memory.

The three timing signals have a different relationship during a refresh transaction (see Figure 5). The two-clock-cycle timing of the bus matches the two-clock-cycle nature of the CPU itself. The bus is therefore eliminated as an execution speed bottleneck.

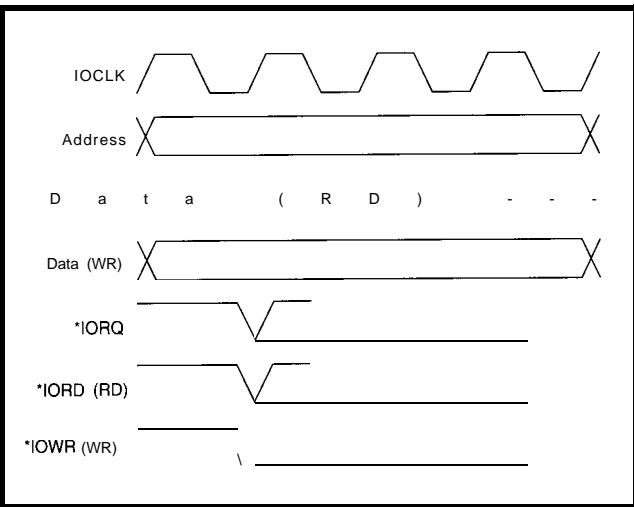


Figure 7—The timing for I/O transactions looks just like it does in a Z80. In this case, IOCLK is a scaled version of the processor clock and is under program control.

As you can see in Figure 6, these three signals make interfacing the Z380 to DRAM simple. Here, the left-side signals are from the Z80380, and the right-side signals connect to a standard DRAM SIMM. This scheme is used in the Zilog evaluation board for the device. It assumes one T3 wait state, which allows for 80-ns DRAM.

The timing and signals for the I/O interface are shown in Figure 7. They enable direct connection to any of the Z80 peripherals, as well as the more common Z85xx series of peripherals such as the 28530 SCC.

The timing of the I/O interface is set by the signal IOCLK, which can be BUSCLK divided by two, four, six, or eight. Therefore, the Z380 can run at full speed while employing slower peripheral devices. The IOCLK speed is set by a field in an internal I/O control register and can be changed as needed on-the-fly.

EVALUATING THE Z380

Zilog provides a simple evaluation board for the Z380 which contains EPROM, SRAM, and DRAM sockets, and a pair of serial I/O ports for connection to a terminal or PC. A simple debug monitor, originally written for the Z80, is included with source code. Also included are a Z80 assembler and linker that run on a PC.

Unfortunately, none of the software provided takes advantage of the power of the Z380, but it's better than nothing. Third-party support for the Z380 is available in the form of an assembler, C compiler, and emulator.

Even though limited support makes using the Z380 a bit of a challenge, its price (about \$10) and ease of interfacing to memory and I/O are appealing. These features should make the Z380 a serious contender for those projects where you can't do it in a single-chip microcontroller, and you don't want to use a full-blown PC. □

Monte J. Dalrymple worked for Zilog for over 15 years, designing many of the company's most successful chips. He was the architect and one of the designers of the Z380. Currently, he is designing new chips for Systemeye International. Monte may be reached at hcdp38a@prodigy.com.

SOURCES

Z80-Z380
Zilog, Inc.
210 East Hacienda Ave.
Campbell, CA 95008-6600
(408) 370-8000
Fax: (408) 370-8056

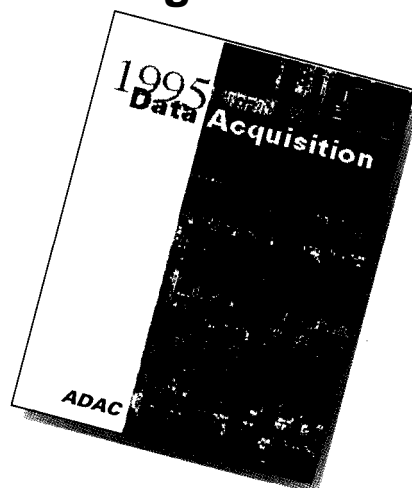
REFERENCES

Zilog, Inc. *Z380 Microprocessor Unit User's Manual*, 1994.
Zilog, Inc. *Z380 Microprocessor Unit Preliminary Product Specification*, 1994.

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

FREE Data Acquisition Catalog



1995

PC and VME data

acquisition catalog

from the inventors of
plug-in data acquisition.

Featuring new low-cost

A/D boards optimized

for Windows,

DSP Data Acquisition,

and the latest

Windows software.

Plus, informative

technical tips and

application notes.

Call for your free copy

1-800-648-6589

A D A C

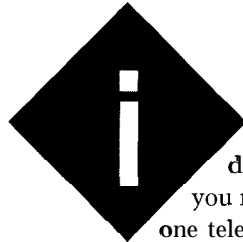
American Data Acquisition Corporation
70 Tower Office Park, Woburn, MA 01801
phone 617-935-3200 fax 617-938-6553

#106

Connect the Personal PBX to the Real World

FEATURE ARTICLE

Richard Newman



In *INK* 48, I addressed everything you need to call from one telephone to another, including dial tone, DTMF (Dual Tone MultiFrequency) decoding, ringing the called party, and setting up a conversation through an analog-switch matrix.

In this article, I want to take you outside your own phone system. I present a circuit that interfaces the Personal PBX, a small switching system, to the outside world so that a telephone company central-office interface may be connected.

I'll also present an improved subscriber-line-interface circuit (SLIC) which, when coupled with the central office (CO) interface, reduces the amplitude and frequency loss of the conversation.

SLICs AND COs

Let's review the difference between a SLIC and CO interface. The SLIC provides loop current to the telephone set. Ringing signals indicate a call has arrived.

The CO interface terminates the loop current provided by the central office and detects a ringing condition applied to the line. These definitions apply to the system you are building or working on. A CO interface can:

- detect a call sent from the central office
- terminate the line answering or originating a call
- interface the central office with the speech matrix transferring the audio from one 2-wire circuit to another
- detect line current indicating if the caller hung up before we have

- protect itself from lightning and overvoltage surges by breaking the physical connection when a failure occurs

Figure 1 shows the line interface. As you can see, it includes metallic coupling, loop-current sensor, polarity reversal correction, and the line seizure relay.

When relay K1 is not activated, DC loop current does not flow. The circuit is considered "on hook" or ready to receive an incoming telephone call.

When the central office applies 90 VAC at 20 Hz to ring the line, AC current passes through the nonpolarized coupling capacitor C1. This AC current flows through the optoisolator. Its output at the not-signal point is a 20-Hz square wave which is TTL compatible.

The not-signal output is fed to the system's microprocessor. Its frequency and duration reflect the central office's ringing pattern. Personalized ringing, available to electronic central offices, enables up to three telephone numbers to signal one physical telephone line. The pattern of the ringing determines which logical line is signaled. Through software which decodes this pattern, the switch determines the personalized ring signaled by the central office.

Because the transformer T1 is in the circuit when incoming ringing voltage is applied, the circuits attached to the transformer's secondary must be protected. Zener diodes D1 and D2 clamp the waveform to whatever voltage you choose. For example, a 12-V zener limits the positive and negative swings to around 12 V.

When K1 is activated, loop current from the central office flows through the optoisolator and causes it to light continuously. The opto's not-signal output goes low and stays low for as long as loop current is flowing.

Any time the central office sends a supervision pulse—which appears as a momentary loss of loop current—the opto's output momentarily goes high. A supervision pulse is sent from the CO when the calling party hangs up before you do.

Current flowing through T1 causes the audio signal on the CO to couple

Richard presents a circuit that connects the Personal PBX small switch with the "outside" world—the telephone company central office. A SLIC interface used in conjunction with the PBX enhances conversation quality.

with the 2-wire analog port and vice versa.

Diode D3 protects the opto during spikes when the circuit is seized. Fuses F1 and F2 protect both the interface and central office from failure or high-voltage surges. When a surge hits tip, ring, or both, one of the MOVs sinks the current to earth ground, and the associated fuse opens, and the fuse opens, arcing occurs if the current is sufficient, so the MOVs need to be of sufficient rating to handle it.

Resistors R1 and R2 may not be necessary in everyone's opinion, but they provide a small drop across them. This drop reduces the amount of cur-

SIGNAL CONVERSION

Figure 2 shows the 2-wire-to-2-wire converter which interfaces the 2-wire port of the line interface to the 2-wire port of the analog-switching matrix.

The 2-to-2-wire converter consists of two back-to-back 2-to-4-wire converters. The gain in any one direction is always less than one. However, the complexity of the implementation depends on the terminating impedance of the 2-wire side as well as the transmission characteristics of the CO line, SLIC, and switch matrix.

Looking at the right half of Figure 2, you can see that a typical 2-to-4-wire converter is made up of two op-amps

ceive signal has a portion of the transmitted signal with it and you try to convert it to 2-wire, the converters form a closed loop and oscillate.

If you reduce the gain, you eliminate the oscillation, but your goal of reducing the loss by adding gain won't be accomplished. You might as well have connected the coupling transformer directly to the switch matrix.

The measure of the converter's ability to separate the transmit and receive signals is called *transhybrid rejection*. Depending on the application, there are several ways to improve rejection. A CO interface might be more reactive with elements of resis-

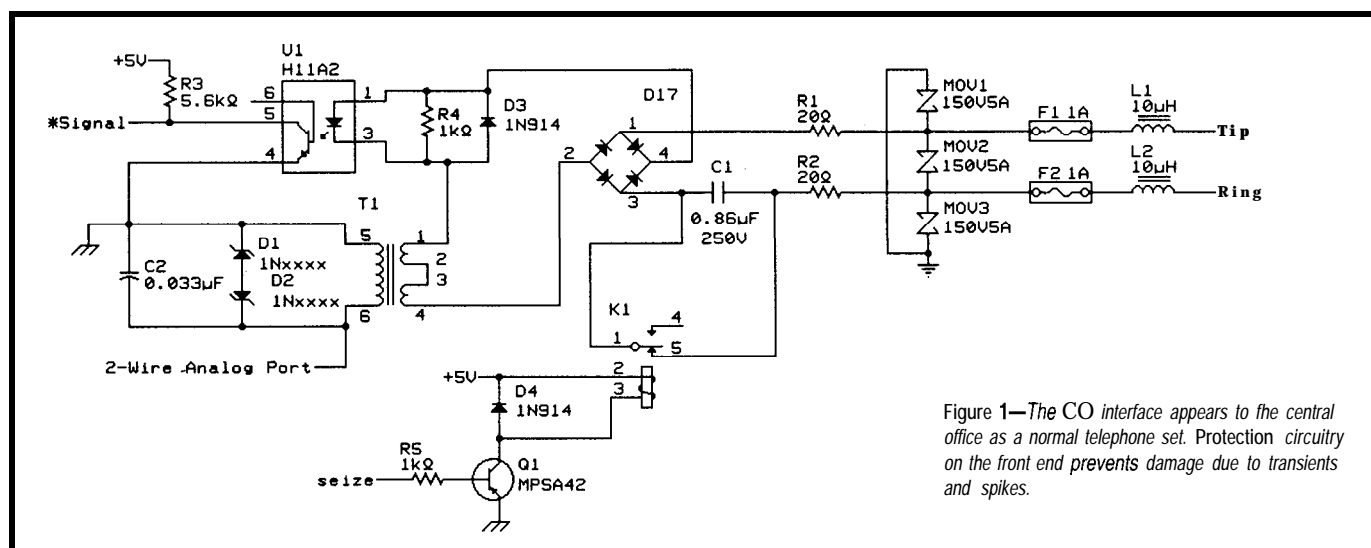


Figure 1—The CO interface appears to the central office as a normal telephone set. Protection circuitry on the front end prevents damage due to transients and spikes.

rent across T1 so a wider variety of loops can be accommodated. These resistors are required for real-world performance.

Diode bridge D1/D2 permits the CO's connection to the interface regardless of its polarity or whether the CO inflicts polarity reversals on the circuit. Again, real-world situations require that anyone be able to hook up a telephone line. The bridge ensures that hook up occurs even if the wires are reversed.

Relay K1 is activated by a simple inverting driver. A TTL low on the seize input causes the interface to take the telephone line off hook and draw a dial tone from the CO. A high on the seize input causes the interface to drop the call (if one was in effect) and return the telephone line to the idle or on-hook state.

and a handful of discrete components. This converter transmits the input signal from the 4-wire input side to the 2-wire bidirectional side but keeps the signal from showing up on the 4-wire output side. It transmits to the 4-wire output any signal which shows up at the 2-wire bidirectional side but not on the 4-wire input.

Since the converter separates a transmit signal from a receive signal, you can add some gain to the output of the converter to overcome the insertion loss imposed by the physical-line interface. The amount of gain you add depends on how well the converter separates the signals. If it can't do the job well, the transmit signal appears in the receive signal.

While this mixing of signals might be acceptable for some applications, it's disastrous for ours. When the re-

tance and capacitance, but a very short SLIC might be almost purely resistive.

Resistor R6 is 680 Ω because most central office lines lie somewhere in the range of 600-900 Ω. Note that the resistor's value matches the attached line's impedance, not the resistance of the transformer's secondary.

Resistor R21 is the terminating impedance for the analog-switch matrix. Because R21 faces another resistor of the same value (R21 in the other circuit), when two 2-to-2-wire converters are in a conversation through the switch matrix, the reflected energy is minimized and rejection should be excellent.

Continue looking at the right side of Figure 2. The signal applied to U2a pin 3 appears at the 2-wire analog port but not at U2c pin 7. The input signal is subtracted from the output signal.

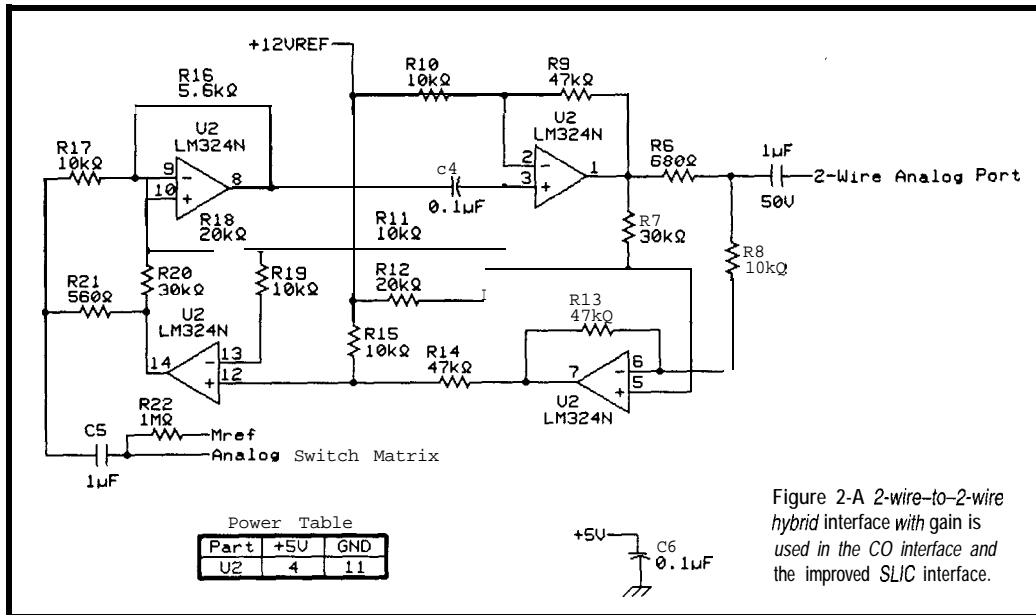


Figure 2-A 2-wire-to-2-wire hybrid interface with gain is used in the CO interface and the improved SLIC interface.

You therefore minimize loss of the signal injected into the 2-wire analog port and transferred to the balanced subscriber line. The small loss imposed by the switch matrix and capacitive coupling can be reduced further by the gain added in the 2--to-2-wire converter.

A typical call, shown in Figure 4, arrives at the phone line in 2-wire format. It terminates when the CO interface activates. The audio signal is coupled to the interface's 2-wire output, and from

The lefthand side operates identically. Resistor R22 provides the DC carrier for the AC signal to ride through the single-supply switch matrix. The DC carrier is less than half the switch-matrix supply voltage.

Note that the Personal PBX is a positive-supply-only system. Thus, all of the analog portions have a reference supply. In this case, reference is 12 V.

In my last article, I described a SLIC that had a transformer in it with a purely resistive feed. While this works well, we can do better.

THE SLIC INTERFACE

Figure 3 shows the SLIC which must be used with the converter illustrated in Figure 2. The features of this model include:

- reduced parts count
- elimination of the transformer
- a constant current feed that gives subscriber-line low-impedance characteristics to DC and high-impedance characteristics to AC

The SLIC shares the same relay drive and current detector as the CO interface, so controlling the SLIC is uniform in the system. Making stalling low deactivates relay K1, so the telephone set is attached to the audio inter-

face. When stalling is high, tip is removed from the audio interface and applied to a common ringing bus so that the telephone's bell can operate.

When the phone attached to tip and ring is lifted, current flows through R5 and R6 (both 20-Ω resistors). Current is limited to a rate set by the ratio of R2 to R4 and the resulting voltage (measured at the base of the PNP transistor Q2). When current flows, the optoisolator lights, signaling the CPU that the telephone set has been lifted off hook.

The act of limiting the set's current to one specific flow rate makes the subscriber-line low-impedance to DC signals (the 24-VDC carrier which powers the telephone set) and high-impedance to AC signals (the speech which rides on top of the DC carrier).

there, it is passed to the first 2--to-2-wire converter. Inside the converter, it is made 4-wire by separating the transmit and receive portions.

The separate signals receive added gain in the process. They are then recombined into another 2-wire signal, which is sent to the analog-switch matrix. This matrix passes the signal to the selected output where the process is reversed, only now the signal is sent to an extension interface. The result is a quality, full-duplex conversation between a telephone extension on the PBX and the outside phone line.

SOFTWARE CONTROL ISSUES

If you downloaded the code for the original project from the BBS, you'll notice that the system CPU must keep

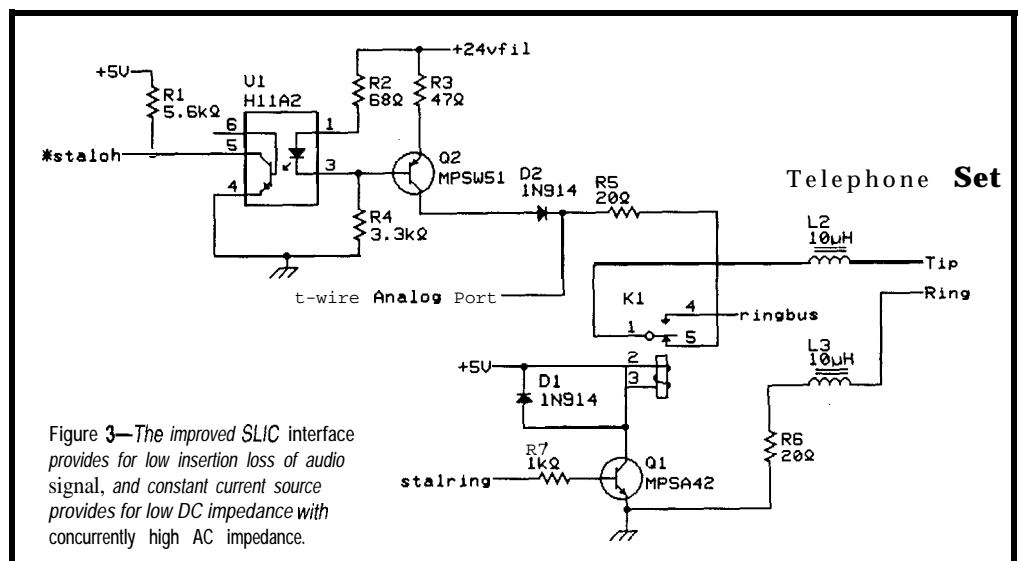


Figure 3—The improved SLIC interface provides for low insertion loss of audio signal, and constant current source provides for low DC impedance with concurrently high AC impedance.

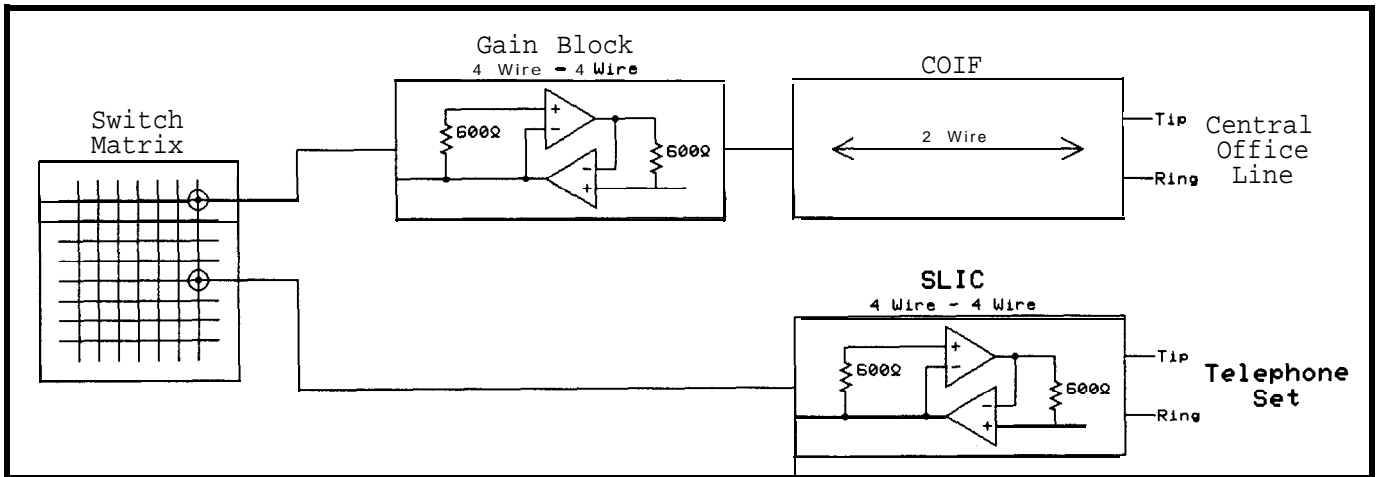


Figure 4— A conversation goes through a number of building blocks to get from the CO to the telephone set.

track of finite and shared resources. One DTMF receiver, for example, can service all the phones on the system, albeit one at a time.

Once the finite resource is used and no longer needed, it must be freed up, even if the call established is still in progress. This same resource management and queuing must be applied to the central-office circuit, touch-tone sender, and call-progress receiver.

Since you're adding central-office-calling capability to your switch, you must now make a decision about how external calls are to be set up. *Set up* refers to the way a customer's dialed number is conveyed to the equipment connected to the central office interface.

It includes two choices. The first, called **pass through**, does not require a system to have physical DTMF sender

hardware. This option saves on cost and board space.

The second choice is called either *store and forward* or **selective/adaptive routing**. This option requires a DTMF generator to be available to the system.

If you choose pass-through routing, you can implement features such as toll control. However, you can't selectively route the call to a preferred

CADPAC

PCB AND SCHEMATIC TOOLS
2 for the price of 1

PCB II & SUPERSKETCH FOR NEW LOW LOW PRICE

only \$159 US

** features comparable to packages costing thousands!
** must be tried to be believed
** "easiest product to use for designing PCBs"
** customers call it "the 8th wonder of the world!"

!!REDUCED!!

R4 SYSTEMS INC.
1100 GORHAM ST. SUITE 11B-332
NEWMARKET ONTARIO
CANADA L3Y 7V1
905 898-0665 FAX 905 898-0683

FREE DEMO
WRITE OR CALL
TODAY

Email r4system@astral.magic.ca

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

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

508-369-9556
FAX 508-369-9549

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

destination to implement features like least-cost routing or alternate operator services. These features are not possible because you don't have a DTMF sender and must depend on tones from the subscriber's line to dial out to the CO interface.

When a customer picks up a telephone set, they receive a dial tone from the PBX tone generator. The customer *must* dial a trunk-group access code, like 8 or 9. The PBX then conferences in the selected central-office circuit because the SLIC, DTMF receiver, and CO interface share a common audio path.

If you don't want toll control, you can drop the DTMF receiver at this point and attach the SLIC directly to the CO interface. As far as the PBX is concerned, as soon as the trunk-group access code is dialed and the user receives the outside dial tone, the call is set up and in progress. No common resources are assigned, and the call's next state is to hang up.

If you choose the pass-through arrangement and want toll control, you

must monitor the digits the customer dials to the outside line. If the customer dials digits conflicting with a preferred dialing pattern, the CPU releases the outside line and gives a busy signal from the PBX's dial-tone generator.

Once the toll-control plan is satisfied (i.e., enough digits are dialed that the system knows where the call is going), it releases the DTMF receiver even if a complete number is not dialed. After all, if the customer dials anything other than 0 or 1 for the first digit, you can be sure it won't be a long-distance call. There's no need to keep the DTMF receiver assigned?

There's one exception to this rule. To provide station message-detail recording (a record of the date, time, trunk, station, and number dialed) to the serial port when the call hangs up, keep the DTMF receiver assigned long enough to get the complete number.

STORE AND FORWARD

If you implement either the store-and-forward or adaptive/selective-

routing algorithm, you'll be rewarded with superior flexibility in handling a call. When users dial, they dial into the PBX. The central processor stores dialed numbers in a queue and analyzes them to determine the most efficient and inexpensive way to send a call through the public network.

As the user dials, the central processor sets up a concurrent call within the system between the CO interface, a DTMF sender, and a call-progress receiver. This call has its own speech path, so the user doesn't know a concurrent call is in progress and can't hear the interaction between the CO and CPU setting up the outgoing call.

Using store and forward, the CPU can insert numbers into the user's dialed digits. It can even call an alternate long-distance carrier, wait for a second dial tone, and dial the customer's long-distance account code plus the number that the user originally dialed from the calling station.

Again, the network-access-control activity is going on behind the scene without the user's knowledge. The

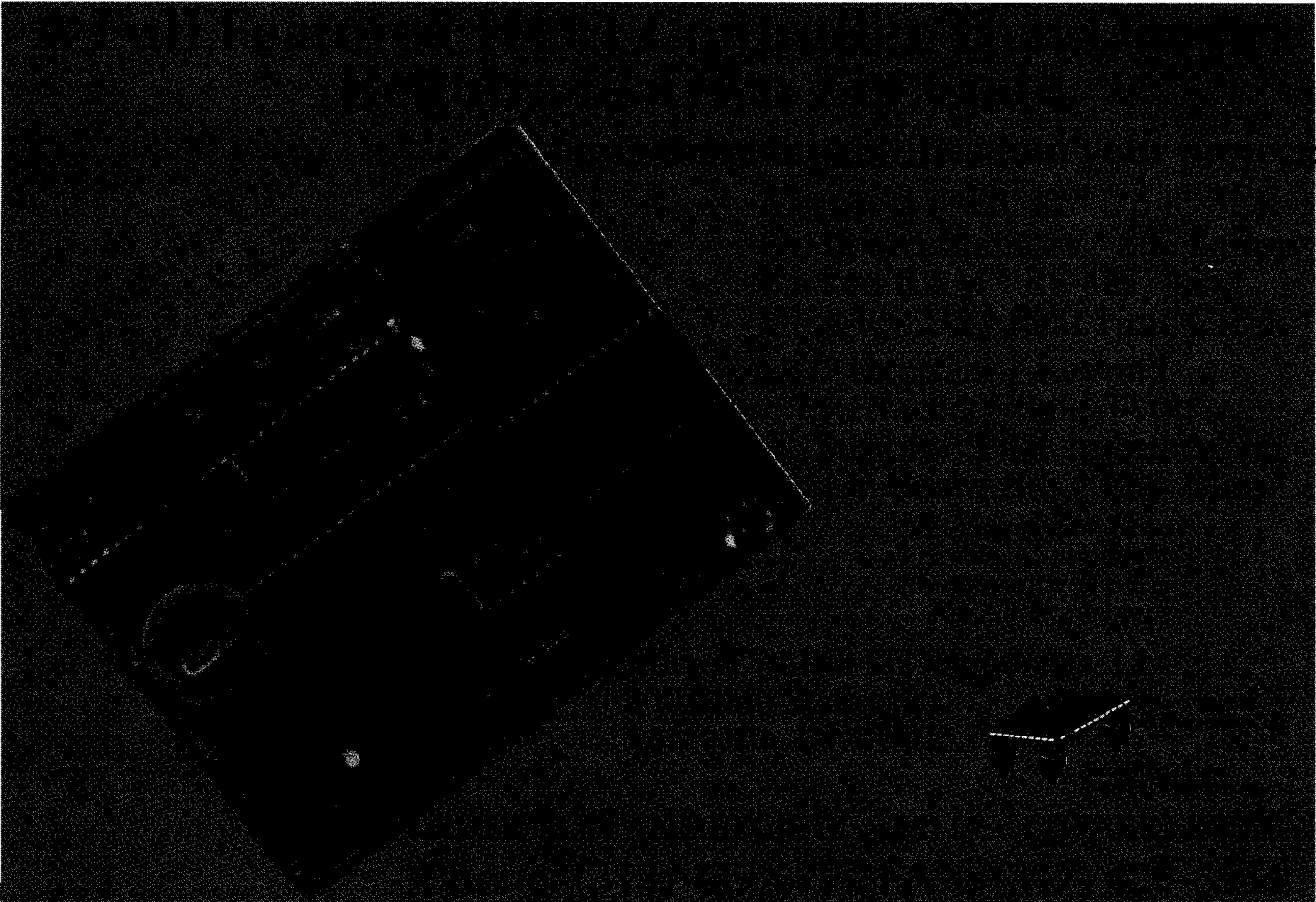
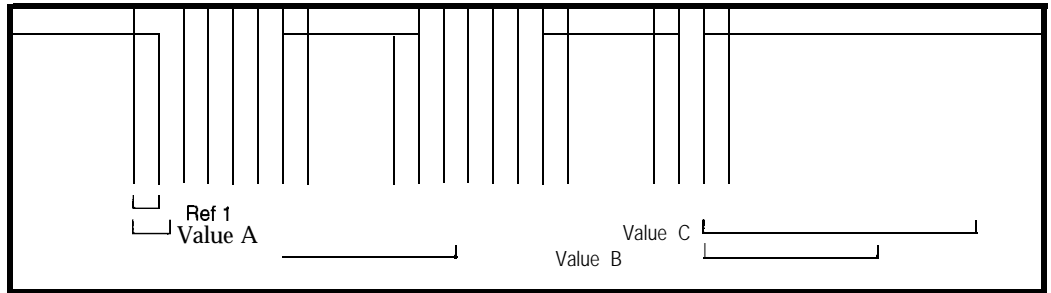


Figure C—Using three counters and a timer, it's possible to detect the presence of a ring signal, the cadence (for distinctive ringing), and the number of rings.



store-and-forward method is useful if you want to make the PBX as transparent as possible and yet wish to provide sophisticated features.

INCOMING CALLS

Of course, incoming calls must also be considered. You may want to provide configurable options in the computer's memory dictating which port to forward a call to when it arrives on a CO circuit.

For example, you might send a call to station 10 when it arrives on CO1, while another call goes to station 15 when it arrives on CO2. You may also have calls forward to another extension after the call has rung the first line for a selected number of rings.

You can easily add unique features by changing the supporting software (rather than hardware). For instance, you can decode the personalized ring from the telephone company and send the call to a different station for each of the personal rings. Thus, one phone line in a home could receive calls for up to three people and automatically route the call to the appropriate extension and answering machine.

To implement a feature like personalized ring decoding, you need to monitor the I/O port connected to the CO interface's optoisolator. When the optoisolator detects ringing from the central office, the associated I/O port toggles between 0 and 1 at the ringing frequency.

Figure 5 shows the digital output of the optoisolator. The algorithm goes something like:

- 1) when low, reset counter 1 to 0
- 2) wait for high
- 3) when high continuously, increment counter 1
- 4) if counter 1 exceeds preset value A, CO ringing is no longer present. Increment counter 2 (i.e., the number of bursts being sent).
- 5) if low, return to state 1

- 6) if signal stays high and counter 1 exceeds value B, increment counter 3 to count of the ringing cycles.
- 7) if signal stays high and counter exceeds value C, reset everything because ringing has not been present for awhile and the remote caller has abandoned the incoming call. Reset the station being rung to idle.

Value A is longer than one cycle of the frequency the central office rings. If the variable is larger than this value, central office is no longer signaling.

Value B is longer than the longest burst. If ringing is encountered again before the variable reaches this value, you're still in the ringing cycle but have encountered another burst.

Value C is longer than the longest complete ring, inclusive of all bursts. This value helps determine when an incoming call is abandoned.

Of course, when an incoming call is detected, you ring the destination station and let the CO interface remain idle. The caller receives ringback from the central office, which sees the call is not yet answered. When the destination station answers, engage the CO interface, making an audio connection between the CO and SLIC.

If you answer the CO interface and provide ringback from the PBX, you consume finite resources (i.e., call progress sender and the DTMF receiver). Then, if the destination station doesn't answer and the call is long distance, the caller is charged because the CO interface was activated and answered.

TIME TO HANG UP

The interface presented here improved conversation quality dramatically, making the Personal PBX a full-fledged switch rather than a super intercom.

These same interfaces can be massaged into a wide variety of systems and form factors. For instance, a board for a personal computer can provide a complete, low-cost PBX system for home and offices.

With a complete PBX, you can enhance the software for custom features such as toll restriction, remote access, and PC-based remote control of telephones.

Look for an article in April on Caller ID. I'll show you how to make a stand-alone caller ID decoder, interface Caller ID to a personal PBX, and take advantage of it in software.®

Richard Newman is an electrical engineer living in Dallas, TX. He designs specialized communications and industrial automation equipment either in partnership or on contract. He can be reached at ricardo@netcom.com.

SOFTWARE

A C source code example of a call-processing algorithm implementing selective/adaptive routing is available from the Circuit Cellar BBS and on Software on Disk for this issue. See the end of "ConnectTime" for downloading and ordering information.

SOURCE

PCB, 8 SLICs, 3 COs, source code, EPROM, and schematics
 Caden Development
 4335 Cedar Springs, Ste. 201
 Dallas, TX 75219
 (214) 522-8935

IRS

404 Very Useful
 405 Moderately Useful
 406 Not Useful

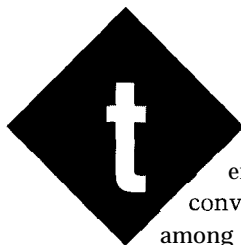
LPT:Analog!

FEATURE ARTICLE

A 12-bit A/D Converter Printer Port Adapter

Are you having problems getting collected data from instruments into your computer without a fancy interface? If so, LPT:Analog! may help. It interfaces analog-output instruments and most PCs.

David Prutchi



These days, high-end instruments converse fluently among each other and with PCs through GPIB, making it easy to acquire data from experimental setups. Unfortunately, however, collecting data from a setup without an integrated computer interface is usually difficult and full of pitfalls.

Using an A/D converter add-in card is often an impractical alternative. To start with, opening the computer enclosure every time the instrument needs to be connected to a different computer is a major inconvenience.

Portability is further complicated by the fact that most laptops and notebooks lack ISA slots for add-in cards. PCMCIA data acquisition cards do not help much either since very few desktop systems are fitted with suitable interfaces.

Moreover, most of the bells-and-whistles of full-featured data-acquisition cards are often unnecessary when interfacing to the processed analog output of an instrument.

The device of Photo 1 provides a simple and convenient interface between analog-output instruments and

most PCs on the market. Instead of connecting to the computer's expansion bus, it plugs into a parallel printer port, which it uses for serial I/O and as a power source for a one-channel ADC.

Despite the simplicity of this adapter's circuitry, it is capable of achieving very good performance. It has a maximum sampling rate of 75k samples per second, 12-bit resolution, and $\pm\frac{1}{2}$ LSB nonlinearity. With it costing a total of \$35 in parts, you'll find this instrument to be an affordable and versatile gadget.

IT COULDN'T BE SIMPLER

Not too long ago, designing a high-performance A/D converter would have been a major undertaking. Recently, however, mixed-mode IC technologies have evolved to the point where manufacturers are offering inexpensive well-behaved data-acquisition solutions in single miniature packages.

The MAX187 is one of Maxim's single-chip A/D converters, featuring a 12-bit, 8.5- μ s successive-approximation converter, a 1.5- μ s track-and-hold, on-chip clock, a precision 4.096-V reference, a high-speed three-wire serial interface, and single +5-V power supply requirement.

As shown in Figure 1, a MAX187 is at the heart of LPT:Analog!. Power for the MAX187 is supplied directly from an output line of the printer port or through a MAX756 step-up DC-DC converter. The other two output lines control the .SHDN (shutdown) pins of the ICs. The serial interface of the MAX187 requires only three digital lines and can directly connect to the printer port.

The complete circuit fits within a D-subminiature hood for the DB-25

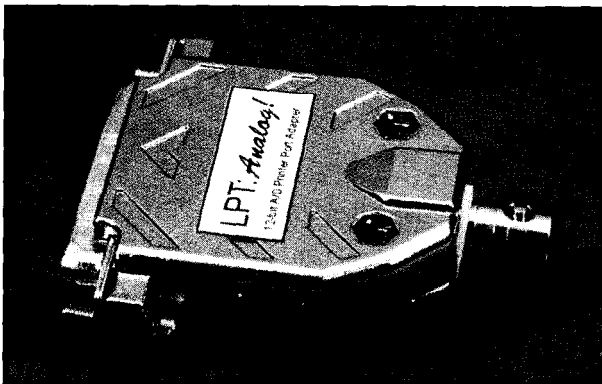
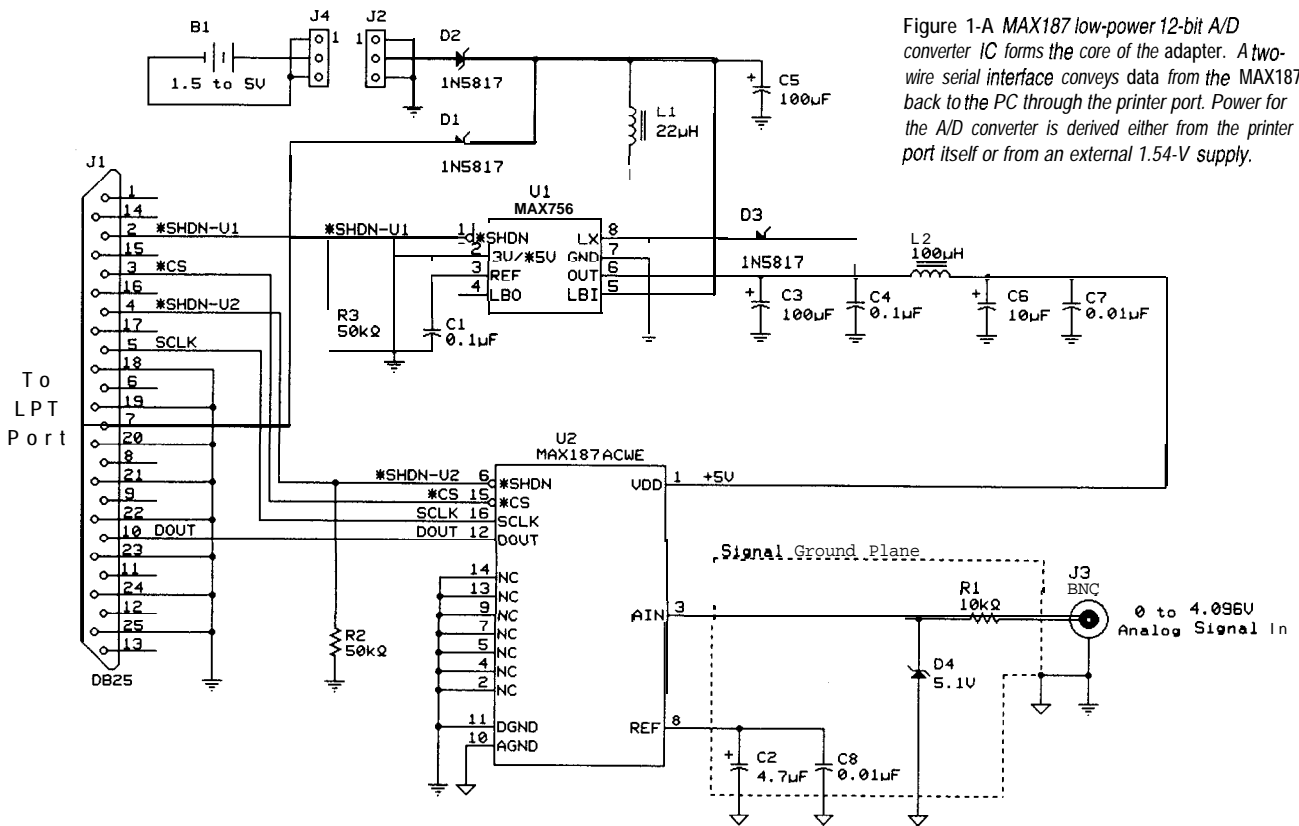


Photo 1—The LPT:Analog! adapter houses a 12-bit A/D converter which connects to a PC through the printer port. Power for the A/D converter can usually be derived from the printer port, but a side connector is available for an external power supply whenever necessary.

Figure 1-A MAX187 low-power 12-bit A/D converter IC forms the core of the adapter. A two-wire serial interface conveys data from the MAX187 back to the PC through the printer port. Power for the A/D converter is derived either from the printer port itself or from an external 1.5-V supply.



connector. A panel-mount female BNC connector extends beyond the hood through the cable opening, forming the analog signal input of LPT: *Analog!*

A 3-pin header is used for J2 and protrudes from a side of the hood through a notch lined by a small rubber grommet. A metallized-plastic hood reduces interference pickup by the sensitive input circuitry of the MAX187.

The analog signal to be measured is connected to the analog-input line AIN of the MAXI 87. Voltages of 0-4.096 V can be converted by the A/D

converter into distinct digital codes for every 1 mV of change.

The input signal, however, can go as high as 5.3 V or as low as -0.3 V without causing permanent IC damage. Since exposure to higher voltages may happen by accident, a 5.1 -V zener diode clamps excessive input voltages.

Current limiting is provided by a 10-kΩ resistor. This scheme measures signals from low-impedance sources only. This limitation occurs because a voltage drop across the resistance of the voltage source under measurement can be caused by current leakage at the zener V-I curve knee.

INTERFACING WITH THE PRINTER PORT

The MAX187's A/D conversion initiation and data-read operations are controlled by the *CS and SCLK lines. As shown in Figure 2, an A/D conversion is initiated by a falling-edge on the CS line. At this point, the track-and-hold holds the input voltage, and the successive-approximation process begins.

The start of conversion is acknowledged by the MAX187 changing the state of the DOUT line from high impedance to a low state. After an internally timed 8.5-µs conversion

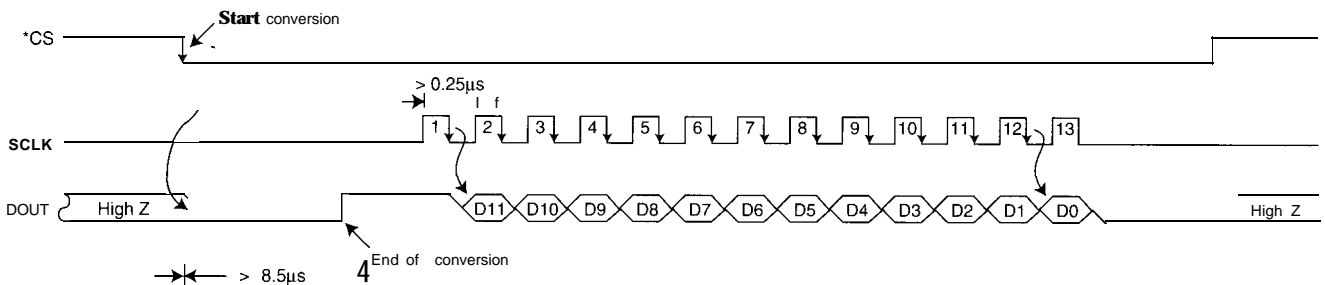


Figure 2—Data acquisition and serial protocol timing within the LPT:Analog!. An A/D conversion is initiated by a falling edge on the *CS line. After conversion, data is read out in serial format, shifted from the sequential-approximation register on each falling-edge transition of SCLK.

period, the end of conversion is signaled by the DOUT line going high.

Once conversion is complete, data can be obtained in serial format and shifted from the sequential-approximation register on each falling-edge transition of **SCLK**. Since there are **12** bits, a minimum of 13 falling-edge pulses are required to shift out the **A/D** converter's result.

Bits 1 and 3 of the LPT 8-bit output port (378h for LPT1:.) are toggled by software to implement the control portion of the MAX187 serial protocol. Bit 6 of the printer status port register (379h for LPT1:.) receives the serial data from the MAX187.

Stealing power from the printer port is not easily accomplished for every computer. The MAX187 requires a minimum supply of 4.75 V with a current of up to 2.5 mA. Not every printer port can supply enough current at the necessary voltage.

Typically, printer ports are specified for their ability to drive a single TTL load of the printer's input port and not by a standardized source impedance. The designer is then free to select between TTL, LS TTL, TTL-compatible CMOS, or other technologies to suit the overall design of the computer system.

For this reason, computers in the market have very different printer-port driving capabilities, and *LPT:Analog!* is designed to cope with these.

Computers capable of delivering the 4.75 V at 2.5-mA minimum supply requirements power the MAX187 directly by way of Schottky diodes D1 and D3. Other computers can step up an output line to +5 V using U1, which is capable of supplying at least 3 mA at 1.5 V.

Still, others, like laptops and notebooks, cannot supply the necessary power. In this case, an external battery pack or regulated power supply operates *LPT:Analog!* by way of D2. The supply mode is controlled by appropriately selecting the state of the *SHDN pins of U1 and U2. However +5-V power is derived, a pi filter formed by C3, C4, C6, C7, and L2 ensures a clean supply to the A/D converter.

In addition, you may notice that two separate ground planes, one analog

Listing 1-This sample program demonstrates data acquisition with *LPT:Analog!* The program runs under QuickBASIC. that the default LPT1:.

```
' Printer port locations
CONST prinop = &H378          ' Printer Output Port
CONST prinstat = &H379       ' Printer Status Port

' Define variables and control pin locations
CONST pow = 64, notcs = 2, notshdn = 4, sclk = 8
DIM vout AS SINGLE, clocknum AS INTEGER, dat AS INTEGER,
    powmode AS INTEGER
DIM power AS INTEGER

' Initialize
OUT prinop, 0                ' Clear printer port
CLS                          ' Clear screen

' Compensate for computer speed
' Primitive timing scheme implemented.
' Better resolution achieved through PC hardware timer
PRINT "Calculating loop timing . . ."
TIMER ON
ON TIMER(1) GOSUB delayt      ' Collect data for 1 s
FOR delay = 1 TO 2000000: NEXT delay
convspeed:
TIMER OFF
SCREEN 2
CLS
PSET (1, 10)
PRINT "Calculating conversion speed . . .";
TIMER ON
ON TIMER(10) GOSUB convt      ' Collect data for 10 s
FOR acq = 1 TO 1000000
    GOSUB acquire
        y = INT((4.096 vout) * 45) + 10
        LINE -(1, y)
        FOR delay = 0 TO 0: NEXT delay
    NEXT acq

' Acquisition and display control
start:
PRINT
PRINT "Please enter power supply mode:"
PRINT "<1> for DC-DC converter ON"
PRINT "<0> for DC-DC converter OFF"
INPUT powmode
IF powmode = 1 OR powmode = 0 THEN
    power = pow + powmode
ELSE
    BEEP
    PRINT "Valid choices are <0> or <1>. Reenter power-supply mode!"
    PRINT
    GOTO start
END IF
TIMER OFF
PRINT
PRINT "Please enter desired sampling frequency [S/s]"
fsamp:
INPUT fsamp
IF fsamp > (1 / convt) THEN
    ' check that desired sampling frequency doesn't exceed sampling
    ' frequency. Achieved by this computer running QuickBASIC.
    BEEP: PRINT "Maximum sampling frequency for this computer = ";
        1 / convt: " [S/s]"
    PRINT "Please reenter desired sampling frequency !"
    GOTO fsamp
END IF
delsamp = INT(((1 / fsamp) convt) / delayt)
```

(continued)

```

PRINT "Actual sampling frequency to be used = ";
      1 / (convt + (delsamp * delayt)); "[S/s]"
FOR delay = 0 TO INT(3 / delayt): NEXT delay ' pause for 3 s
SCREEN 2 ' CGA graphics mode 640x200
GOSUB acquire ' determine 1st display point
y = INT((4.096 vout) * 45) + 10
start1:
CLS
LOCATE 2, 2: PRINT "4.096V";
LOCATE 7, 2: PRINT "3.000V";
LOCATE 13, 2: PRINT "2.000V";
LOCATE 19, 2: PRINT "1.000V";
LOCATE 25, 2: PRINT "0.000V";
PSET (80, y)
FOR i = 100 TO 640
  GOSUB acquire
  y = INT((4.096 vout) * 45) + 10
  LINE -(i, y)
  IF INKEY$ <> "" THEN GOTO progend ' press any key to escape
  FOR delay = 0 TO delsamp: NEXT delay ' wait before next conv.
NEXT i
GOTO start1
progend:

' Leave program
OUT prinop, 0 ' power down LPT: Analog!
SCREEN 0 ' return to text-mode screen
END
acquire:

' Acquisition loop
OUT prinop, power + notshdn + notcs ' power up but keep CS' off
convert:
OUT prinop, power + notshdn ' convert by asserting CS'
FOR delay = 0 TO delcon: NEXT delay ' wait for at least 8.5 us
dat = 0 ' clear A/D accumulator
FOR clocknum = 11 TO 0 STEP -1 ' clock 12 bits serialy
  OUT prinop, power + notshdn + sclk ' clock pulse rising edge
  FOR delay = 0 TO delclk: NEXT delay ' wait at least 0.25 us
  OUT prinop, power + notshdn ' clock pulse falling edge
  dat = dat + (2 ^ clocknum) * (INP(prinstat) AND 64) / 64
  ' read bit and accumulate
NEXT clocknum ' next bit
OUT prinop, power + notshdn + sclk ' one more clock
FOR delay = 0 TO delclk: NEXT delay
OUT prinop, power + notshdn
OUT prinop, power + notshdn + notcs ' deassert CS'
vout = dat * 4.096 / ((2 ^ 12)) ' translate D/A data to V
RETURN
delayt:

' Calculation of delays
delayt = 1 / delay
PRINT "single delay loop = "; delayt; " seconds"
' Calculate number of iterations for conversion wait loop
IF delayt > .0000085 THEN
  delcon = 0
  ELSE
  delcon = INT(.0000085 / delayt) + 1
END IF
' Calculate number of iterations for clock pulse wait loop
IF delayt > 2.5E-07 THEN
  delclk = 0
  ELSE
  delclk = INT(2.5E-07 / delayt) + 1
END IF
GOTO convspeed

```

(continued)

and one digital, are shown in Figure 1. Ideally, the signal ground plane should be used as the reference for the analog-input signal.

This same ground plane should be constructed to shield the analog portions of the A/D converter (i.e., the input network and the voltage reference filtering and decoupling caps). The analog and digital ground planes should be connected at a single point, preferably directly to J1's ground pins.

If you use the 8-pin DIP version of the MAX187 instead of the 16-pin SO packaged version, join the planes at pin 5 of the IC. Make the connection to the ground pins of J1 at this point.

SOFTWARE

Listing 1 presents a sample program for driving the LPT:*Analog!*. The program flow starts by initializing the ports. Notice that use of the standard LPT1: is assumed. You may need to change the output and status port locations to suit your installation.

Sampling rate is regulated by inserting FOR/ NEXT loops to introduce delay between samples. The number of loops required to reach the correct delay is based on a calculation of the time for the computer to complete a data single acquisition and display operation as well as of the delay introduced by the addition of a FOR/NEXT loop.

The two timing parameters are obtained by measuring the number of loops that can be executed within 10 s. Obviously, a complete prototype of the acquisition and display loop must be included to increase the precision of the estimate.

The actual acquisition subroutine starts by powering up the A/D converter, but keeping *CS deasserted. Conversion is then initiated by asserting CS, and a wait of at least 8.5 μ s takes place before it attempts to read the conversion data.

After this delay, the A/D converter's accumulator variable is cleared, and the 12 bits are clocked in serially. The value of each bit is read from the status port and is multiplied by the decimal value of its binary position before being accumulated. The routine ensures that each clock pulse is at least 0.25 μ s wide.

Finally, one more clock pulse is inserted to reset the A/D converter. The 'CS line is deasserted, and A/D data is translated to volts.

This program is intended only as an example of implementing the serial protocol required to collect data from the LPT:*Analog!* through the printer port. Major enhancements could be made to it.

First of all, QuickBASIC imposes a significant limit on data-acquisition speed. Even running on a 90-MHz Pentium PC, the sampling rate is limited to about 1.4 kHz. True compilers such as C++ increase the effective sampling rate to approach the 75-kHz maximum sampling rate supported by the MAX187.

Another improvement can be made by eliminating the delay loops that control timing. Instead, you can control the acquisition process from interrupts generated by high-resolution hardware timing [1,2].

You could also add threshold-triggered acquisition, a nice virtual-instrument panel, and fancier graphics. If

Listing 1-continued

```
convt:
' Display A/D sampling information
SCREEN 0
convt = 10 / acq
PRINT "Conversion time = "; convt; " seconds"
PRINT "Maximum sampling frequency = 'I; 1 / convt; " samples/s"
GOTO start
```

you write such a piece of software, please share it with the rest of us through the Circuit Cellar BBS!

Last, if you intend to use LPT:*Analog!* for data logging, you may want to consider powering down U1 and U2 by setting their "SHDN lines low between acquisitions. Since current drain drops to only 20 μ A while data is not actively sampled, this modification is especially useful when external batteries power the device in long-term logging applications.

NOT ALL SIGNALS FIT 0-5 V

Input signals rarely fit exactly within LPT:*Analog!*'s 0-4.096-V range.

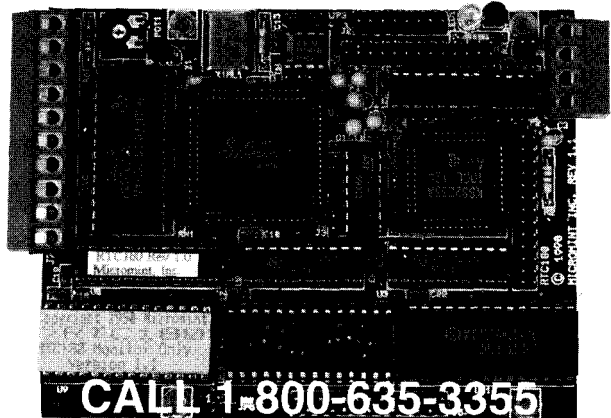
Signals of smaller amplitude than the full range waste resolution, while signals outside the full range end up being clipped by the protection circuitry to the limits of the range.

However, measuring a signal which spans between 0 V and a value larger than 4.096 V is easily accomplished. A resistive voltage divider such as that of Figure 3a scales a large unipolar signal to the desired range.

As shown in Figure 3b, a small unipolar signal requires only an op-amp-based amplifier to take advantage of the A/D converter's full resolution. Signals riding on a median different than the A/D converter's midpoint

**STOP
LOOK
LISTEN** !

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



CALL 1-800-635-3355



MICROMINT, INC.

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

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

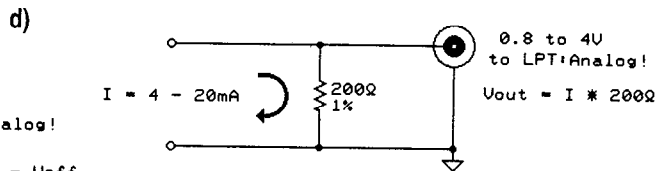
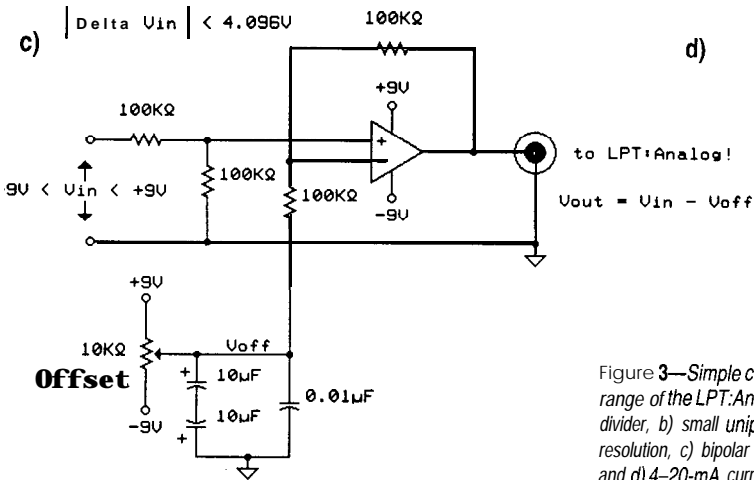
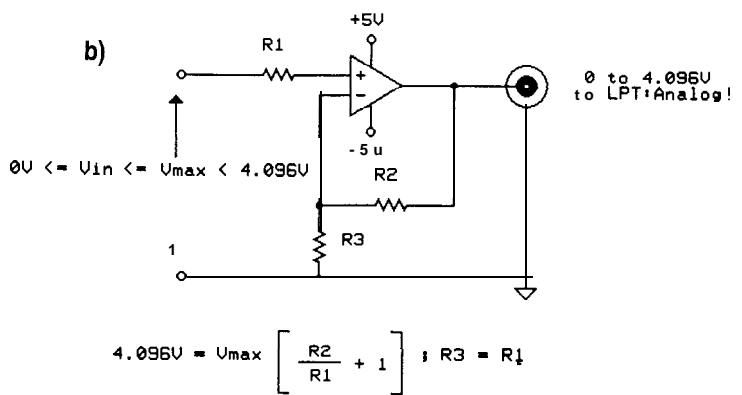
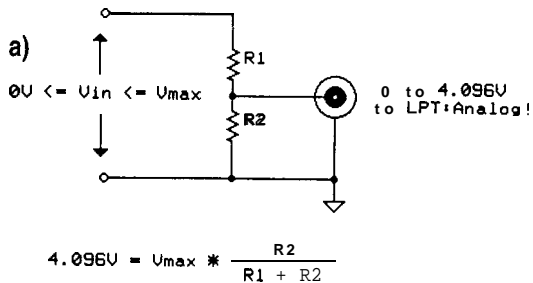
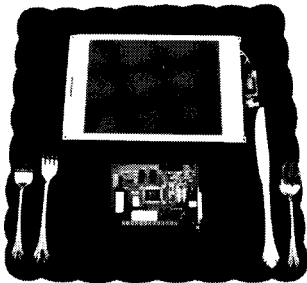


Figure 3—Simple circuits can be used to scale different signals to fit within the 0-4.096-V range of the LPT:Analog! a) Large unipolar signals can be attenuated by a voltage divider, b) small unipolar signals can be amplified to make use of the A/D converter's full resolution, c) bipolar signals can be converted into unipolar signals by introducing offset, and d) 4-20-mA current loop signals can be read through a resistive shunt.

Flat Panels Served Here



House Specialty - Monochrome LCD Kit \$199

NEW LOWER PRICES!

VGA LCD Controllers for PC ISA Bus

Earth LCD/M Monochrome LCD Controller \$99*
EarthVision/ISA Color LCD Controller \$249*

*When purchased with LCD

Display Kits Include LCD, Controller, & Backlite

Monochrome 9.4" \$199
Color Single Scan 8.2" \$399
Color Dual Scan 9.4" \$649
Color 9.4 Active Matrix \$895

Displays Only

Prices start at:

Mono VGA LCD \$49
Color VGA Single Scan \$99
Color VGA Dual Scan \$299
Color VGA Active Matrix \$599



"The Flat Panel Solutions Company"

P.O. Box 7089 • Laguna Niguel • California • 92607

h: (714) 448-9368 • Fax: (714) 448-9316 • <http://www.flat-panel.com>

Programmable Controllers Just Got Easier!

Smaller than a
credit card

Develop reliable
multi-tasking systems
in hours, not weeks,
with the new Micro G™
EasyStart Kit. Beginners

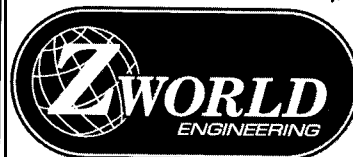
and experts alike will love our
simplified version of C. The kit includes:

- ◆ Micro G controller with 32K RAM, 128K flash EPROM, 14 bidirectional digital I/Os, and one analog input.
- ◆ Prototyping board with switches, LEDs, beeper and low-pass filter
- ◆ Dynamic C EasyStart™ our integrated development environment (editor, compiler, and source-level debugger) for Windows
- ◆ Schematics, reference manual, power supply and cables

It's everything you need to start serious development. **Only \$279.**

Call our AutoFAX 916.753.0618 from your FAX. Request data sheet #35.

"a Hulk Hogan
of capabilities"
Byte, Oct.95



1724 Picasso Ave.

Davis, CA 95616

916.757.3737

916.753.5141 FAX

(2.048 V) can be offset appropriately by using the circuit of Figure 3c.

Current measurements can be obtained by using a suitable shunt. For example, as shown in Figure 3d, the popular 4–20-mA current loop, used to convey information from many industrial instruments and sensors, can be converted to a voltage by using a metal-film 200- Ω ±1% resistor shunt across the input terminals of LPT:Analog!.

Since the 4–20-mA current is translated into the 0.8-4-V range, some measurement resolution ends up being wasted. If the full 12-bit resolution is desired, you may use a 255- Ω ±1% resistor instead. An op-amp should then be used to introduce a 1.02-V offset to the measurement.

REMEMBER NYQUIST

Not only the amplitude range of a signal is important, but also its frequency range must be considered for properly acquiring data.

When sampling a continuous signal, information may be lost because no data is available between sample points. As the sampling rate increases, a larger portion of the information is made available.

According to Nyquist's theorem, to correctly sample a waveform, the sampling rate must be at least twice that of the highest-frequency component of the waveform. Ignoring this rule results in aliasing—a process in which signal components of frequency higher than half the sampling rate appear as components with a frequency equal to the difference between the actual frequency of the component and the sampling rate.

Because aliased components cannot be distinguished from real signals after sampling, aliasing is not a minor source of error. For this reason, the proper selection of acquisition rate is imperative in acquiring meaningful data [3].

If noise or high-frequency components beyond the frequency band of interest are nevertheless present, an antialiasing filter with high roll-off should be placed between the signal source and the analog input of the A/D converter.

IN CONCLUSION

Unlike ordinary data-acquisition setups, LPT:Analog! offers a great degree of flexibility and portability. Virtually all PCs come equipped with a parallel printer port. Reconnecting this A/D converter to the computer typically does not require delving into the enclosure. Moreover, since the use of the printer port is standardized, the same software runs on any PC without reconfiguration.

Sporting a sampling rate of 75k samples per second, 12-bit resolution and $\pm\frac{1}{2}$ LSB nonlinearity, the performance of LPT:Analog! often surpasses the effective resolution of many high-quality signal displays, oscilloscopes, and chart recorders.

As long as the signal to be converted is correctly scaled and sampled, this A/D adapter should have you acquiring high-quality data from your older analog-only instruments in no time at all. \square

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

REFERENCES

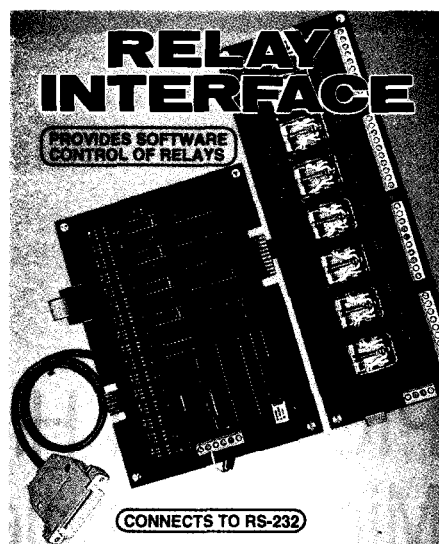
1. D.P. Schulze, "A PC Stopwatch," *INK* 19, 22–23, 1991.
2. B. Ackerman, "High-Resolution Timing on a PC," *INK* 24, 46–49, 1992.
3. D. Prutchi, "Spectral Analysis: FFTs and Beyond," *INK* 52, 1994, 2039, 1994.

SOURCE

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

IRS

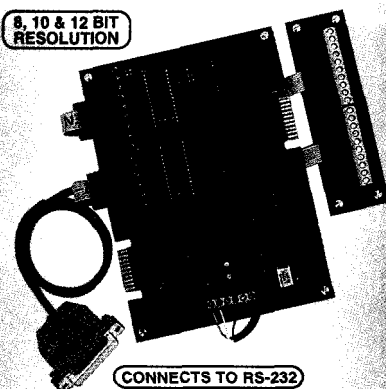
407 Very Useful
408 Moderately Useful
409 Not Useful



RELAY INTERFACE
PROVIDES SOFTWARE CONTROL OF RELAYS
CONNECTS TO RS-232
AR-16 RELAY INTERFACE (16 channel).....\$ 89.95
Two 6 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.
AR-2 RELAY INTERFACE (2 relays, 10 amp)....\$ 44.95
RD-8 REED RELAY CARD (6 relays, 10 VA).....\$ 49.95
RH-8 RELAY CARD (10 amp SPDT, 277 VAC)....\$ 69.95

ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232
ADC-16 A/D CONVERTER* (16 channel/8 bit)....\$ 99.95
ADC-8G A/D CONVERTER* (6 channel/10 bit)....\$124.95
Input voltage, amperage, pressure, energy usage, light, joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately). Includes Data Acquisition software for Windows 95 or 3.1
ADC-8E TEMPERATURE INTERFACE* (8 ch.)...\$ 139.95
Includes term. block & 8 temp. sensors (-40' to 148' F).
STA-8 DIGITAL INTERFACE* (8 channel).....\$ 99.95
Input on/off status of relays, switches, HVAC equipment, security devices, keypads, and other devices.
PS-4 PORT SELECTOR (4 channels RS-422)....\$ 79.95
Converts an RS-232 port into 4 selectable RS-422 ports.
CO-422 (RS-232 to RS-422 converter).....\$ 35.95

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

* FULL TECHNICAL SUPPORT...provided over the telephone by our staff. Technical reference & disk including test software & programming examples in TurboBasic, GW Basic, Visual Basic, Visual C++, Turbo C, Assembly and others are provided.

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

* CONNECTS TO RS-232, RS-422 or RS-485...use with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 18,200 baud).

FREE INFORMATION PACKET...use our 800 number. Fax or E-mail to order, or visit our Internet on-line catalog. URL: <http://www.cen.net/~johnhart/eect1.html>
Technical Support (614) 464-4470

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

Internet E-mail: eect@ibm.net
International & Domestic FAX: (614) 464-4444
Use for information, technical support & orders

ELECTRONIC ENERGY CONTROL, INC.
360 South Fifth Street, Suite 604
Columbus, Ohio 43215-5481

FEATURE ARTICLE

Ron Cates

PIC17C44 High-End Microcontroller Architectural Overview

Here's an MCU that provides increased performance and an improved migration path. Discussion focuses on the instruction set, register file, bus structure, and variable word-length instructions.



Systems are becoming more complex as market demands change rapidly.

Suppliers need to respond to market demand just to maintain customers.

Fundamental in this quest for improvement is the need for a streamlined time to market. For companies to remain competitive, the cost of production, compatibility between lines of processors, and modular software development is critical.

Microchip's PIC 17C44 8-bit field-programmable microcontroller unit (MCU) provides features that increase performance and make migration easier. The architecture blends well-known characteristics particularly well-suited to MCU implementation.

Many systems now use MCUs, so features can be implemented in software and modified more easily. However, the choice of MCUs should be made carefully since it has a significant impact on many factors in the product-development process.

In this article, I'll discuss several characteristics of the PIC17C44 and their impact on software. Topics include instruction set, register file, bus structure, and variable-word-length instructions. I'll end with a brief discussion of memory technology and its impact on MCUs.

PIC16/17 FAMILY CONCEPT

The PIC16/17 architecture combines several well-known architectural features that result in a flexible, high-performance system at an attractive price-to-performance ratio. The PIC16/17 uses a Harvard-type implementation that separates the instruction and data paths into two buses, a simple RISC-like instruction set, instruction pipelining, and a register-based memory architecture.

The dual-bus structure of Harvard architecture increases the memory bandwidth available to the CPU, which in turn enhances performance. The unique combination of these architectural features delivers superior performance, compact and efficient code, and easy migration across the PIC16/17 product family.

The PIC16/17 architecture supports a family concept that is an architectural level above most MCUs, one that maintains a consistent migration path for users over a long period of time.

Before beginning to implement hardware, the PIC16/17 architects decided that the instruction set limited the long-term migration path in a computer architecture. An instruction set should be scalable as technology advances. Otherwise, it limits hardware implementation and performance in the future.

The PIC16/17 architecture is therefore based on the concept of a Virtual Instruction Set (VIS) as shown in Figure 1. By allowing the instruction size to vary across various families, an optimal price-to-performance point is achieved.

While the instruction set in a particular member of the family is of fixed size, the types and lengths of instructions vary in scalable fashion between implementations. For those instruction types common to all families, each instruction on one processor is essentially a superset or subset on another family.

For example, Figure 1 shows the **ADDWF** (add W register to register file F) for all three families. As indicated, the opcode is the same for all three devices. The only difference is in the number of registers supported within the instruction.

Since the program and data memory are a large portion of the die area, it's important to maximize their use. If the instruction word was fixed at 16 bits, base-line devices would simply have four zeros added to the program word with little benefit. For data-manipulation operations, small implementations (small number of registers and program memory) aren't penalized by carrying extra register-addressing capability.

Branches are handled in much the same way. Each member's branch length varies to match the program memory-size capability, so bits aren't wasted within the word. Address tags can consume a large portion of the program and must be minimized.

PIC17C44

The PIC17C44, shown in Photo 1, is the newest member of the high-end family. The PIC 17C44 provides:

- 8 KB x 16 OTP program memory
- 454 bytes of data RAM
- three timers
- two capture channels
- two PWM channels
- 8 x 8 unsigned-multiply instruction (single-clock execution)
- a high-speed universal synchronous/asynchronous receiver and transmitter (USART).

Photo 1—Microchip's PIC 17644 8-bit OTP microcontroller features the world's fastest speed and 8K x 16 program memory.

The combination of large OTP memory and high-speed execution with C compilers and fuzzy logic is ideal for today's market demands.

The PIC 17C44 has two different buses. One bus handles instructions and the other, data. Figure 2 shows a simple block diagram of it.

Overall system performance improves dramatically with the dual-bus approach. It also makes the virtual instruction architecture very straightforward.

The PIC16/17 architecture also uses a two-stage pipeline that improves performance in a cost-effective manner. Since all instructions are 16 bits long and fetched in a single memory

access, an instruction is available for execution on every cycle. All 58 instructions, except for taken branches and table read/write, execute in one cycle (system clock divided by four).

REGISTERS

The register implementation is a major architectural feature of this chip. All system RAM locations reside in the register file and are available for every CPU cycle, so performance increases substantially.

No compiler is needed to manage a

small register file effectively by optimizing register-allocation algorithms. All registers are available for data manipulation on every cycle.

Another effect of the register file is code compaction. Much of the code in MCU applications moves data to and from main memory for calculations and manipulation. Since most MCUs have a very small working register set, the problem is greater than on larger microprocessors. In general, PIC16/17 code requires fewer instructions than other MCUs—sometimes as much as 50% less than comparable 8-bit MCUs.

As well, the PIC 17C44 offers a **MOVFP** and **MOVPF** instruction which transfers data in a single cycle, resulting in even greater efficiency.

INSTRUCTIONS AND MEMORY

The PIC 17C44 includes an 8 x 8 unsigned multiply instruction that executes in a single cycle. At 25 MHz, the PIC 17C44 performs a multiply operation in only 160 ns, much faster than most 16-bit MCUs. The high-speed multiply support makes the device well-suited for computationally intensive, fuzzy-logic, or low-end DSP applications. Table 1 shows some

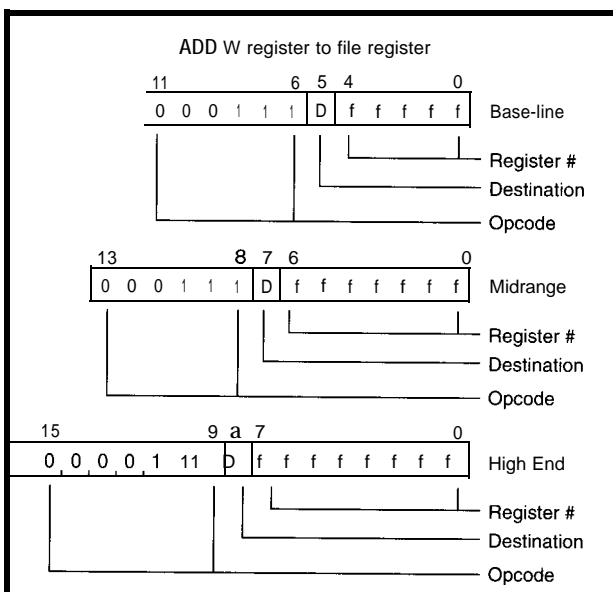
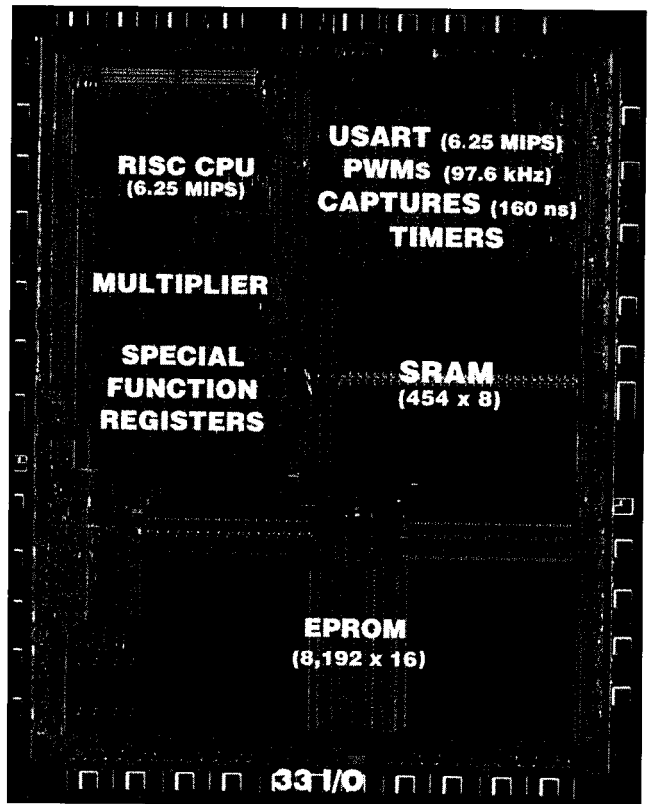


Figure 1—The PIC16/17 Family Virtual Instruction Architecture is based on the ability to change instruction length across families. The main determinants of instruction length are branch length and the number of registers directly addressed.

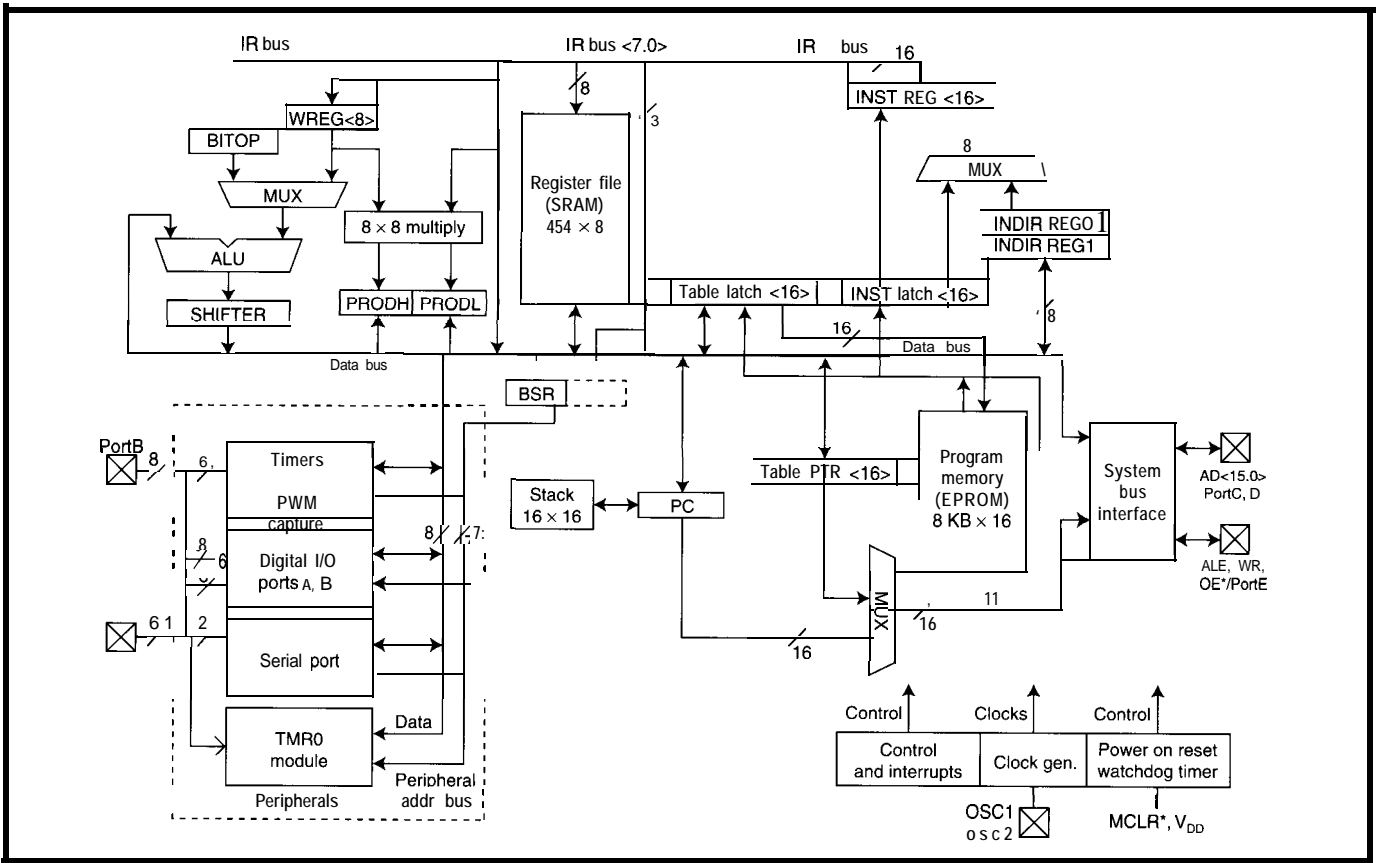


Figure 2—This PIC17C44 block diagram illustrates the peripheral capabilities. Key blocks are the single-cycle 8 x 8 multiplier and high-precision PWMs.

RTOS TCP/IP

US Software's Embedded System Suite is an exceptional RTOS, Networking and file system that includes all the tools and utilities you need for your design.

USNET® Networking

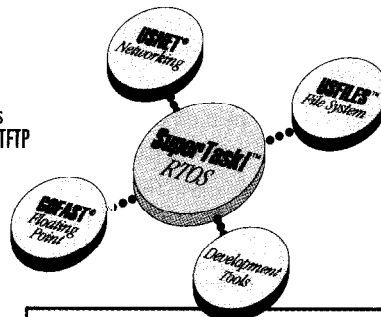
- Complete TCP/IP protocol suite.
- RTOS & Processor independent
- Compact - only 25K of space
- Romable and Reentrant
- Fast and user Configurable
- 100 megabit support
- Includes Client/Servers TELNET, BOTP, FTP, TFTP
- Protocols: TCP, UDP, IP, ICMP, PPP, SLIP, ARP, RARP
- Full Source Provided

SuperTask™ RTOS

- Compact and Fast
- Romable & Reentrant
- User Configurable
- Full Featured - Over 70 System Calls
- FastTaskWch
- Optional File System
- Low Interrupt Latency
- Full Source Provided

GOFAST® Floating Point

- RTOS independent
- Romable & Reentrant - fast
- "Link & Go" Usage
- Replaces C Compiler Library
- Designed for Embedded Use



Processors Supported
68HC11, 68HC16, 683XX, 680X0, Z180, 8051, 80196, 80x86, PM386, 80960, SPARC, MIPS, SH, PowerPC, ARM, and others.



14215 N.W. Science Park Drive • Portland, Oregon 91119
800-356-7097 • TEL: 503641.8446
FAX: 503-644-2413 • EMAIL: ussw@netcom.com

benchmarks for other arithmetic functions.

Like other PIC17C4x family members, the PIC 17C44 supports three modes of operation: microprocessor, extended MCU, and MCU. In microprocessor mode, all program memory resides in off-chip external space. The external memory space supports up to 64K words of program ROM or RAM.

In extended MCU mode, the first 8K locations of program memory are in on-chip program EPROM. The remaining 48K words are in external combination of nonvolatile and volatile memory. The MCU mode uses only on-chip memory resources. Figure 3 shows the memory map for the different operational modes.

The PIC 17C44 supports direct and indirect addressing of data memory. For direct addressing, the memory address is contained in the instruction word. For indirect addressing, the memory address is held in special function register FSRO or FSR1.

The special function registers are configured to autoincrement, auto-decrement, or remain static, allowing

flexible access to arrays or ring buffers. The program counter is a 16-bit read/write register which supports computed jumps and table lookups.

The PIC 17C44 uses a simple RISC-like instruction set that contains 58 instructions. All instructions are 16 bits and are therefore fetched in a single memory cycle. The PIC 17C44 uses a simple two-stage pipeline, so most instructions execute in a single cycle.

Taken branches and table read/write operations require two cycles for execution. The instruction set supports extensive bit and byte manipulation to further increase performance in computationally intense applications.

COMMUNICATIONS

The PIC 17C44 offers several high-performance peripherals for interface to devices such as motors, analog sensors, and computer controllers such as PCs or embedded DOS systems. Many systems are migrating to networked total digital electronic control.

For high-speed communication, the PIC 17C44 provides the Serial Communications Interface (SCI). The SCI supports full-duplex asynchronous and half-duplex synchronous operation. Maximum data transfer rates are 250 kbps asynchronous and 6.25 Mbps for synchronous communications.

The synchronous mode supports either master or slave operation. The module includes a dedicated baud-rate generator that uses the CPU oscillator as a clock source. A programmable clock-divider chain enables most common bit rates to be synthesized without separate clock sources.

TIMER

For interface to various devices such as motors, the PIC 17C44 contains four timer modules with vectored interrupt support. For enhanced timebase support, two input captures and two PWM outputs are provided.

TMRO is a simple 16-bit overflow counter with a programmable prescaler that may be set for values from 1 to 256. The TMRO clock source is the CPU oscillator or an external clock. With an external clock, the module

8th Annual
Circuit Cellar
DESIGN CONTEST

CALL for ENTRIES
ENTRY DEADLINE
AUGUST 2, 1996

The time is now for you to start thinking about your entry in the 8th Annual Circuit Cellar Design Contest. Entering is easy, just contact Rose at:
Circuit Cellar Design Contest
4 Park Street
Vernon, CT 06066
Tel: (860) 875-2199 or Fax: (860) 872-2204

You'll be sent an official Entry Form and a complete set of rules. All entries must be received by August 2, 1996. You may enter as many projects as you wish, but each entry must be accompanied by a separate Entry Form.

Sponsored in part by Dataman Programmers, Inc.

Routine	Program Words	CPU Cycles	Time (μs) @ 25 MHz
8 x 8 unsigned multiply with 16-bit result	1	1	0.160
16 x 16 signed multiply with 32-bit result	36	36	5.76
16 x 16 unsigned multiply with 32-bit result	24	24	3.84
8 x 8 signed multiply with 16-bit result	6	6	0.960

Table 1—The PIC17C44's math routines show the mathematic precision of the hardware multiplier can be extended.

increments on either the rising or falling edge. The maximum external clock frequency supported is 50 MHz.

TMR1 is an 8-bit timer/counter with an 8-bit period register and interrupt capability on overflow events. The clock source can be internal or provided externally on the RB4/TCLK12 pin.

TMR2 is identical to TMR1 and shares the same clock source. TMR1 and TMR2 can be concatenated to form a 16-bit timer with TMR2 as the most-significant byte and TMR1 the least significant.

TMR3 is a 16-bit timer/counter with a 16-bit period register. The clock source can be the system oscillator or external via the RB5/TCLK3 pin. The

timer resources are general purpose, but there are dedicated resources associated with some timers. TMR1 and TMR2 are used as timebases for the two PWM outputs, and TMR3 is the timebase for the two input captures.

For increased reliability, the PIC-17C44 has a Watchdog Timer (WDT) that must be cleared on a regular interval when enabled (an EPROM fuse provides the enable/disable function). The WDT provides a recovery mechanism from software malfunction. Unless the WDT is cleared before it times out, a reset is generated.

The WDT has a dedicated on-chip RC oscillator for the clock source. No external components are required and the oscillator continues to run during

sleep even though the system clock has been stopped. If the external oscillator fails, the device resets when the WDT times out.

A reset event returns all I/O pins to the input state, so the designer can disable all control signals with the proper selection of pull-up or pull-down resistors. System reliability and safety are greatly improved by the WDT with a separate RC oscillator.

PACKAGING

The PIC 17C44 is housed in 40-lead PDIP and ceramic windowed packages for through-hole applications. Surface-mount applications are supported in 44-lead Plastic Leaded Chip Carrier (PLCC) and Thin Quad Flatpack (TQFP). The TQFP is ideal for space-constrained applications such as PCMCIA cards.

Up to 33 digital I/O pins are available for control of external devices. All I/O pins can sink up to 35 mA and source up to 20 mA. Two pins, RA2 and RA3, provide up to 60-mA sink capability for increased drive applica-

Designing Something?

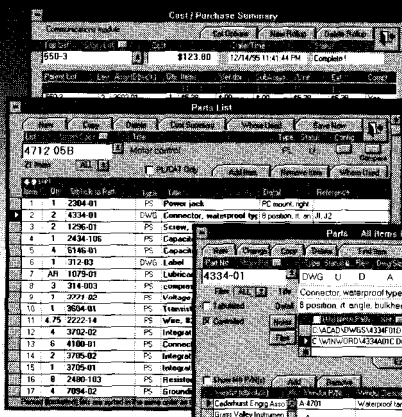
For Engineers
Techs and
Designers.

This Parts List Software will
help you stay organized.

Easily create
and manage
multi-level
parts lists for
products in
development.

Keep track of:

- Part Specs
- Drawings
- Suppliers
- Product and Parts Costs
- Engineering Stock



V1.5

Parts Vendor's™

for independent and departmental projects. Use alone or in a workstation.

Version SE: \$99 + shpg Call 800-280-5176

EXTended: \$299 + shpg TrilogY 916-273-1985

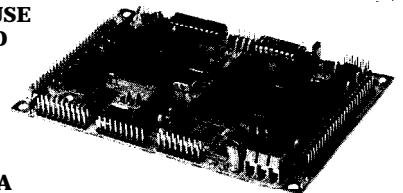
Requires 486, 8 meg min. ram, Windows™ P.O. Box 2270, Grass Valley, CA 95945 fax.916-477-9106

\$129.⁹⁵

QTY. 1

THAT'S RIGHT! \$129.95 FOR A FULL FEATURED SINGLE BOARD COMPUTER FROM THE COMPANY THAT'S BEEN BUILDING SBC'S SINCE 1985. THIS BOARD

COMES READY TO USE AND FULLY LOADED FEATURING THE 68HC11F1 PROCESSOR. ADD A KEYPAD AND AN LCD DISPLAY AND YOU HAVE A STAND ALONE CONTROLLER WITH ANALOG AND DIGITAL I/O. OTHER FEATURES INCLUDE:



- * 8 HIGH-DRIVE OUTS & 12 PROG. DIGITAL I/O LINES
- * 8 CHANNELS OF FAST 8 BIT A/D & OPT 4. CHAN. D/A
- * TIMER/COUNTERS WITH PWM
- * UP TO 2 RS232/485 SERIAL PORTS
- * BACKLIT CAPABLE LCD INTERFACE
- * OPTIONAL 16 KEY KEYPAD & INTERFACE
- * 160K OF MEMORY SPACE, 64K INCLUDED
- * 68HC 11 ASSEMBLER & MONITOR INCL., BASIC OPT.

1985-1995
10
YEAR
ANNIVERSARY

ENAC, inc.

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

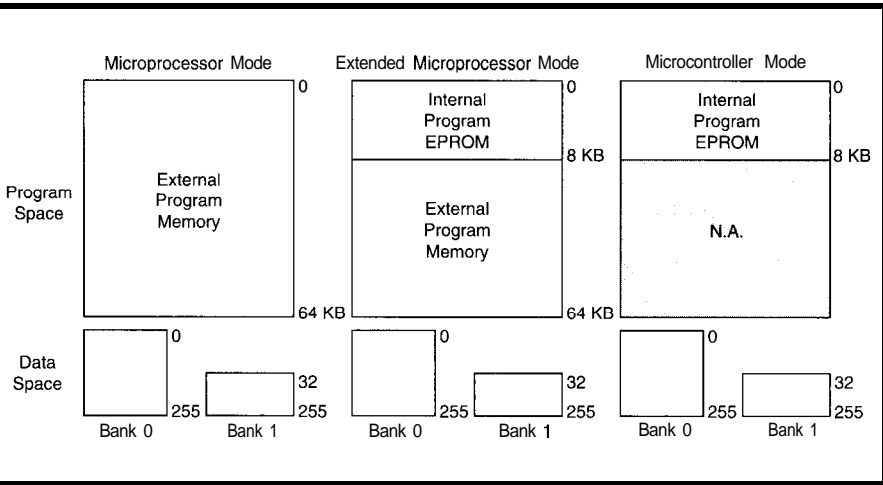


Figure 3—The PIC17C44 operating modes offer the user a highly flexible solution. Modes include extended microcontroller mode to support both internal-program and external-program or data memory.

tions. Any I/O pin is capable of driving LEDs directly for enhanced user interfaces.

CONCLUSIONS

The PIC16/17 architecture has become successful because of its attractive cost-to-performance ratio. Because it uses the concept of a virtual instruction to tailor the CPU to sys-

tern resources, the PIC16/17 offers 8-bit MCU users a superior migration path.

The PIC17C44 provides a large OTP program memory and high-speed computation capability for the high-end family. In addition, the architecture ensures compatibility into the future as process technologies unfold and user requirements increase.

With a superior migration path, the PIC16/17 offers ease-of-use programmability across all three families today and in future versions as well.

Ron Cates is strategic marketing manager for Microchip Technology. He has spent more than 20 years in marketing and engineering functions at Microchip, VLSI Technology, Motorola, and General Dynamics. His technical achievements include four U.S. patents, and he has authored more than 75 technical articles. Ron may be reached at (602) 786-7609.

CONTACT

Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224
(602) 786-7200
Fax: (602) 786-7277

I R S

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

B.G.MICRO, INC. P.O. Box 280298
Dallas, Tx 75228
Orders Only 1-800-276-2206
Tech Support 214-271-9834
Fax 214-271-2462
Local Orders 214-271-5546

Our Internet address is BGMICRO@IX.NETCOM.COM

SWITCHING POWER SUPPLIES

Switching Power Supply #1
A full 220 Watts of awesome computer power - Completely enclosed w/fan - Manufactured by Level -JL & CSA Approved.
Input: 110/220 Vac
Output:
+5 Volts @ 23 Amps
-5 Volts @ 500 MAmps
+12 Volts @ 8.5 Amps
-12 Volts @ 500 MAmps
Measures: 8-3/8"x5-7/8"
x5-7/8" **\$17.99**

Switching Power Supply #2
Compaq model SMP-80HB - Completely enclosed w/fan - UL/CSA approved - A very compact 73 Watt power house - Input: 110/220 Vac - Output: +5 Volt @ 10 Amps +12 Volts @ 1.5 Amps -5 Volt @ .3 Amps -12 Volts @ .3 Amps - Measures 3-3/4"x2-3/4"x6-3/4" **\$9.99**

CATALOG PLUG IN TRANSFORMERS

Plug in transformer #1. 15 vDC/600 mA. Makes a great battery charger. CSA and UL approved. Works well with 7812 regulator. Barrel connector. Specially priced to move. #1 **\$2.79**

TERMS: (Unless specified elsewhere) Add \$4.00 postage, we pay balance on Orders Under \$50. Orders over \$50.00 add \$5 for insurance. No C.O.D. Texas Residents add 8.14% Tax. 90 Day Money Back Guarantee on all items. All items subject to prior sale. Prices subject to change without notice. Foreign Orders U.S. Funds Only. We cannot ship to Mexico or Puerto Rico. Canada, add \$7.50 minimum shipping and handling. Countries other than Canada, add \$15.00 minimum shipping and handling.

Call and place your order today! 1-800-276-2206

386 SBC \$83 OEM (1K) PRICE

INCLUDES:
- 5 SER (8250 USART)
- 3 PAR (32 BITS MAX)
- 32K RAM, EXP 64M
- STANDARD PC BUS
- LCD, KBD PORT
- BATT. BACK. RTC
- IRQ0-15 (6259 X2)
- 0237 DMA 6253 TMR
- BUILT-IN LED DISP.
- UP TO 8 MEG ROM
- CMOS NVRAM

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

\$95 SINGLE PIECE PRICE

UNIVERSAL PROGRAMMER
- DOES 6 MEG EPROMS
- CMOS, EE, FLASH, NVRAM
- EASIER TO USE THAN MOST
- POWERFUL SCRIPT ABILITY
• MICROCONT. ADAPTERS
• PLCC. MINI-DIP ADAPTERS
- SUPER FAST ALGORITHMS

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

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

MVS BOX 850 MERRIMACK, NH (508) 792 9507

EMBEDDED PC

F E B R U A R Y 1 9 9 6

Nouveau PC 42 59 **PC/104 Quarter**

High-Speed Network Technology 47

R. E. Billings

Chassis and Enclosures

A PC/104 Packaging Overview

Dave Cox & Paul Olsen

Unscrewing the Inscrutable 54

Ed Nisley

65 **Applied PCs**

Small Displays for Embedded Systems

Russ Reiss



PC/ 104 DEVELOPMENT TOOL

Embedding PC/104 cards into systems is simplified by using the **TCDEV** hardware and software package from Saelig. Compatible with industry-standard PC/104 cards from over 80 manufacturers, TCDEV provides a standalone environment that mounts PC/104 cards. It accommodates a floppy (or optional mini hard disk) and includes ready-made connectors for mouse, serial, parallel, and VGA ports.

TCDEV becomes a prototyping host PC to run a compiler and debugger when attached to a target processor board, screen, and keyboard. Code can be tested on the target hardware, and development can take place on the identical setup used in the final system. There is, therefore, no need for a host PC, serial link, and complex cable assemblies.

A variety of utility software comes with the development system. And, with the appropriate hardware, RAM disk, ROM disk, flash memory, and

PCMCIA technologies are fully supported.

When development is complete and the right PC/104 board inserted, TCDEV becomes a high-performance PC with more potential than most desktop machines.

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

#510

FIFTH-GENERATION SBC

WinSystems has announced the LBC-5x86 Single-Board Computer. Based on the 100-MHz superpipelined 5x86 CPU from Cyrix, the unit offers fifth-generation processing performance while maintaining full PC compatibility and 20% power-supply savings compared to a '486-DX4/100 in an equal system configuration. The unit is well-suited for applications such as automated industrial equipment, SCADA, medical instrumentation, transportation, communications, and test/measurement.

The board measures 5.75" x 8.00" and is currently available with an operating speed of 100 MHz. Integrating the basic AT-peripheral complement, the board includes the system controller, two 8-channel interrupt controllers, real-time clock, three counter/timers, seven DMA channels, keyboard controller, and speaker port. A precision power-fail reset circuit, activity LED, and watchdog timer makes the unit ideal for remote or unattended operation. A broad complement of onboard I/O includes two serial channels, each with RS-232, and optional RS-422/485 levels, Centronics parallel I/O port, floppy-disk controller, IDE hard-disk interface, and 16-bit PC/104 expansion connector.

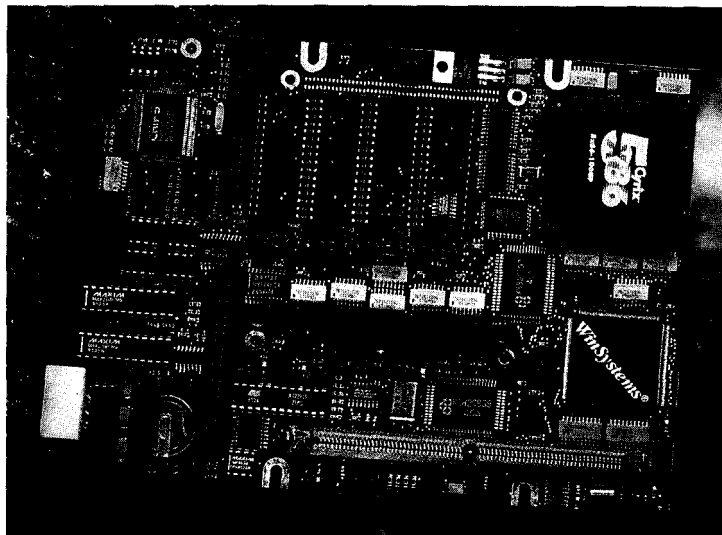
A 72-pin SIMM socket accepts up to 64 MB of system DRAM. Four onboard 32-bit EPROM sockets support up to 2 MB of flash and battery-backed SRAM or up to 4 MB of EPROM. Multiple jumper-selectable memory maps permit these boards to support bootable ROM or RAM disk. An installable device driver for use with MS-DOS and ROM-DOS is provided.

The LBC-5x86-100-0M (no memory installed) sells for \$995.

WinSystems

715 Stadium Dr. • Arlington, TX 76011-6225
(817) 274-7553 • Fax: (817) **548-** 1358

#511



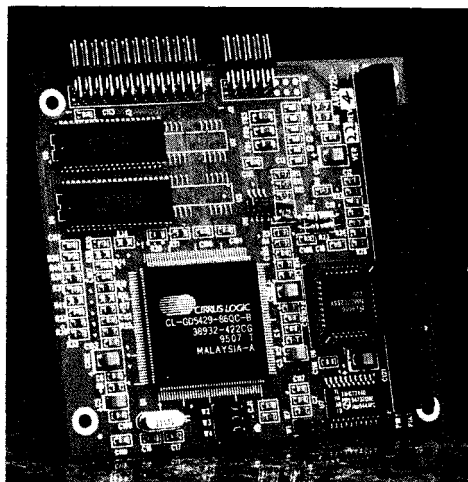
Nouveau PC

edited by Harv Weiner

PC/104 GRAPHICS ACCELERATOR

Ampro announces a new PC/104 display-controller module with hardware graphics acceleration that displays over 16 million colors on VGA CRTs. The **MiniModule/SVG-II** display interface is Fully software compatible with five popular video standards (VESA, VGA, EGA, CGA, and MDA), complies with the PC/104 V2 specification for compact (3.6" x 3.8") embedded PC modules, and is jumper configurable to operate with either 8- or 16-bit PC/104-compatible CPUs.

Resolutions of up to 1024 x 768 with 256 colors and 640 x 480 pixels with 16M colors ("true color" VGA) are supported with 1 MB of display memory. The module directly drives analog monitors. Its VESA feature connector interface enables direct connection of EL display panels and supports a number of specialized video input and output interfaces. The module's built-in GUI accelerator engine offers



exceptional performance under Windows, Windows 95, and other GUI environments.

The MiniModule/SVG-II is supported by virtually all operating systems, drivers, utilities, and applications for both text and graphics because of its register- and BIOS-level compatibility with nearly all standard PC video controllers. SVGA software development is greatly simplified by the module's VESA-enhanced video BIOS, which supports the high-resolution display modes beyond normal VGA.

The module requires a single +5-V supply for operation and is rated for an extended operating temperature range of 0-70°C with wider temperature ranges available on special order.

The MiniModule/SVG-II is priced at \$187 in OEM quantities.

Ampro Computers, Inc.
990 Almanor Ave. • Sunnyvale, CA 94086
(408) 522-2100 • Fax: (408) 720-1305

#512

386EX PROCESSOR ON PC/104 FORMAT

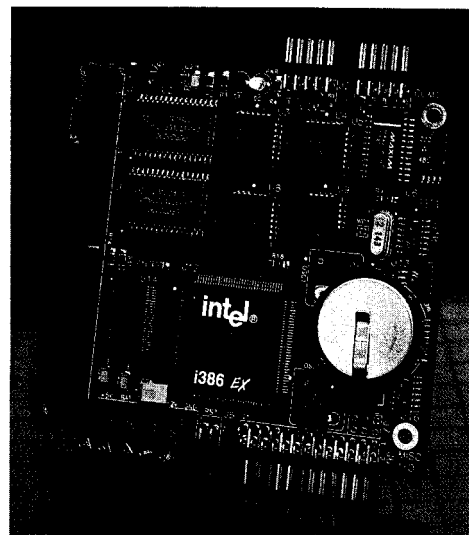
The **SBC1386EX** computer from Micro/sys uses Intel's '386EX processor and is optimized for embedded applications. Because the board integrates a number of devices, it offers embedded '386 performance at a low price. Unlike some embedded '386s, it includes entire '386 real- and protected-mode architectures. Systems can be created with self-contained code, a real-time operating system, DOS, or even Windows. The '386EX is based on a 32-bit static core processor featuring several power-management modes.

This 3.6" x 3.8" PC/104-form-factor computer operates at 25 MHz. It has 1 MB RAM and 5 12 KB preinstalled flash memory, with firmware resident in the write-protected boot block of the flash memory. Two PC-compatible COM ports, 13 TTL I/O lines, a real-time clock, three timers, a watchdog timer, interrupt, and DMA controllers are also onboard. Application programs are loaded through a COM port and programmed into the onboard flash memory for diskless, embedded operation.

Firmware preinstalled in the onboard flash memory includes a system BIOS, a royalty-free operating system that executes standard EXE files, a download manager, and a debug manager that works with Borland's Turbo Debugger. A **DK1386 Development Kit** is supplied free of charge with the first SBC1386EX ordered. The kit comes with a firmware license, download cable, and complete documentation.

The SBC1386EX has a full 16-bit PC/104 interface that accepts additional I/O cards for system expansion. Available PC/104 modules offer analog and digital I/O, serial ports, modems, network interfaces, disk and PCMCIA adapters as well as LCD, keypad, VGA, and touchscreen-operator interfaces. An enclosure for a stack of up to four PC/104 modules is also offered for packaging SBC1386EX systems.

The SBC1386EX sells for \$495 in single quantity.



Micro/sys

3447 Ocean View Blvd. • Glendale, CA 91208 • (818) 244-4600 • Fax: (818) 244-4246

#513

Nouveau NPC

EMBEDDED SYSTEM BIOS

General Software has announced **Embedded BIOS 3.0**, its third-generation BIOS designed for embedded systems and consumer electronics based on the 'x86 CPU architectures. Version 3 now includes integrated support for Flash File system, PCMCIA socket services, in-place flash BIOS updating, and advanced power management. It also comes with its own **miniDOS** and **COMMAND.COM** that runs from ROM for consumer applications that don't need a full desktop DOS.

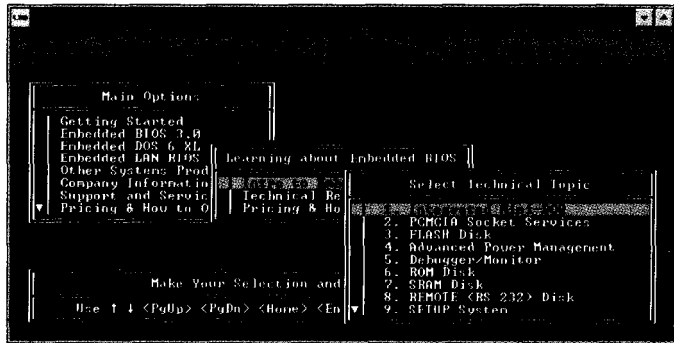
With MiniDOS integrated into the BIOS, there is no need to purchase and license a separate DOS to launch applications. MiniDOS uses only enough RAM for scratch space and runs directly from ROM, saving valuable RAM space for application code. Despite its small footprint, MiniDOS is a full-featured DOS that has all of the power of a desktop DOS. It supports **CON F I G . S Y S**, **AUTOEXEC.BAT**, **COMMAND.COM**, device drivers, and standard utilities.

Embedded BIOS 3.0 was designed for embedded and volume **commodity electronic** products, but it is **highly compatible** with desktop BIOS standards. Embedded BIOS runs all major desktop operating systems including MS-DOS, Windows 3.x and 95, OS/2 Warp, NetWare 386, and General Software's own high-end real-time Embedded DOS **6-XL**.

Embedded BIOS comes as a full-source adaptation kit with over 250 source-level options and over 100 binary-level options that can be configured with its binary configuration program. In addition, Embedded BIOS can be coupled with add-on personality modules to enable support for Intel, AMD, and other 'x86-based families of embedded architectures.

General Software, Inc.

320 108th Ave. NE, Ste. 400 • Bellevue, WA 98004 • (206) 454-5755 • Fax: (206) 454-5744
E-mail: general@gensoft.wa.com



#514



LOW-VOLTAGE EMBEDDED MICROCONTROLLER

AMD introduces low-voltage and industrial-temperature versions of its popular 16-bit **Am 186EM** and **Am 188EM** microcontrollers for embedded applications. The **Am186EMLV** and **Am188EMLV** enable designers to reduce the size, power consumption, voltage requirements, and cost of embedded systems, while

increasing performance over **80C186/188-based** designs.

The highly integrated microcontrollers maintain software and peripheral set compatibility with 80C 186 standards while operating at 3.3 V. An innovative bus design enables the **Am 186EMLV** to achieve 3.3 VAX MIPS while using inexpensive **1 1 0-ns** memory. Additional features integrated into the microcontrollers include 2 serial ports, 32 programmable I/O lines, 2 additional interrupt channels, **glueless** interface to memory (including PSRAM), enhanced chip-select support, and a **16-bit** reset configuration latch.

The **Am186EMtV** and **Am188EMLV** are available in **20-** and **25-MHz** versions optimized to meet the needs of most common embedded applications such as disk drives, hand-held terminals, fax machines, terminals, printers, telephones, modems, and industrial control. The microcontrollers are recurrently available in **100-pin** TQFP and PQFP packages and are \$8.86 in volume.

Industrial temperature versions offer the same features with an expanded temperature tolerance of **-40°C** to **+85°C** for outdoor and industrial environments. The **Am 186EM-I** and **Am 188EM-I**, available in **100-pin** PQFP packages, start at \$9.67 in volume.

Advanced Micro Devices, Inc.

One AMD Place • P.O. Box 3451
Sunnyvale, CA 940883453 • (408) 732-2400
<http://www.amd.com/>

#515

Nouveau NPC

DIGITAL OUTPUT CARD

The **CIO-DO48H** is a unique ISA-bus compatible board that contains 48 lines of dedicated digital output. This architecture eliminates the resets and glitching associated with standard programmable digital I/O cards on power-up and in certain applications.

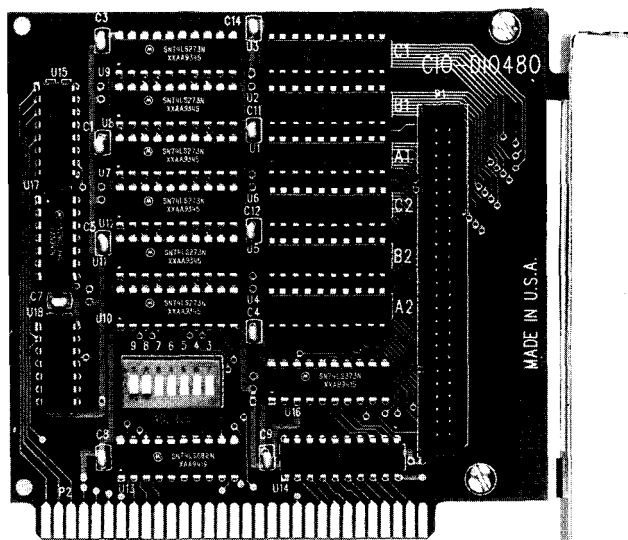
For example, when switching inductive loads through relays, programmable I/O cards are subject to resets from transient voltages and initialize all ports as input. The CIO-DO48H always initializes in output mode. Although the card requires no programming to initialize the output registers, it has a similar 82C55 register map. This added feature lets software written for 82C55-based I/O cards run without modification.

The CIO-DO48H is a high-drive device capable of sourcing 15 mA and sinking 64 mA. Solid-state relay modules, some relays, and LEDs may be activated directly from the card. High-drive dedicated digital output boards are also available with 96 and 192 lines of control.

The CIO-DO48H sells for \$99.

Computer Boards, Inc.
125 High St. • Mansfield, MA 02048
(508) 261-1123 • Fax: (508) 261-1094

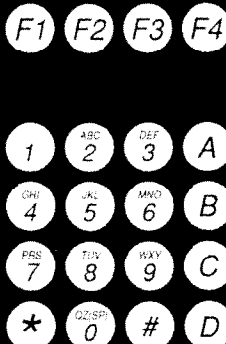
#516



Nouveau PC

VIEW-PORT™

AVAILABLE
MARCH 15



VIEW-PORT™ LOW-COST LCD TERMINAL

View-Port is Micromint's solution for information and control systems that need a low cost networked display terminal. View-Port combines a backlit 4x20 LCD display with a keypad in an attractive yet rugged package. Advanced features include automatic scrolling, input message buffering, wide supply voltage, and high speed operation. View-Port also provides a software controlled, 2 A-rated contact closure output for external annunciator lights, alarm bells, and so on.

FEATURES

- Backlit 4x20 LCD display
- Small size; 5" x 7" x 1.5"
- F-4 keypad and four function buttons
- Low power 9-15 V operation
- RS-232C or RS-485 at 300-115k bps
- Uses ANSI control codes
- Simple ASCII command protocol
- Automatic scrolling
- Software controlled contact closure output
- 16 character ID allows hundreds per network
- HCS II LCD Link function compatible
- Wall mount or handheld use

CALL 1-800-635-3355
TO ORDER

\$249
COMPLETE

MICROMINT, INC.

4 Park Street, Vernon, CT 06066 • (860) 871-6170 • Fax (860) 872-2204
Europe (44) 0288 658122 • Canada: (514) 336-9426 • Australia: (3) 467 7194

R.E. Billings

High-Speed Network Technology

Traffic jams used to happen only on the way to and from work. However, with the vast increase in network traffic, data congestion is becoming a problem at work also. Billings offers a fix with new high-speed networking.

With the rapid increase of interest in local area networks (LANs), client/server computing is quickly becoming the backbone of data processing systems. As LANs expand and data processing tasks become more complex, these networks become congested, leading to poor performance and more complicated customer installations.

Today's applications require LANs with high-bandwidth capabilities. Databases are becoming larger and more sophisticated, and greater numbers of users now access them. More importantly, the industry has made a mass migration toward applications involving high-resolution color graphics.

In these environments, it's desirable to store executable files in central data servers (file servers), rather than on the local disk drives of work-

stations. These programs must be constantly maintained and updated to ensure compatibility with new peripherals and to take advantage of the latest revisions. This approach is especially advantageous in large organizations, where it's impractical to update every workstation because of the technical labor required.

However, when a substantial number of users simultaneously load program files over the network, bandwidth quickly becomes a serious problem. Even networks with fewer than 100 workstations can become unusably sluggish when users actively load and execute programs for the Windows environment over the network.

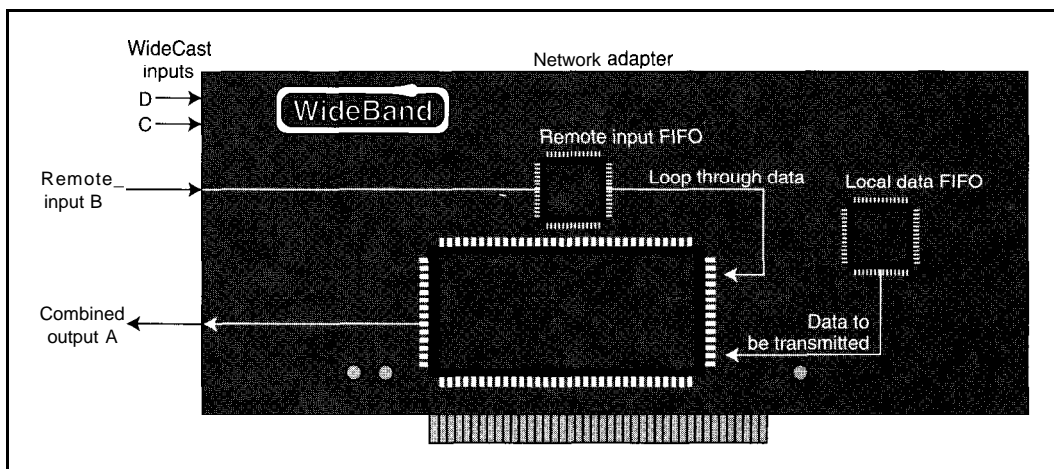


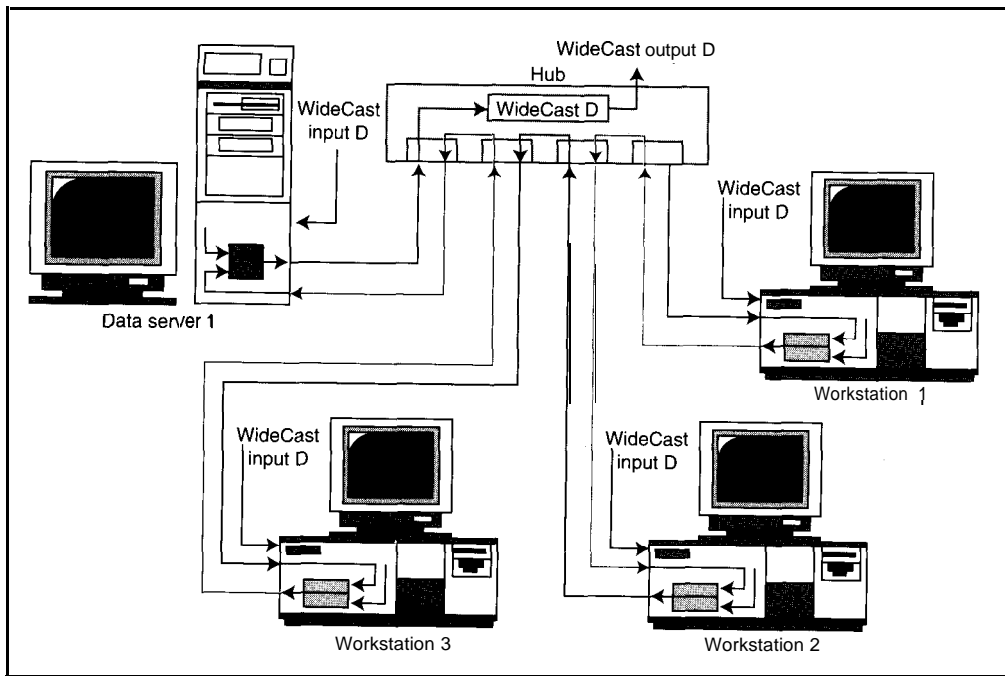
Figure 1: The WideBand network adapter uses all four pairs of a UTP-5 cable. Three pairs function as inputs, and one pair is an output.

Figure 2: In a single-hub, shared-bandwidth network configuration, the Wide-Band chained method of transmitting data provides two-way network communications with no data collisions.

To alleviate these problems, many install more data servers and divide large LANs into smaller LANs, connected by routers or bridges. While this approach greatly improves performance, it is costly and creates delays and complications when users need to share information over a wide area.

The industry is responding to these problems with a diversity of new and innovative products. Ethernet adapters with data rates of up to 100 Mbps are commercially available and quickly becoming affordable. Another approach, switching-hub technology, dedicates a portion of the LAN to a single or a small group of users. Many Token Ring LANs have also met the challenge of finding ways to increase performance and are operating at 16 Mbps.

Most networking managers anticipate the introduction of ATM [Asynchronous



Transfer Mode) as the solution to their networking bandwidth problems. Although most analysts see ATM as the wave of the future, its emergence has been slower than predicted.

As Doug van Kirk reports in InfoWorld, "predicting ATM's future isn't easy. Just describing it can be tricky because ATM doesn't neatly fit the layered models com-

mon to existing networks, and the specification itself does not encompass such things as speed and protocols.

"ATM is a sophisticated switched networking system that hosts an active application at each end. Although it breaks data into 53-byte 'cells,' ATM is not a packet-switched or router network architecture. In fact, for every stream of data sent, ATM

creates a virtual circuit among two or more points."

Van Kirk believes that on-line services, newspapers, and cable-television providers will use ATM to unify voice and data transmissions. In his opinion, it is the pipe these industries need to deliver large amounts of information to a desktop or set-top box.

However, as he points out, before this can happen, users need faster PCs, ATM-aware applications, and lower prices. [1]

Existing networks have achieved a degree of

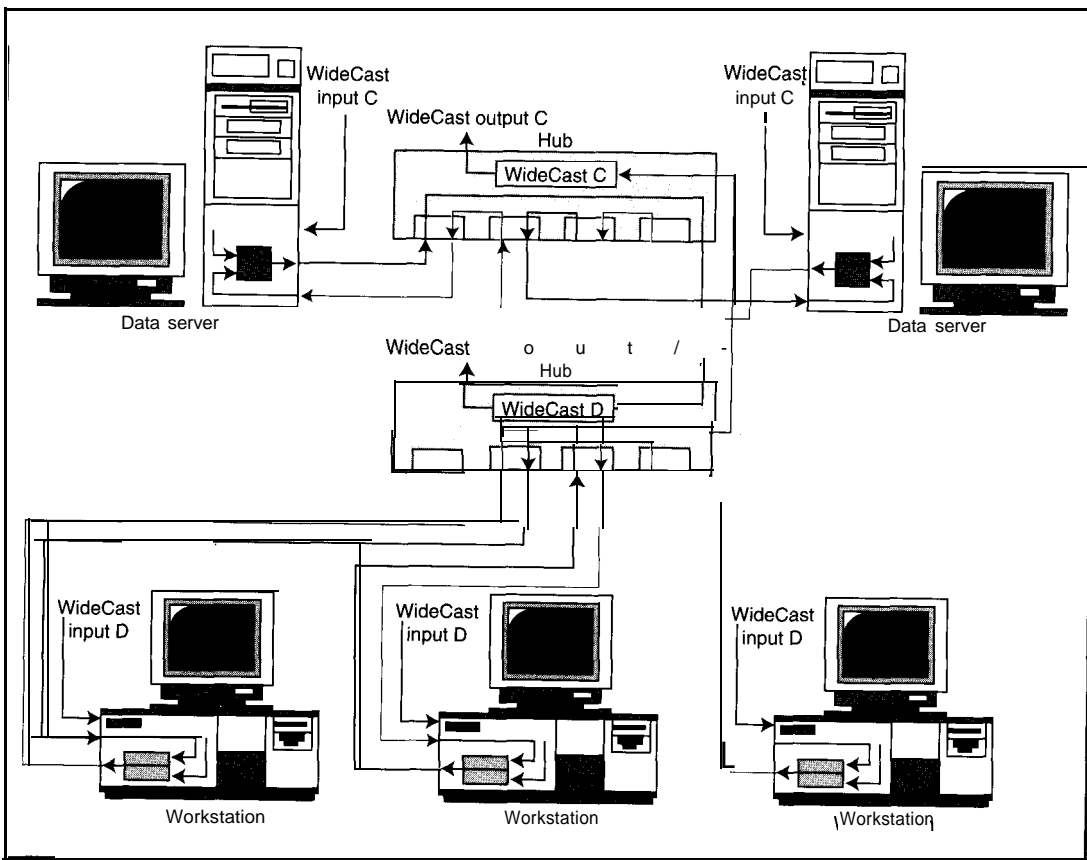


Figure 3: A dual-hub, full-bandwidth WideBand network is another possible configuration. By separating the transmissions of the workstations from those of the servers, available bandwidth is doubled.

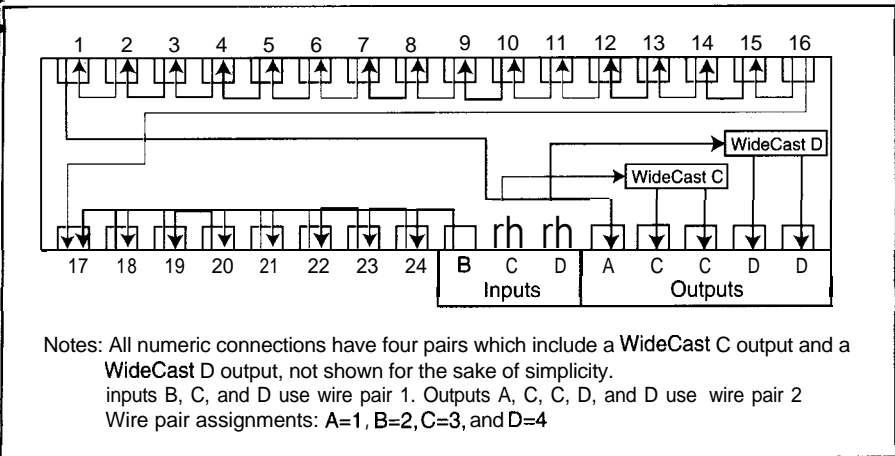


Figure 4: The WideBand hub provides a simple method of connecting computers into a LAN. Hubs can be cascaded as needed:

interoperability through a foundation in the seven-layer OSI model. With its dedicated point-to-point connections, ATM is a clear deviation from current technologies.

To implement the ATM approach, application software must be modified to become ATM aware. In some cases, the topology differences between ATM and today's networking schemes requires logical decisions which customized physical-layer interfaces and drivers can't provide.

The WideBand networking method uses the asynchronous-transfer approach of ATM in a proprietary implementation designed to be more compatible with existing networking models. You can install WideBand networking adapters in existing client/server or peer-to-peer installations simply with an interface at the physical level.

WideBand's interoperability results from a topology which offers communications from every computer on the network to all other computers on the network, including both clients and servers. The high-speed capabilities of ATM combine with the advantages of Ethernet to achieve low cost, incremental expandability, and versatility.

THE WIDEBAND SOLUTION

To understand the topology of a WideBand network, let's first consider the data connections and flow path of a network adapter. As Figure 1 shows, three of the WideBand cable pairs are inputs to the network adapter, while a single pair is an output. Each of these connections has a specific and designated purpose.

Cable pair 1 is the combined output A of the WideBand Network Adapter. Data packets that are input to the network adapter through remote input B are combined with

locally transmitted data and then sent to the hub via the combined output.

Ethernet synchronizes data transmissions from one computer to others on the LAN through a technique of data collision detection and recovery. In contention-type networks, data collisions cause a considerable percentage of network bandwidth to be lost when the network is heavily used.

This technology, however, is a contentionless protocol. Data collisions are

prevented by a loop-through approach which is accomplished in the network adapter or hub. Data packets from other computers enter the adapter through remote input B (see Figure 1).

These incoming packets are stored in the remote-input FIFO. The FIFO has been sized with enough depth to enable the temporary storage of the largest packet supported by the network. Local data to be transmitted is loaded into a separate FIFO device through the interface bus with the computer. A microprocessor or state machine synchronizes the packets to be transmitted.

At the beginning of the reception of a remote packet, the state machine detects the change of the FIFO empty flag, signaling the arrival of an incoming packet. On detection, the state machine immediately starts transmitting the incoming packet via combined output A.

If the local machine creates a packet for transmission and the empty flag of the remote-input FIFO indicates that no remote packet is being received, the state machine transmits the local packet via output A.

HALF-SIZE 5x86/486
Highly integrated, versatile and powerful SBC with
onboard Ethernet & VGA

VIPER 806
HALF-SIZE 5x86/
486DX2/DX4 SBC

TEKNOR designs and manufactures the world's most advanced and reliable industrial SBC and systems. Call TEKNOR today at 1-800-387-4222 for detailed product specs.

KEY FEATURES

- 5x86/486DX2/DX4 μ P
- Flat panel/CRT SVGA support
- 1 Mbyte video DRAM
- Feature Connector for video overlay
- 10Base-T / 10Base-2 Ethernet Controller
- Plug & Play Compatibility
- 4 Mbyte Bootable Flash Disk
- 128 Mbytes of DRAM
- Local bus IDE hard disk
- Flash BIOS
- RTC with battery backup
- PCI/104 compatible
- Watchdog timer
- Power fail/low battery detector
- Extensive power management features
- ISA passive backplane

• CANADIAN HEADQUARTERS: 616 Cure Boivin, Boisbriand, PQ J7G2A7
Tel: (514) 437-5682 ~ 1-800-354-4113 ~ Fax: (514) 437-8053

• U.S. HEADQUARTERS: 7900 Glades Road, Boca Raton, FL 33434
Tel: (407) 883-6191 ~ 1-600-387-4222 ~ Fax: (407) 883-6690

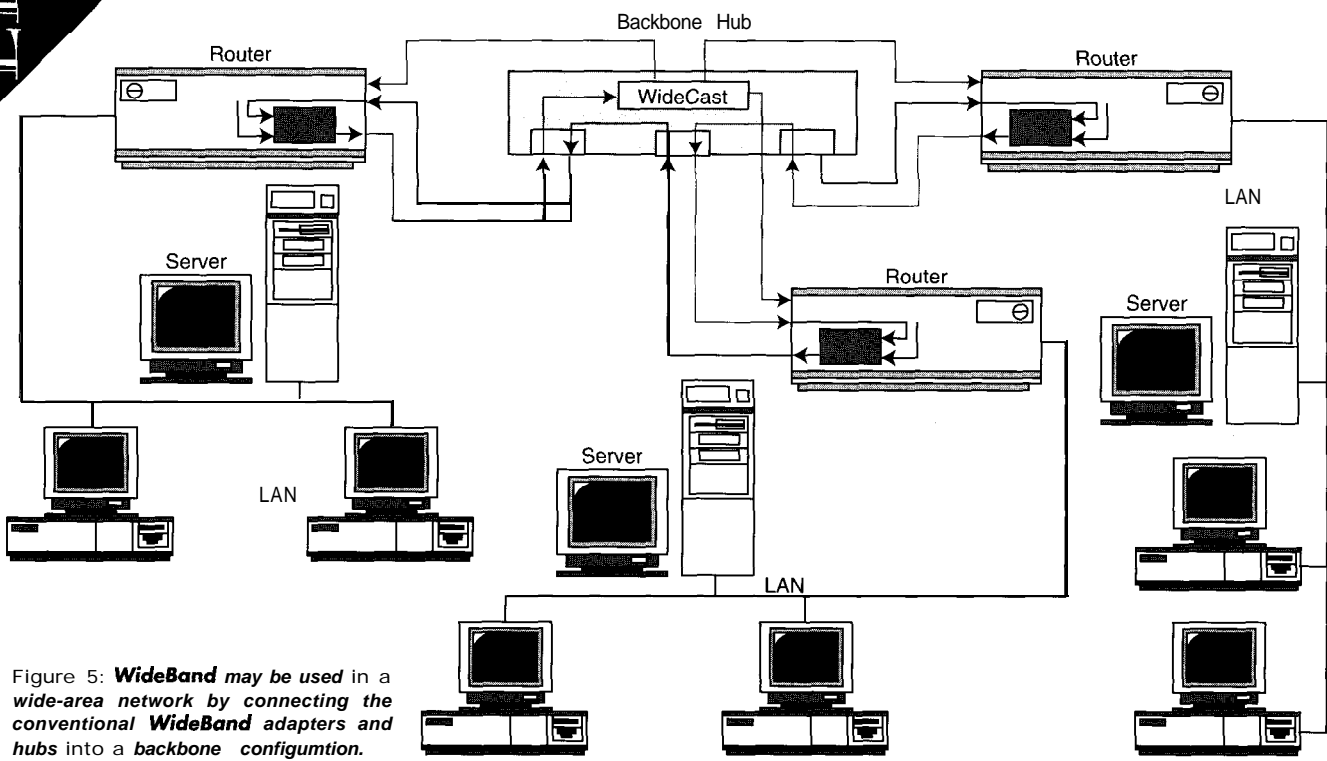


Figure 5: **WideBand** may be used in a wide-area network by connecting the conventional **WideBand** adapters and hubs into a backbone configuration.

If a remote packet is received by the state machine while it's transmitting a local packet, the remote packet is stored in the remote-input FIFO to be transmitted immediately after the local packet has completed transmission. Since combined output A transmits at the same data rate as

remote input B and since the remote-input FIFO can store an entire incoming packet, no data overflow occurs. By this method, local packets are inserted into the datastream.

WideCast inputs C and D have a special and unique function within a **WideBand**

network. The adapter is able to input data through either input C or input D, one at a time (selectable under software control). These two inputs to the local machine can be used in diverse ways, depending on the requirements and constraints of the individual installation.

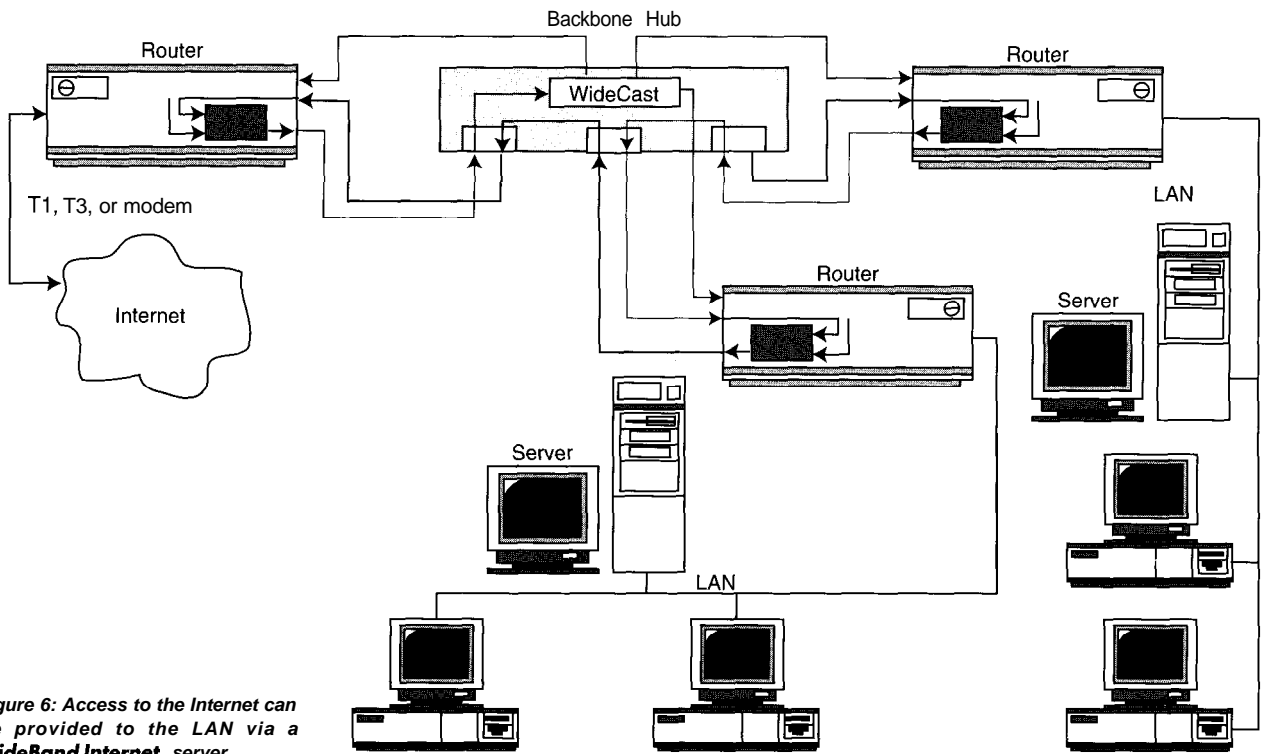


Figure 6: Access to the Internet can be provided to the LAN via a **WideBand** Internet server.

Figure 7: In a wide-area network configuration, **WideCast** input **D** distributes local area traffic and input **C** takes care of high-speed, enterprise-wide communications. This strategy simplifies E-mail installations and accelerates wide-area traffic.

THE WIDEBAND NETWORK

Figure 2 depicts the **WideBand** Network Adapter installed in the local area network. In this example, workstation 1 initiates the chained datastream. The local data transmitted over the network by workstation 1 travels to the hub, where it is rerouted (or chained) to input B on workstation 2.

As described in the previous section, workstation 2 synchronizes the transmission of workstation 1 with its own local transmission, sending the combined output to the hub. From there, it is chained to workstation 3.

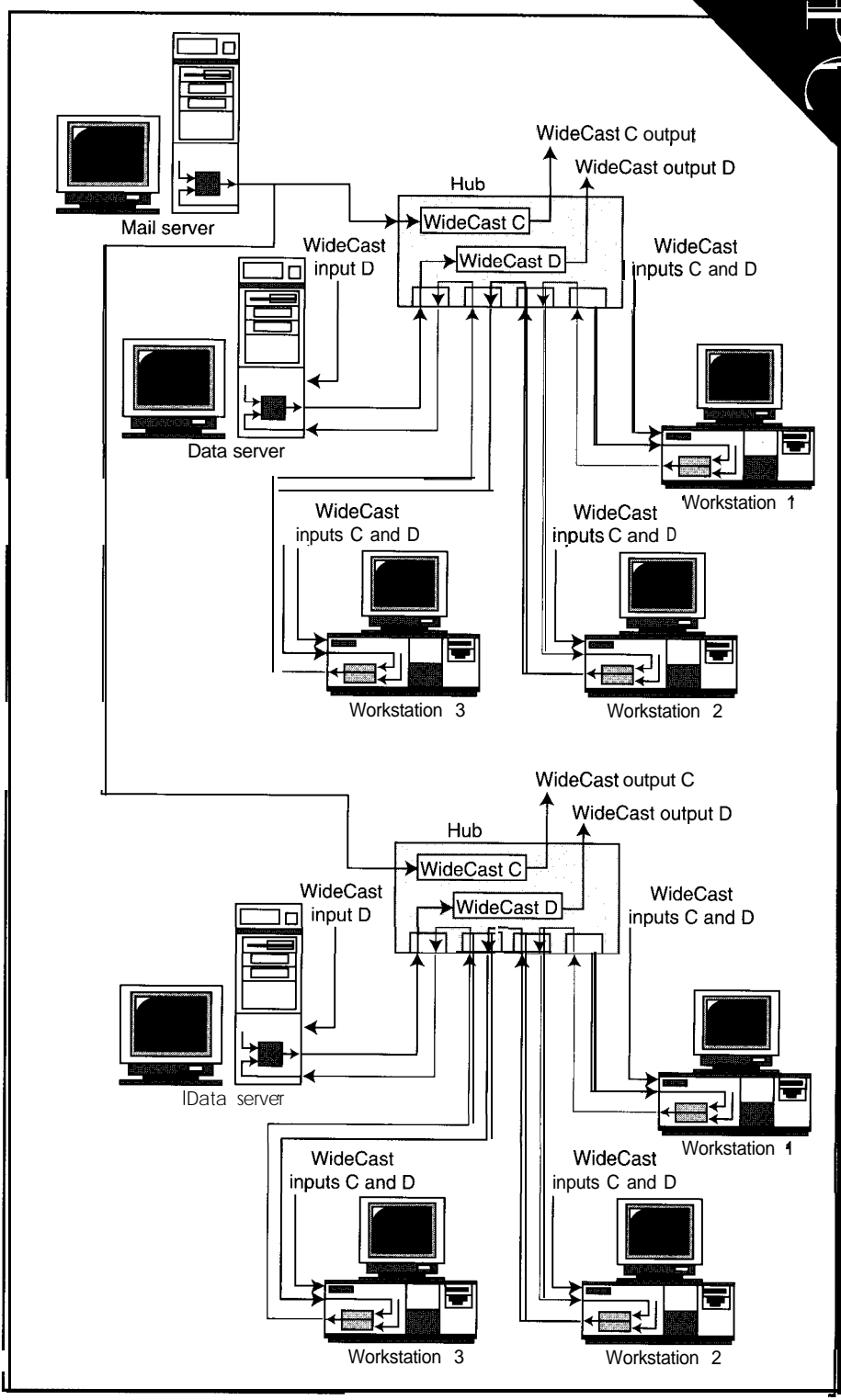
Transmissions from workstation 3 contain the combined queries or transmissions of all three workstations. These transmissions are then chained through the hub to server 1.

Although in many installations, data transmissions from the workstations are addressed to server 1, they are not picked off the datastream at this time. Instead, the combined transmissions are synchronized with the output from the server. The combined result then returns to the hub as an input to **WideCast** D.

In **WideBand** terminology, **WideCast** refers to a signal or transmission which is simultaneously sent to a number of computers. In the example illustrated in Figure 2, **WideCast** output D is delivered simultaneously as **WideCast** input D to each of the workstations and to the server.

The output D signal is delivered to every computer connected to the hub. (This connection has been omitted from the figure for simplicity.) The output is transmitted over pair four of the UTP5 cable connecting each computer to the hub. Even in installations with **WideCast** channels C and D, only a single UTP5 cable is needed to provide all four data-communication paths.

As packets are received at each computer, the network adapter selects and retrieves those packets addressed to the local computer. Much like Ethernet, the network adapter monitors all packet trans-



missions on the channel. This is how **two-way** communication is achieved in the **WideBand** environment.

The chained method of synchronizing data transmissions by various computers is also similar in some ways to Ethernet. In both cases, only one computer at a time can actually transmit data. In the **WideBand** network, however, there are no data collisions so the data rate is faster.

Figure 3 shows a **WideBand** network which separates the workstations' transmissions from those of the servers. In the example shown in Figure 2, the bandwidth of a single **WideBand** hub limits the combined transmission of all workstations and servers. By dividing the chained outputs of the workstations from the chained outputs of the servers, available bandwidth is doubled.

As you can see, Figure 4 illustrates a simplified wiring diagram of a WideBand hub. The hub provides a simple method of connecting computers into a LAN. It's also possible to interconnect or cascade multiple hubs by using special I/O connections.

Since there's no need for collision detection with the WideBand design, the Ethernet limitation on hub interconnections is eliminated. Leaving a hub port vacant breaks the chaining from channel to channel. You can divide networks into smaller segments to increase bandwidth in implementations such as the dual-hub version depicted in Figure 3.

The WideBand hub's design makes innovative approaches to implementation possible. Figure 5

depicts a wide-area network in which the conventional WideBand adapters and hub are connected to a backbone configuration. In Figure 6, a backbone provides the LAN with access to the Internet. In this installation, a high-speed Internet interface (such as a T1 or T3) is coupled, full speed, all the way to the desktop.

In Figure 7, components are connected to a wide-area network in which WideCast D distributes LAN traffic. WideCast C provides a high-speed, company- or enterprise-wide channel which simplifies E-mail installations and accelerates wide-area communications.

When workstations are not monitoring WideCast D to retrieve local data, the adapter monitors WideCast C to retrieve locally addressed mail and messages.

In the implementation illustrated in Figure 8, two servers operate in parallel or mirrored mode. All server queries are delivered simultaneously to both servers over WideCast C from the server hub.

The output of file server 1 is delivered to each of the workstations over WideCast C from the workstation hub, whereas the

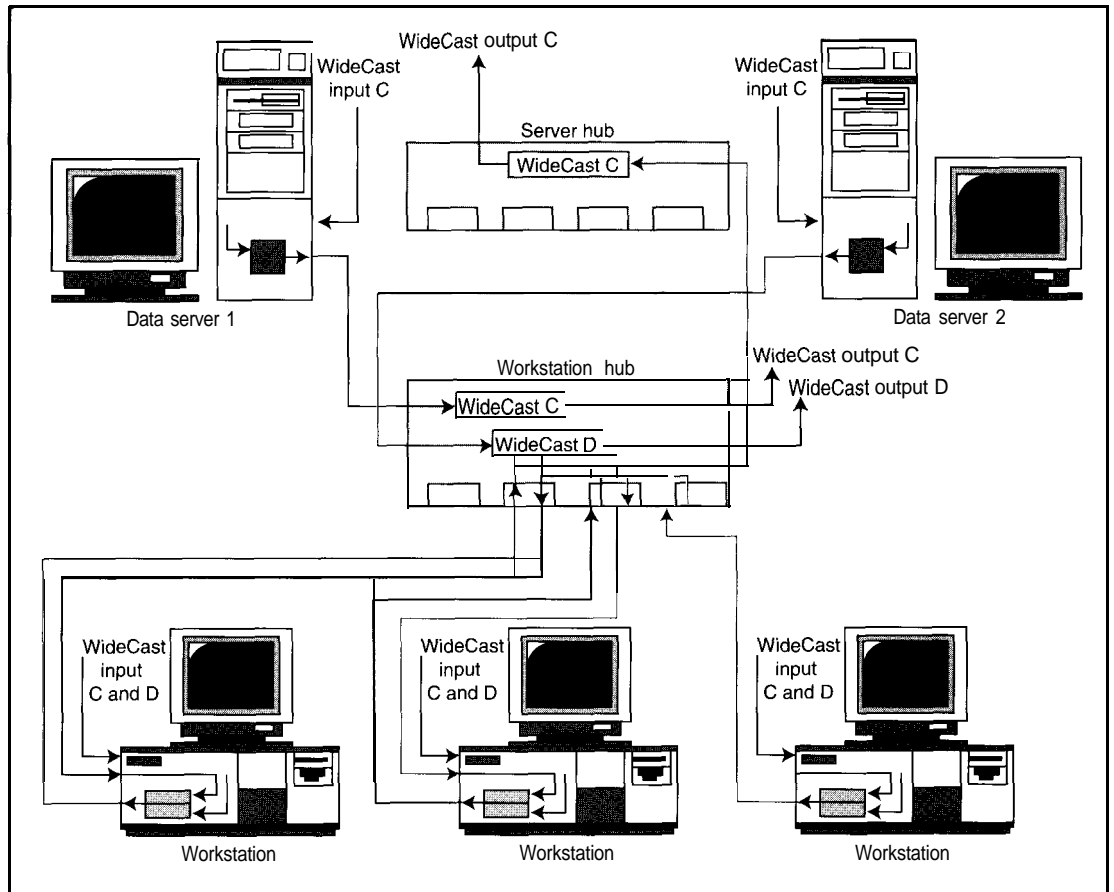


Figure 8: One configuration of a WideBand network is the "mirrored" mode, which provides complete redundancy of the data servers, even down to the cable pair.

output transmissions of server 2 are delivered to each of the workstations via the workstation hub WideCast D.

When WideBand is configured in this manner, both hubs operate in parallel,

processing requests and responding to the workstations.

Significantly, however, the servers are completely independent and redundant, even down to the cable pair over which the

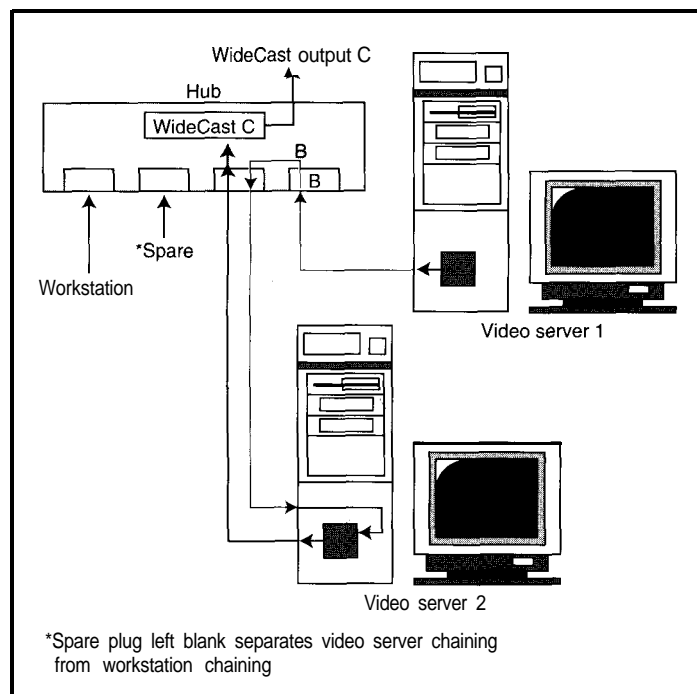


Figure 9: A WideBand network can also include remote video inputs. Video servers can be chained through the hub to provide many channels of video to the network.

Data Rate:	256 Mbps per cable pair
Byte Rate:	32 MBps per cable pair
Raw Bit Rate:	320 Mbps per cable pair (1 O-bit words)
Encoding:	8-B or 1 O-B ATM compatible
Output Signal:	100k ECL serial
Bit Error Rates:	10 ⁻¹² or better

Table 1: The technical specifications for **WideBand** are given for each cable pair of a UTP-5 cable.

server data is delivered to the workstations. If file server 1 malfunctions and a workstation therefore fails to receive a response to a request, the workstation can independently switch to input channel D and continue processing with server 2.

In Figure 9, a video source is connected to the hub's input channel B. A second video source is connected to the next port, continuing down the chain. Since video data is time sensitive, packets can automatically be sent over the network in synchronization with the demands of the video capture device, as shown.

TECHNICAL SPECIFICATIONS

In a **WideBand** network, data is transmitted asynchronously using ATM-compatible 8-B/10-B coding. The actual bit rate over the cable is 320 Mbps on each cable pair. Eight-bit data is converted into 1 O-bit data for transmission to:

- maintain clock synchronization
- provide a method of hardware error detection
- enable the transmission of control characters

The decoding of 1 O-bit data back to its original 8-bit format on the receiving side results in a useful data throughput of 256 Mbps per cable pair or 32 MBps. Table 1 shows the technical specifications of **WideBand**.

Transmission distances over UTP5 cable are presented in Table 2, along with transmission distances over other types of cabling and fiber. Applications requiring greater transmission distances use passive equalization to increase cable length. Table 2 also provides data-transmission distances for systems compensated with passive equalization.

CONCLUSIONS

WideBand networking is a sophisticated yet simple approach to increasing data-transmission bandwidths in local-and wide-area networks over existing cabling.

WideBand uses the basic technology of ATM but modified so it is more readily compatible with existing application software and the OSI seven-layer model. Through its many configurations, **WideBand** provides a versatile alternative in high-speed networking. **EPC**

Dr. Roger Billings, president of **WideBand** Corp., is *the* inventor of **WideBand** networking and client/server computing. He is a technical director *at the International*

Academy of Science *and may be* reached at billings@science.edu.

REFERENCE

[1] van Kirk, Doug, "ATM: Hype or Hope?" *InfoWorld*, October 30, 1995, 71-72.

SOURCE

WideBand Network Adapter
WideBand
 26900 East Pink Hill Rd.
 Independence, MO 64057
 (816) 220-3000

IRS

- 413 Very Useful
- 414 Moderately Useful
- 415 Not Useful

Cable Type	Uncompensated Transmission	Compensated Transmission
UTP-5 Unshielded Twisted Pair	40 m (130')	80 m (260')
UTP-3 Unshielded Twisted Pair	18 m (60')	Not Recommended
RG-58 A/U Coax (50 Ω)	35 m (115')	70 m (230')
RG-59 A/U Coax (75 Ω)	75 m (250')	150 m (500')
RG-62 A/U Coax (93 Ω)	98 m (325')	200 m (650')
Fiberoptic LED driver	1000 m (3300')	NA

Table 2: **WideBand** transmission distances vary with the type of transmission medium. Distance can **often** be increased with passive equalization.

Byte Craft

**C Compilers of choice —
for the chip of your choice!**

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

**We respond to your
C compiler needs.**

BYTE CRAFT LIMITED

Byte Craft Limited
 421 King St. N., Waterloo, Ontario
 CANADA N2J 4E4
 (519) 888-6911
 Fax: (519) 746-6751
 BBS: (519) 888-7626

MPC for PIC16/17Cxx

C38 for MELPS 740

COP8C for COP8

C6808 for MC68HC08

Z8C for Z8

C6805 for MC6805

Unscrewing the Inscrutable

With off-the-shelf components, tweaking becomes the norm for engineers. Here, Ed presents some references and sources he's used while writing decidedly nonstandard 80x86 programs and building oddball hardware.

Does anyone know why manufacturers call their data books "literature" rather than, well, just data books? Certainly, no data book has yet entered *The Canon of English Literature* and none, I hazard, ever will. The works have little to do with timeless verities of the human condition.

Nevertheless, anyone contemplating new code or hardware had best begin by cracking the books. Oral tradition, speculation, and experimentation take you only so far. At some point, you need The Facts.

One side effect of the PC revolution has been the proliferation of computer books available to the general public. Regrettably, Sturgeon's Rule ("Ninety percent of everything is [junk]") applies to technical publications as well as everything else. At your local bookstore, *The Undocumented PC* shares the shelf with *DOS For Dummies*. Although the latter may appeal to a wider audience, which has more value to an embedded programmer?

In this article, I'll discuss some of the references and sources I've used while

writing decidedly nonstandard 80x86 programs and building oddball hardware. While I can't pretend to cover the whole field, you'll get a good start on the subject.

Rather than burden the text with ISBN codes and prices, I've collected such minutia into the reference section. Check the end of the article for contact numbers and net addresses.

A FIRE AT THE CORE

Building 8051-flavored microcontroller systems requires relatively little esoteric hardware knowledge. When you venture into the PC market, however, design rules become much more gnarly. Wringing maximum performance from, say, a Pentium chipset driving a PCI Local Bus requires far more high-speed digital experience than most engineers [myself included!] now have or may ever acquire.

That scarcity of designers leads directly to the proliferation of embedded PC system boards. At this late date, you can, and probably should, buy a huge chunk of

predesigned hardware and get on with your project. Unless you have high volumes or have peculiar requirements, it makes no sense to design your own PC-clone board.

All the hardware on an embedded PC exists for the sole purpose of connecting the CPU to your I/O devices. From that perspective, you should be on very familiar terms with the CPU at the center of your project. After all, if that chip doesn't matter very much, what else does?

Intel pretty much defines the CPU standard in the PC arena. Their documents, while often reviled for tucking vital information in footnotes, describe the baseline functions found in clone CPUs and support chips. You'll find Intel's recent manuals present more information with much-improved style, so don't let your previous experiences (or folklore) dissuade you.

Rather than list Intel's voluminous collection of hardware data books and manuals, I recommend you get McGraw-Hill's complete Computer Books catalog. They now distribute all of Intel's manuals, so calling

Intel merely adds another step to the process. Pick the books relating to the CPU on your board and settle back for a long weekend's reading.

Although Intel has fewer software manuals, they're of equally high quality. The *Intel 486 Microprocessor Family Programmer's Reference Manual* may be the best guide to the vital minutia required for 80x86 programming. It seems many of the popular CPU-level books descend from this tome, typos and all. While you may need them, get this fundamental reference first.

For example, *Hummel's The Processor and Coprocessor* presents a somewhat less imposing and more readable introduction to the CPU with more descriptive text and diagrams. Unfortunately, it may be out of print, as I haven't seen it advertised lately.

For those writing system-control code, the *80386 System Software Writer's Guide* illustrates how you bolt typical operating-system routines onto '386 machinery. Later CPUs use much the same techniques, so don't be scared off by references to ancient '386 hardware. Those of you using '386EX chips should feel right at home.

Perhaps a project written in a high-level language doesn't need all the gruesome CPU hardware details. Once you descend into assembly language, these books become required reading. If your project needs protected-mode code, go directly to the source and study hard!

CONNECT THE PINS

Essentially by definition, a microprocessor requires support circuitry around the CPU chip. Early designs used discrete logic, which soon yielded to PALs of increasing complexity, until nowadays all the glue logic condenses into a single LSI block. For historical reasons, we call the support circuitry a "chipset" regardless of the actual number of packages.

In some systems, the support chip may be the biggest epoxy blob on the board. The '386EX moves much of that logic onto the CPU chip itself, thus blurring the distinction between microprocessors and microcontrollers.

As you might expect, system support chips contain dozens of registers regulating everything from DRAM timing to cache management. With all the logic on a single chip, you get the advantage of a consistent interface to the functions. The good news: smaller, cheaper, faster, better.

The bad news: different. Everybody designs those LSI blocks differently and tosses odd features into the mix. Make sure you have (or can get) documentation on those key chips!

In principle, the BIOS configures the chipset correctly whenever you turn the power on and, when it finishes, hands a properly tuned system to your code. If you plan on a BIOSless system or need a peculiar setup, good chipset documentation can be the make-or-break point of a deal.

Even though most of the standard PC peripherals appear on a single embedded-PC board, you also get an alphabet-soup mix of external bus interfaces: PC/104, STD32, PCI, VLB, ISA, and even VME. You

The sole substitute for an experience which we have not ourselves lived through is art and literature.

Alexander Solzhenitsyn

may need bus-level design or programming information to get the I/O gadgetry working, particularly if you plan to design your own unique peripherals.

Solari's ISA and EISA Theory and Operation remains the fundamental reference for desktop PC bus-hardware design, covering XT, ISA, and EISA signals, timing, and electrical specifications. Plan on spending lots of time with this volume, digging out all the relationships and figuring the implications. Not for programmers or diletantes!

Addison-Wesley's Trade Computer Books division recently published a category-killer series of books first introduced by Computer Literacy: Shanley and Anderson's *PC System Architecture Series*. Intended primarily for programmers, not hardware designers, they describe nearly everything about standard PCs.

While you can't throw away all your other references, these books tie together all those unrelated snippets of information. They descend from training courses run by *MindShare* and, as a result, their block diagrams tend toward "fat liners" suitable for overhead projectors. That quibble notwithstanding, the text explains complex subjects in sufficient detail that you understand not only what happens inside the

hardware, but why it works that way.

Start with *ISA System Architecture*, read 80486 S.A. (forgive the abbreviation) and *Pentium Processor S.A.*, then hit the *EISA S.A.*, *PCI S.A.*, or *PCMCIA S.A.* volumes as needed. In a mere three kilopages or so, you'll be up to speed. Worth every dollar and hour spent on 'em, honest.

For hardware-oriented readers, Frank Gilluwe's *The Undocumented PC* covers PC operation and BIOS functions with attention to the lowest register and bit levels. You'll find considerable insight into How It All Works with lots of sample code, diagrams, and tables of essential values.

None of these works can catapult a rookie hardware designer to the level required for contemporary PC design. They describe how to use existing hardware that meets the appropriate specs, without giving the extremely low-level electrical specs required to synthesize a new design. Judging from the Annabooks catalog, you may find those details in some of their other books, but I don't have them on my shelf and can't give a firsthand report.

ON THE BIOS LEVEL

Using PC-compatible hardware doesn't mean you can load and run a standard desktop program, however. Even for embedded-PC work, starting with a PC BIOS makes for much simpler development. You can, in principle, ignore many of the gritty chipset details and get on with the job.

If you go the whole course by embedding DOS atop the BIOS, you have a tidy, flat, rectangular, PC clone festooned with copyright stickers and an invoice cloyed with per-unit royalty payments. Depending on your application, this may actually be a Good Idea. For others, a compatible BIOS and your own well-written code gives the best payback.

Both Phoenix and AMI publish books detailing their BIOS interfaces. Neither goes into much behind-the-scenes detail, so you must look elsewhere (perhaps into *The Undocumented PC*) for undocumented interfaces and tricks. However, you will find lists of software interrupts with their input and output register values, along with some text that describes the functions in detail.

If you can tolerate even less description, Brown and Kyle's *PC Interrupts* tabulates every interrupt and function used by every

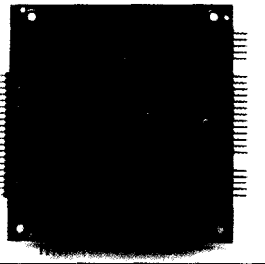


Sets the Pace in Low Power, High Performance PC/104 Technologies



50 MHz
486 SXLC2
\$543

Features:
486SXLC2
50 MHz
387SX
8K cache
2MB DRAM
3W typical
SSD
EEPROM
WDT
IDE & FDC
2 Serial
EPP
PS/2 mouse
Keyboard
Quick Boot
Virtual devices

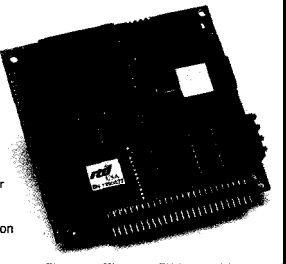


**CMi486SXLC2-1 Fully Integrated PC-AT
with Virtual Device Support**



200 kHz
12-bit DAS
\$498

Features:
200 kHz
12-bit A/D
16SE/8DI
Scan
Burst
Multiburst
1K CGT
Triggers
FIFO buffer
Data marker
12-bit D/A
16-bit DIO
+5V operation



**DM5408-2 200 kHz Analog I/O Module
with Channel-Gain Table**

**Make your selection from:
9 cpuModules™**

SuperXT™, 386SX, 486SXLC2, and 486DX2 processors. SSD, 8MB DRAM, RS-232/422/485 serial ports, parallel port, IDE & floppy controllers, Quick Boot, watchdog timer, power management, and digital control. Virtual devices include keyboard, video, floppy, and hard disk.

7 utilityModules™

SVGA CRT & LCD, Ethernet, keypad scanning, PCMCIA, intelligent GPS, IDE hard disk, and floppy.

18 dataModules®

12, 14 & 16-bit data acquisition modules with high speed sampling, channel-gain table (CGT), sample buffer, versatile triggers, scan, random burst & multiburst, DMA, 4-20 mA current loop, bit programmable digital I/O, advanced digital interrupt modes, incremental encoder interfaces, opto-isolated digital I/O & signal conditioning, bpto-22 compatibility, and power-down.

&Real Time Devices USA

200 Innovation Boulevard • P.O. Box 906
State College, PA 16804-0906 USA
Tel: 1 (814) 234-8087 • Fax: 1 (814) 234-5218
FaxBack®: 1 (814) 235-1260 • BBS: 1 (814) 234-9427

RTD Europa RTD Scandinavia
Budapest, Hungary Helsinki, Finland
Fax: (36) 1212-0260 Fax: (358) 0 346-4539

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

BIOS, DOS, I/O adapter, and programming API under the sun. You can also get the information, plus network manager interrupts, on their *Uninterrupted Interrupts* CD-ROM with a Windows multimedia viewer (!) interface.

Hogan's Programmer's PC Sourcebook has no descriptive text whatsoever to distract you from the tables and charts, which cover everything from chip pinouts to cable wiring, card sizes, and Windows API functions. Although you can look up something you distinctly remember forgetting, this book will not teach you anything. Each entry includes bibliographic pointers to the original sources, which simplifies figuring out where you went wrong.

MESSES, METHODS, AND ALGORITHMS

One key advantage of an embedded PC lies with software: you use essentially the same compilers and debuggers made familiar by long hours at desktop PCs. In effect, you have a standard PC squashed into a different form factor rather than an alien monstrosity requiring entirely different development tools and techniques.

A second advantage appears at the bookstore. Regardless of which PC programs you use, someone has already written the book. Need help with compilers, linkers, debuggers, editors? You've got it! While stores may devote a shelf or two to other desktop machines, a quick look illustrates the difference. The purist may argue that quantity doesn't imply quality, but the fact remains that you'll get more good books on a subject if you have more books.

For example, the sheer number of people rummaging amid the innards of PCs, coupled with the (relatively) standard hardware and software, makes Addison-Wesley's Undocumented [Whatever] series not only possible, but profitable. Yes, UNIX hacking has a long tradition, but check your bookstores for the results.

If you work with embedded DOS, a copy of Schulman's Undocumented DOS provides the background you need to understand compatibility problems and the raw information to track down glitches that crop up. Give it a quick read to acquaint yourself with the topics, then hold it in reserve to smash problems as they pop up.

Apart from a familiar environment, an embedded PC provides performance in nicely-sized increments. Assuming that your

problem depends more on computing power than I/O bandwidth, you can pick any speed you like between 8088- and Pentium-class CPUs. With a bit of foresight and effort, you can use the same program on any CPU with no recompilation.

For a given CPU, though, performance depends more on your choice of algorithm than your bit-twiddling ability. Sedgewick's Algorithms has the techniques required to solve fundamental problems you'll encounter in embedded work. The code fragments use Pascal rather than C, which shouldn't pose a problem to anyone reading this magazine.

Knuth's seminal *The Art of Computer Programming*, now somewhat dated, still deserves a place on your shelf and a part of your head. He covers most of the topics you need to know and provides a basis for understanding the more recent books and articles. Expensive and worth every penny.

In truth, however, Knuth's mathematical formalism may be daunting despite his relaxed writing style. I find that reading his work as part of collecting the information for a given task gives better results than attempting to digest the whole kit and caboodle at once. Your mileage may differ. Check it out at a local bookstore first.

Programmers having little familiarity with fundamental algorithms generally resort to naive solutions that either work slowly or not at all. If you can recognize a problem, look up the best solution and implement it in your code. You'll be well ahead of the competition. Be advised!

If your situation justifies an 80x86 CPU in the first place, it probably involves numeric calculations. The floating-point routines included with your compiler may suffice (beware of reentrancy issues in multitasking environments!) or you may prefer to roll-your-own optimized routines. Morgan's Numerical *Methods* solves common numeric problems, ranging from integer arithmetic to polynomial evaluation. His minimal coverage of trigonometric and transcendental functions probably mirrors the need for those routines in practice.

Once you have the algorithm nailed down, Abrash's *Zen of Code* Optimization turns on the nitpicking, obsessive, 80x86 assembly-language afterburner. You won't need this level of optimization very often, but nothing can replace a keen mind armed with some good algorithms and detailed hardware knowledge. Beware: this stuff

can be so fascinating that you won't get anything else done. Save it for last!

You'll find several useful books in the lists that I don't have room to discuss here. Given the tonnage of books appearing every day, you'll surely discover others that merit careful study. Expect to buy a few duds along the way, don't believe everything you read, and for sure, don't believe the hype!

SHOP 'TIL YOU DROP

Buying technical books requires much less effort than it used to, if only because publishers realize how large the PC market has become. Unfortunately, the drive to occupy shelf space attracts far more bad books than good ones, which means you must weed through the offerings rather carefully. A good first step requires nothing more than several hours browsing at all your local bookstores.

If your area lacks good bookstores with a wide selection and they can't (or, perish the thought, won't) order the titles you need, start shopping by mail. Many publishers advertise in the usual PC magazines, and you'll find useful books tucked among the debris.

I find the book reviews in Dr. Dobb's Journal particularly useful, although they don't generally cover embedded-programming topics. *Visual Developer*, formerly PC Techniques, presents excellent reviews with a bias toward visual programming and higher-level design. Folks on the Circuit Cellar BBS have more technical books than most libraries and enjoy helping, too.

You can look up books by title, author, or ISBN as needed. I've omitted the publisher and copyright information, as you can find that easily enough at any bookstore or library by referring to their copy of Books In Print. It may well be that some books I recommend now appear in a new edition with a different ISBN or, alas, have fallen completely out of print.

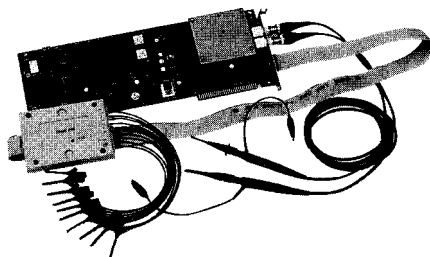
You'll see some blanks in the book lists where my collection lacks the most recent edition. If you can't find them on the shelf at your local bookstore, the staff can surely track them down by author or title without too much trouble. Keep your eyes open for new books while you're there!

McGraw-Hill handles Intel manuals and data books, as well as a host of other publications. You can examine them on the shelves of your local store, order by phone,

PC-Based Instruments

200 MSa/s DIGITAL OSCILLOSCOPE

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



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

DSO Channels

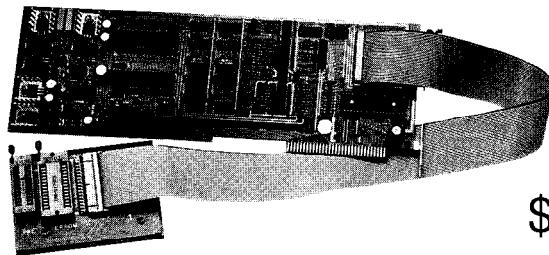
2 Ch. up to 100 MSa/s
or
1 Ch. at 200MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

Logic Analyzer Channels

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

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
PIC
etc..

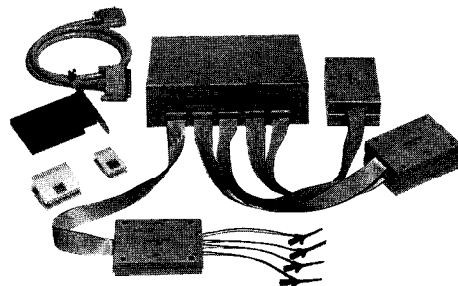


\$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

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



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

Call (201) 808-8990

Link Instruments



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

surf the net, or access their CompuServe catalog. Addison-Wesley publishes a wide variety of good-quality technical books. Trying hitting their Web page or you can find their Trade Computer Books titles in most larger bookstores. Try as I might, I can't find a direct phone-order number, which implies they'd rather have you walk into a bookstore.

Annabooks carries specialized PC books and embedded PC software. They also offer seminars covering embedded PC topics. You can phone or fax your orders.

The Coriolis Developer's Club has an excellent selection of technical books, software, and CD-ROMs, with a heavy accent on graphics. Order via phone, fax, or net. They have free shipping for five books and knock off 10% from the bottom line for ten, making them a good place to stock up.

Computer Literacy Bookstores do a roaring mail-order business. You're welcome to phone or access them via the Web. They carry a surprising number of microcontroller books in addition to PC, mini, and mainframe (yikes!) tomes.

You can view and download quite a selection of technical information, data

sheets, and app notes from various corporate Web sites. I've had good luck feeding company names into the Yahoo search engine, although the usual three-letter abbreviations generate far too many matches. Many high-profile companies format their datasheets with Adobe's Acrobat reader, so plan on a lengthy download when you get started.

Although I've concentrated on books in this article, you can find a wealth of unpublished online information if you have the time to search for it. My casual rummaging tells me that I'm better off allowing someone else the honor of collecting, collating, and verifying the information, but your balance point may be different.

Even though the Internet and World Wide Web get all the publicity nowadays, I find that CompuServe provides more tightly focused discussions and to-the-point responses. You can unearth vast stores of useful information in their libraries, particularly in forums dedicated to single topics or companies. Make sure you use an offline reader such as TapCIS, ATO, or Golden CommPass, as you can still rack up a substantial charge even with their new rates.

IN CONCLUSION

Despite the overlap among all these books, I cannot recommend just one in any particular field. Reading several books on a subject gives you a better grasp of the subject and, perhaps more importantly, allows crosschecking to weed out typographic errors.

The most egregious example I've found lies hidden in a table of PC interrupts. One entry seemed out of place and, after consulting my other references, I realized that the authors had somehow converted a hex interrupt number into decimal, then listed the result as a hex value. Just to confuse things, the correct hex value appeared elsewhere in the table.

Always crosscheck your facts!

In any event, books can provide the knowledge required to start a project. With the fundamentals well in hand, you can avoid many of the blunders I've witnessed (and, occasionally, participated in) over the years.

Nothing, however, replaces experience. As the old saying puts it, "Good judgment comes from experience. Experience comes from bad judgment."

Go forth and get experienced! EPC

Ed Nisley (KE4ZNU), as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at ed.nisley@circellar.com or 74065.1363@compuserve.com.

REFERENCES

Intel Handbooks and Data Books

<i>Intel486</i> Microprocessor Hardware Reference, 240552	\$28
Intel486 <i>Microprocessor Family Programmer's Reference Manual</i> , 240486	\$28
80386 <i>System Software Writer's Guide</i> , 23 1499	\$18
Microprocessors, Vol. 1, 230843 (covers 8086, 80C186, 80286, 80386, and support hardware)	\$25
Microprocessors, Vol. 2, 24173 1 (covers 80486, support hardware, app notes)	\$25
Microprocessors, Vol. 2, 24 1732 (covers Pentium, support hardware, app notes)	\$20
Peripheral <i>Components</i> Handbook, 296467 (glue and system logic for PCI, EISA, DMA, cache)	\$24

CPU, Bus, and BIOS Information

Anderson, <i>PCMCIA System Architecture</i> , O-201 -4099 I-7	\$30
Anderson and Shanley, <i>Pentium Processor System Architecture</i> , O-201-40992-5	\$25
Brown and Kyle, <i>PC Interrupts</i> , O-20157797-6 (replaced by 2nd ed., priced at \$40)	\$33
Also available as <i>Uninterrupted Interrupts CD-ROM</i>	\$50
Gilluwe, <i>The Undocumented PC</i> , O-201-62277-7	\$45
Hogan, <i>Programmer's PC Sourcebook</i> , I-556 15-32 1-X	\$40
Hummel, <i>The Processor and Coprocessor</i> , I-56276-01 6-5	\$50
Programmer's Guide to the AMI BIOS, O-07-001 561-9	NA
Shanley, 80486 <i>System Architecture</i> , O-20140994-1	\$20
Shanley and Anderson, <i>ISA System Architecture</i> , O-201-40996-8	\$35
Shanley and Anderson, <i>PCI System Architecture</i> , O-201-40993-3	\$35
Solari, <i>ISA and EISA Bus Theory and Operation</i> , O-929392-1 5-9	\$90
System BIOS for IBM PC/XT/AT Computers and Compatibles, O-201 -5 1806-6 (replaced by <i>System BIOS for IBM PCs, Compatibles, and EISA Computers</i> , priced at \$30)	\$27

Firmware, Software, Algorithms, and Techniques

Abrah, <i>Zen of Code Optimization</i> , I-883577-03-9	\$40
C Programmer's Guide to Serial Communications, O-672-22584-O (replaced by 2nd. ed., \$40)	\$30
Knuth, <i>The Art of Computer Programming</i> , Vol. 1, <i>Fundamental Algorithms</i> , O-201 -03809-9	NA
Vol. 2, <i>Seminumerical Algorithms</i> , O-201 -03822-6	NA
Vol. 3, <i>Sorting and Searching</i> , O-201 -03803-X	NA
Maguire, <i>Writing Solid Code</i> , I-5561 5-551-4	\$25
McConnell, <i>Code Complete</i> , I-556 15-484-4	\$35
Morgan, <i>Numerical Methods</i> , I-5585 I-232-2	\$37
Nelson, <i>Serial Communications</i> , I-5585 I-28 1-O	\$45
Schneier, <i>Applied Cryptography</i> , O-471 -59756-2	\$45
Schulman, et al., <i>Undocumented DOS</i> , O-201-63287-X	\$45
Sedgewick, <i>Algorithms</i> , O-20 I-06673-4	NA

SOURCES

McGraw-Hill
(800) 822-8158
<http://www.mcgraw-hill.com/>
CompuServe catalog: GO MH

Addison-Wesley
<http://www.aw.com/>

Annabooks
(800) 462-1 042
Fax: (619) 673-1 432

Coriolis Developer's Club
(800) 41 O-01 92
Fax: (602) 483.0193
<http://www.coriolis.com/>

Computer literacy Bookstores
CA: (408) 973.9955
VA: (703) 734.7771
<http://www.clbooks.com/>

IRS

4 16 Very Useful
4 17 Moderately Useful
418 Not Useful

Dave Cox and Paul Olsen

Chassfs and Enclosures

A PC/104 Packaging Overview

Different problems need different solutions. This is especially true in packaging where real estate, power, temperature, shock, and so on can have a dramatic effect. Dave and Paul give an 'overview of what is out there in PC/ 7 04 design.

Like a jeweler selecting a setting for a precious stone, you must carefully choose the appropriate chassis and enclosure for your **PC/1 04** system. By doing so, you can fully address application requirements.

As you probably, know from fielding systems in the real world, no one solution is optimum for any given problem. Solutions must always be viewed in the context of the problem at hand, uncontrolled variables, and future requirements.

The classic paradigm of form follows function is not applicable to most embedded systems. Aesthetic considerations must take a backseat to the more immediate concerns of environments, interoperability, modularity, agency approvals, maintainability, upgrades, expandability, and a host of other concerns.

In most cases, you must also consider the NRE costs, such as tooling, projected sales volume, and target unit pricing, to arrive at a make-or-buy decision for the

final package. You may find it worthwhile to review the general requirements and the way they are addressed by off-the-shelf solutions available through PC/1 04. Solutions are often rugged, predefined, and highly general in nature.

BEGIN WITH THE BASICS

In the PC/1 04 world, system elements usually take one of two basic configurations: baseboard and stack.

Rick Lehrbaum of Ampro has dubbed a stack of PC/104 modules as macrocomponents. They contain a power supply and peripheral hardware or devices.

When macrocomponents are mounted on a baseboard, PC/1 04 modules enable a reduced height profile for the chassis (i.e., as little as one inch), which enables the end item to mount behind a display panel or other enclosure as a substrate.

The basic question you must address: should the baseboard fit the chassis or

should the chassis fit the baseboard? Again, unit cost, lead times, and NRE fees dictate your choice.

For stand-alone systems based on a PC/1 04 stack, the form factor is a given. Your choice is reduced to the type of footprint (i.e., horizontal or vertical). The stack provides inherent structural rigidity and volumetric efficiency, fitting in tight or confined spaces. Applications for this configuration include mounting on a gantry, in an engine compartment, or as an articulating member of a robotic manipulator.

Whether you select stack or baseboard configurations with PC/1 04 macrocomponents, attention must be given to packaging fundamentals. You need to take into account such factors as weatherproofing; susceptibility to shock, vibration, or resonance; EMI/RFI; Tempest shielding; thermal management; and the interconnection scheme for I/O, power, and communications.

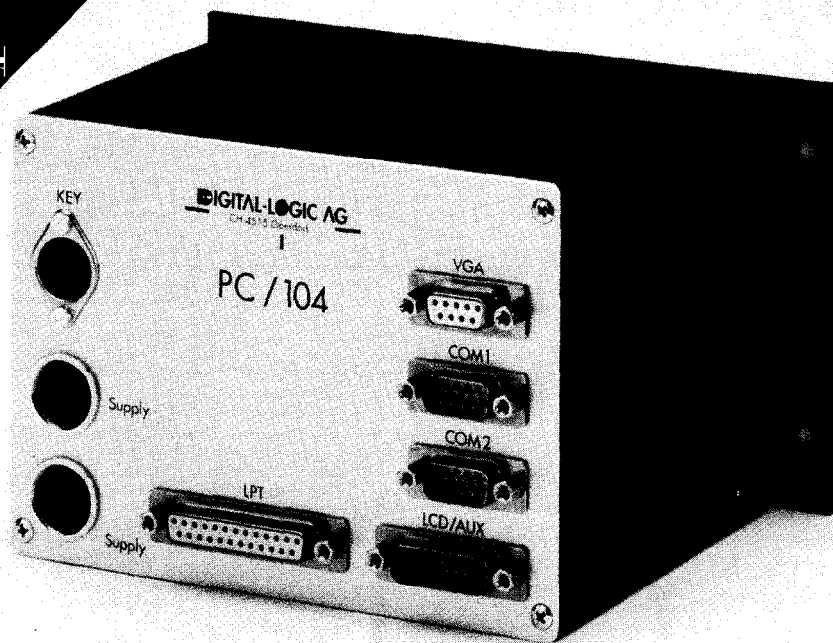


Photo 1: The **MicroSpace PC/104** enclosure supports 1-3 PC/104 boards. Options include Ethernet, floppy disk, hard drive, and power supply. All connections are on the front and it comes equipped for wall mounting and DIN rails. Its dimensions are 176 x 105 x 135 mm.

Ancillary considerations include weight (or moments of inertia), intrinsic safety requirements, true earth grounding, mounting plates or points of fixation, galvanic compatibility, cathodic protection, and so forth. When you've taken these things into account, you are now faced with the make or buy decision.

WHAT'S UNDER THE LID

Before launching into a custom design, you should consider off-the-shelf chassis and enclosure products from Consortium member companies listed in the *PC/104 Resource Guide*.

The Microspace PC/104 enclosure from Digital Logic in Switzerland supports one to three PC/104 modules and an optional 3.5" floppy drive. This 6.25" x 3.75" x 4.8" chassis is pictured in Photo 1. Its 8-W bus power supply operates from +5 VDC or an optional +24 VDC input. The connector panel provides a 5-pin DIN; two 9-pin, a 15-pin, and a 25-pin D sub; and one 1 Obase2 connector.

Another compact solution shown in Photo 2 is the **ENC 104-3** from Micro/Sys. This two-piece, anodized aluminum enclosure accepts a CPU baseboard and is deep enough to hold up to three additional PC/104 modules. It measures 10.0" x 5.25" x 3.495" and has lowered sidewalls

for heat dissipation. Its cable exits correspond to common PC/104 module connector regions. A cable panel on one end includes four DE-9, one DB-25, and two 5-pin DIN connectors. Also, 4" x 1.4" cable cutouts are on two sides. The removable cover is held in place with captive screws.

Kinetic Computer has come up with the **RCC-104**, an innovative chassis with a sealed and conduction-cooled rugged enclosure which accommodates a 3-8-module PC/104 stack. The stack is mounted with the bottom and the top of the stack attached to the enclosure, providing a stable, vibration-resistant mount. Industry standard D-sub, DIN, and circular MIL-type connectors are mounted to the side panel of the enclosure, as shown in Photo 3.

Another unit, the **RCC-104PZ** is a 10.3" x 1.3" x 2.5" pizza-box-shaped enclosure with an integral carrier board that accepts up to three PC/104 stacks, three modules high. It offers an optional vehicle power supply built onto the carrier board.

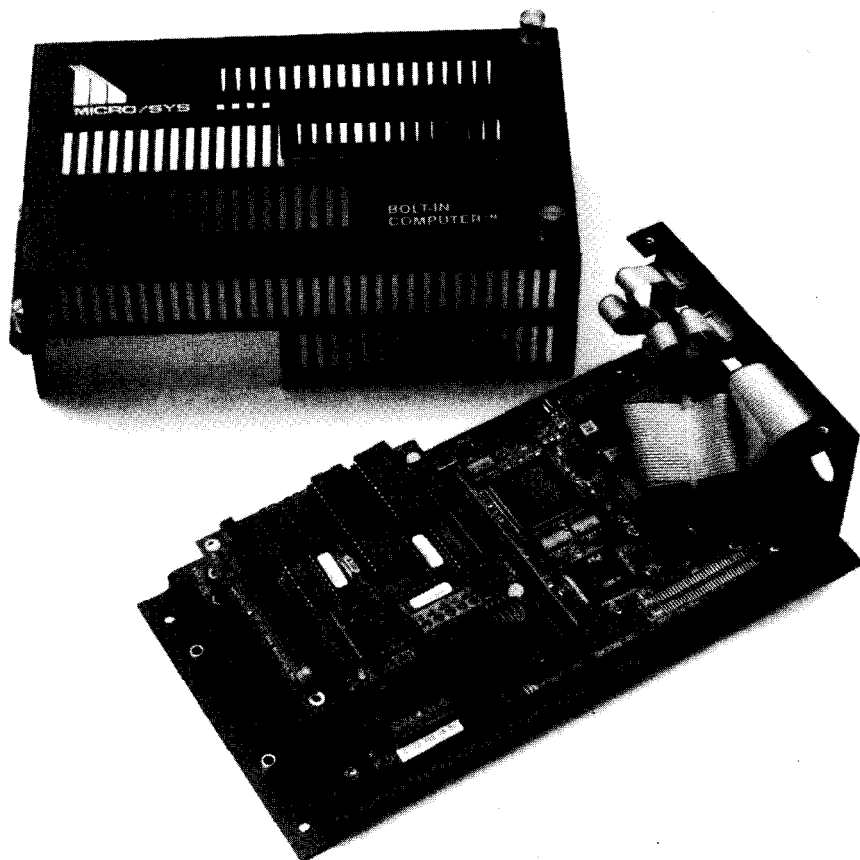


Photo 2: The **ENC104-3** enclosure provides mounting and connector cutouts for an embeddable PC and up to three PC/104 add-on cards. It is a two-piece, anodized aluminum design with connector cutouts on one end. The cover can be removed without disconnecting wiring and is lowered for heat dissipation.

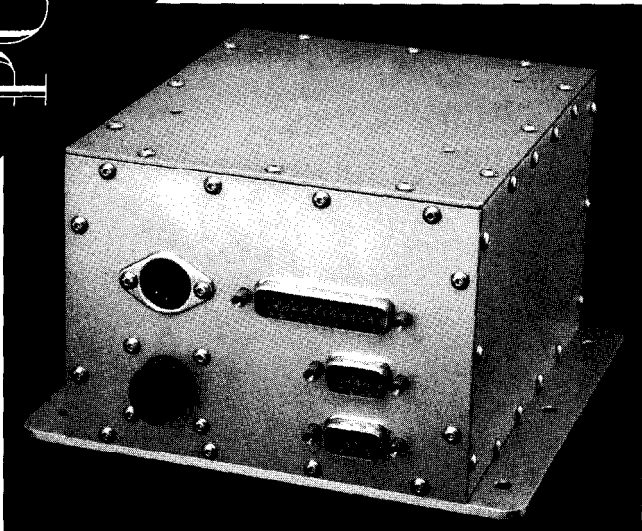


Photo 3: This durable black-anodized aluminum construction is NEMA 4/12 sealed and is conduction cooled. It provides side-panel cutouts for customized connections and is offered with an optional integral power supply. Custom mounting options are available.

The cube and half-cube enclosures from Enclosure Technologies provide another unique PC/104 packaging solution. The half-cube is very compact. It measures 5.5" x 2.75" and can contain three stackthrough and one chassis- or baseboard-mounted PC/104 modules. Because the half-cube and full-cube enclosures are both built using the same base, you can upgrade system functionality to include more modules by simply replacing the card guide and cover with cube-sized equivalents.

The full-cube dimensions are 5.5", and it can contain one baseboard-mounted and seven stackthrough modules. The enclosures are fabricated from 0.050" black anodized aluminum. As you can see in Photo 4, each has a user-configurable I/O panel with knockouts for DE-9, DB-25, and BNC connectors.

An alternative strain-relief panel for external ribbon cables is also available. The internal card-support bracket lets you assemble the PC/104 stack without using standoffs. A quick release version is offered for industrial applications. The company also has an optional 30-W PC/104 stackable power supply with a 12-24-VDC source, shock mounting kit, and wide variety of accessory fans, cables, and interconnect options.

The two most common approaches to using PC/104 modules—either as standalone stacks or in component-like applications—do not fully address the need for enclosing or housing PC/104 modules. The desire to implement the PC/104 stack as an independent product, complete with its own power source and I/O connec-

tions, is becoming more attractive to developers.

These enclosures are examples of how Consortium members are expanding the use of PC/104 modules by providing several off-the-shelf solutions to enclosure demands.

Power requirements vary greatly between designs, so a single-power solution is difficult if not impossible.

Most enclosures are made of some form of metal material. Other types of plastic may be available in the future, providing new ways to use PC/104 stacks.

Addressing the interconnect needs and mounting requirements is also a challenge.

As more designs make their way into the PC/104 form factor, providers of enclosures and housings must be able to re-

spond to a variety of requirements. Key is the need to find solutions for harsh environments, which traditionally have been off limits to PC-type applications.

FAST RAIL TO SUCCESS

The PC/104 chassis system from parvus consists of several primary elements that, when combined, create an embedded PC/104 chassis. These elements can have many options each, thus creating hundreds of possible chassis configurations.

As shown in Photo 5, the parvus solution uses a system of slotted rails and end caps to provide total physical modularity. The rails are fabricated from 6061-T6 anodized aluminum and have precision-machined slots which accept and secure the corners of each PC/104 circuit board. The rails come in 4", 6", and 8" lengths and hold up to 5, 8, and 11 PC/104 cards, respectively.

The zinc-plated #20-gauge-steel end caps hold four rails per stack rigidly in place. The entire assembly measures 4" on each side (complete with cards installed), by the standard length you select. The result is a unique, modular chassis system that lets you construct a PC/104 stack without the tedious placement of spacers and standoffs between circuit modules.

This technique facilitates rapid removal and replacement of defective modules or reorganization of the stack as you tweak the final assembly for thermal management, connector orientation, and cable routing. It also provides four fixed points for

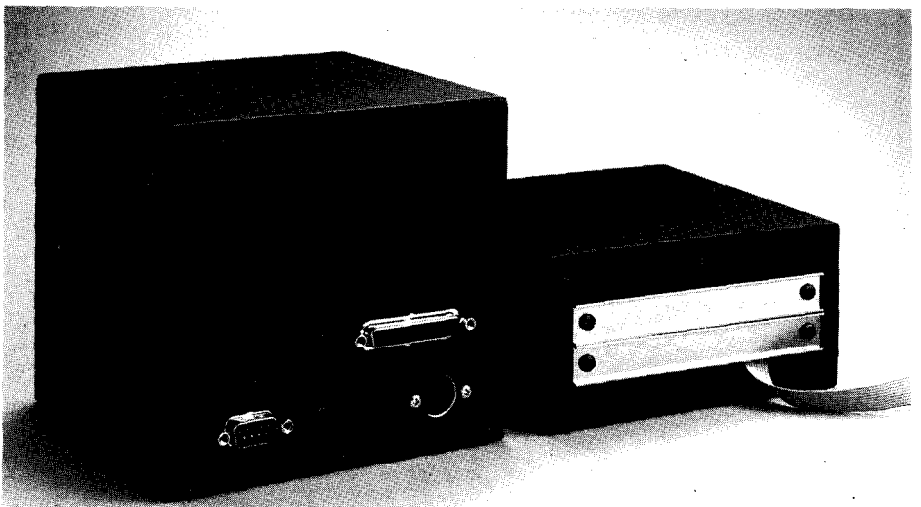


Photo 4: The cube and half cube are designed to be compatible. They are all aluminum construction with black anodized brush finish. Features include card-edge support brackets that eliminate standoffs, industrial quick-release mount, and user-configurable I/O panels.

every card in the system stack, making the overall assembly more robust and shock resistant than a stack bolted to a base-board.

It is far superior to a standard ISA bus configuration with card guides and a single mounting bracket on the end of each card. With it, you totally avoid the assembly issues associated with the non-symmetrical mounting holes in the PC/104 board.

A family of end cops provides a variety of needs. You can select solid cops, cops with cutouts for wiring harnesses, cops with bezel openings for LCD displays or key-pads, cops with mounted connectors, caps incorporating terminal blocks, and caps with peripheral connector holes so you can easily install and connect your wiring harness to standard PC and sub-D miniature connectors.

The end caps attach to all four rails on each end, creating an extremely rugged card cage that contains a PC/104 stack. The complete card cage assembly simply plugs into power, connects to your I/O and video, and is put into operation.

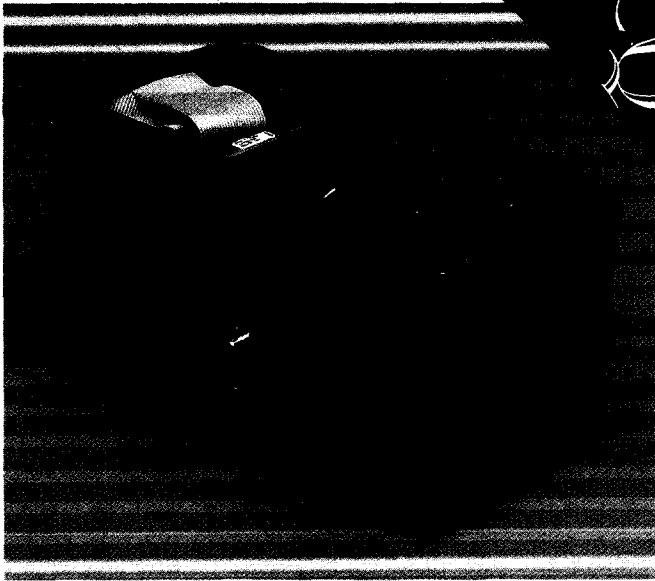
The structural integrity of this system was subjected to a brute force test at parvus one morning when a pickup truck was deliberately driven onto a PC/104 card cage assembly causing only cosmetic damage! While I don't recommend this test of finished goods, it certainly illustrates the cage's ruggedness.

The internal chassis system, consisting of the four rails and two end caps, can be further enhanced using optional applicator caps and graphic overlays. The applicator caps are specific to a particular function. The cap attaches to the end of the internal chassis frame, and in most cases contains a PCB that performs a specific operation. Examples include an LCD or keypad, I/O terminals, graphic displays, and an external connection system.

The graphic overlays make the end or applicator caps look more polished and complete. On LCD applicators, the graphic overlay also becomes a bezel. Other overlays include labels for standard PC connections.

This card cage system can be further enhanced by parvus's PC/104 subchassis and external chassis elements. The subchassis consists of a #20-gauge-steel internal frame, which encloses an 8" internal chassis. The resulting unit creates three

Photo 5: Machined aluminum slotted rails tightly hold all four corners of this PC/104 board in place. Offered in different lengths, an entire PC/104 stack can be mounted in place. Steel endcaps give the designer flexibility. Application-specific boards can be added to the end of the stack in a compact 4" x 4" package.



1" x 4" drive bays and includes internal mounting tabs. An external chassis can consist of either a 4" x 5" extruded aluminum enclosure with lengths of 4.5", 6.5", or 8.5" or a MicroTower, consisting of an 8" internal chassis with three drive bays. Finished, it appears as a miniPC tower.

parvus has also enhanced their rail cages by offering 19" rack mount hard-

ware, various PC/104 form-factor power modules, and snap-track mounting plates. Photo 6 shows several iterations of these enclosures.

A patent pending modular extrusion called the QB.LOK features machined tongue-and-groove outer surfaces on all four sides. This surface enables several enclosed PC/104 systems to securely inter-

For Borland C/C++, Microsoft C/C++, Borland Pascal

RTKernel

Real-Time Multitasking for DOS

RTKernel is a professional, high-performance, real-time multitasking system for MS-DOS and Embedded Systems. It can use DOS device driven and BIOS, and runs other DOS applications as a task - even Windows!

RTKernel is loaded with features: an unlimited number of tasks, excellent performance, a full set of inter-task communication functions (semaphores, mailboxes, synchronous message-passing), real and protected mode support, drivers for up to 38 COM ports and Novell's IPX services, and lots more...

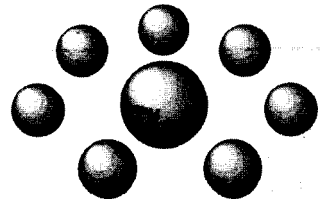
It's ROMable and very compact (about 16K code, 6K data), making it ideally suited for Embedded Systems.

RTKernel is well-documented and easy to use. All hardware drivers always come with source code: kernel source code available at extra charge. No run-time royalties.

Join thousands of satisfied customers!

In North America, please contact:
 On Time Marketing
 88 Christian Avenue Setauket, New York 11733 USA
 Phone (516) 689-6654 Fax (516) 689-1172
 BBS (516) 689-6285 FaxFacts (516) 689-6315 CIS 73313,3177

From other countries, please contact:
 On Time Marketing
 Karolinenstrasse 32 · 20357 Hamburg · GERMANY
 Phone +49-40-437472 · Fax +49-40-435196
 CompuServe 100140,633



- Use RTKernel for:
- ▶ process control
 - ▶ data acquisition
 - real-time simulations
 - background processing

Libraries: \$495
 Source Code: add \$445

On Time MARKETING

Professional Programming Tools

Free demo disk! Request Info Kit I
 Internet: http://www.on-time.com
 ftp.on-time.com
 info@on-time.com

lock like Lego blocks should you need to expand or interconnect packaged end items in the future.

THE POWER FACTOR

Strictly speaking, power supplies are beyond the scope of this article, but finding the right power source for your PC/104 project can be difficult.

Not only do you need to provide the ISA bus voltages (+5 VDC, ± 12 VDC) but you may require -5 VDC for **negative-bias**-voltage components and 3.3 VDC for some newer green CPUs or expansions cards. The input is usually from a direct current source, which may vary over a wide operating-voltage range and have little or no regulation.

Fortunately, several of the Consortium companies offer power supplies that conform to the PC/104 form factor and offer up to 60 W of power to the PC/104 bus. Input voltages usually range from 9 VDC to over 60 VDC with about an 18-V spread per range.

Other small form factor supplies are available with transient suppression and operating temperature ranges designed specifically for vehicles and aerospace applications. Nonetheless, you may need to design small **onboard** regulators for cards that require 3.3 VDC or other non-standard values.

BLASTOFF

The chassis reviewed in this article are well tested—they have been used extensively on oil drilling ships; submersibles; military tracked vehicles; delivery trucks; orbiting spacecraft; aircraft such as helicopters, F-16 fighters, and the Space Shuttle; and a host of autonomous rovers, overhead cranes, and robotic manipulators.

While this review is not exhaustive, it highlights some popular off-the-shelf packaging solutions. No longer is the PC/104 module simply an adjunct to a core processor motherboard. Stand-alone systems entirely based on PC/104 modules proliferate as virtual instruments, PLCs, data acquisition, data logging, and SCADA systems. They may soon show up as point-of-sale terminals, miniature file servers, and a host of factory floor or industrial PC platforms.

WRAP UP

A variety of chassis systems and enclosures from PC/104 Consortium companies eases the task of creating the proper setting for your system. Cost-conscious canned packages and a wealth of modular chassis components give the **flexibility you** need to make for your system.

In fact, the key to PC/104 is found in its flexibility. You can mix ISA cards with a PC/104 stack. You can mix and match subsystem layout and packaging because of the small form factor and connectivity of

the PC/104 modules. You can source components from any of the 150 companies currently supporting the standard. You can select software from the PC world or vendors of PC-compatible, real-time operating systems.

The greatest flexibility, however, is in the allocation of your development budget. With a PC/104-based solution, you can apply more dollars to application refinements instead of basic hardware or software building blocks, thereby speeding your time to market. PCQ.EPC

Dave Cox has over 30 years experience in the electronic industry where he has worked in engineering, R&D, sales, and corporate management. His particular interests lie in systems design and marketing.

Paul Olsen has worked in sales and management for over 13 years. He has experience with companies in the embedded controls industry and as an IBM business partner. Paul may be reached at (801) 483-1533.

SOURCES

Micro/Sys, Inc.
3447 Ocean View Blvd.
Glendale, CA 91208
(818) 244-6600
Fax: (818) 244-4246

The paws Corp.
1214 Wilmington Ave.
Salt Lake City, UT 84106
(801) 483-1523
Fax: (801) 483-1533

Kinetic Computer Corp.
270 3rd St.
Cambridge, MA 02142
(617) 547-2424
Fax: (617) 547-7266

Enclosure Technologies
256 Airport Industrial Blvd
Ypsilanti, MI 48198
(313) 481-2200
Fax: (313) 481-0557

Digital-Logic AG
Nordstrasse 4F, CH-4708
Luterbach, Switzerland
4165415336
Fax: 41 65 42 3650

PC/104 Consortium
P.O. Box 4303
Mountain View, CA 94040
(415) 903.8304
Fax: (415) 967.0995

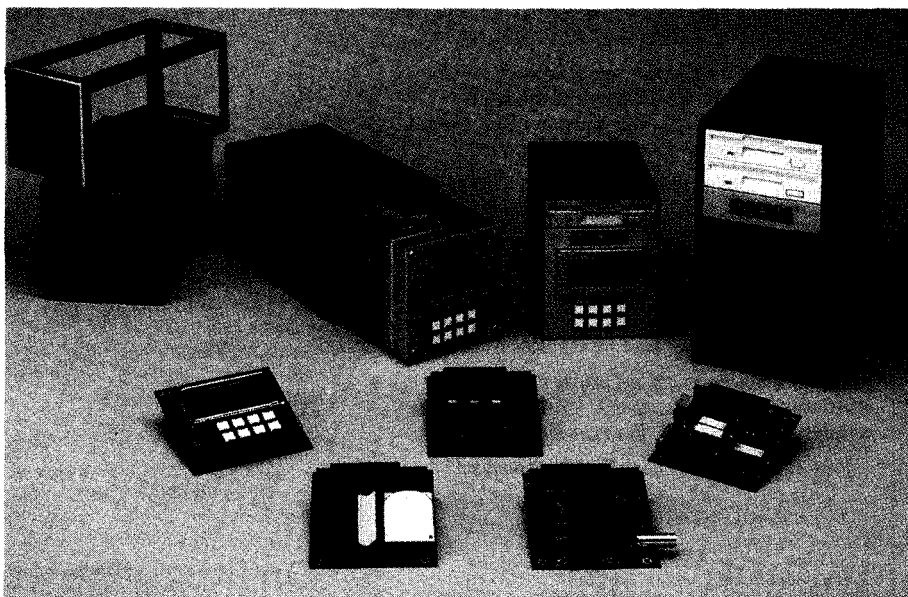


Photo 6: Enclosure requirements vary greatly. In some cases, an open-rail system (upper left) is sufficient. Other times, a fully enclosed system (center) or minitower with bays for various drive components (right) is more suitable. Regardless what setup you choose, operator interface panels are often a requirement.

IRS

4 19 Very Useful
420 Moderately Useful
42 1 Not Useful

Small Displays for Embedded Systems

This month, Russ points out smaller display options available to the embedded system architect. He gets into the nitty-gritty of how designers interface these small, non/PC-compatible units to their embedded PC designs.

Last issue, I mentioned we'd look into packaging embedded PCs. However, it's taken longer than expected to collect information on all the available options. So instead, I'll jump ahead to another important topic: displays for embedded systems.

I don't plan on making this an unabridged treatise on all sorts of display technology. Rather, I'll point out the main options in smaller displays and get into nitty-gritty practical methods for applying these smaller, nonPC-compatible units.

CRTS IN EMBEDDED SYSTEMS

The most obvious display choice for your embedded system is a conventional CRT monitor. While CRTs suffer from many shortcomings, they are readily available, inexpensive, and fully supported by PC software.

However, the list of negatives you must consider is long and extremely critical in some applications. CRTs are:

- large and bulky
- fragile
- subject to dangerous implosion and broken glass
- affected by strong magnetic fields
- capable of releasing toxic chemicals when broken
- lacking in brightness and crispness for certain tasks
- a potential health hazard due to EMI fields
- subject to attracting dust and dirt by high voltages
- relatively short-lived

Manuf.	Model	Pixels (HxV)	Power	Dim (WxHxD)
Planar	EL240.64	240 x 64	N/A	180 x 65 x 19
Planar	EL4836LP	276 x 128	4.8 W	177x99x15
Planar	EL4737LP	320 x 128	4.8 W	200 x 98 x 15
Sharp	LJ320U21	320x240	8 W	179 x 149 x 34
Planar	EL320.256	320 x 256	5.8 W	130x110x31
Sharp	LJ320U27	320 x 256	4.5 W	130x110x33
Planar	EL512.256	512x256	5.9 W	233 x 136 x 17
Planar	EL6648MSS	512 x 256	18 W	260 x 145 x 35
Sharp	LJ512U27	512 x 256	7 W	229 x 149 x 34

Note: Dimensions are overall outside dimensions in mm.

lab/e 1: The General Digital Embedded Display Controller chip drives all the EL displays shown in the table. Set a jumper to select the one you want.

In some applications, these factors are not that important, but in others, they can be the driving force in selecting a suitable display technology.

If a CRT works for your application, there is little I need to say other than consider yourself lucky! But, if you face the constraints typical of most embedded system applications, read on to see what alternatives are available to you.

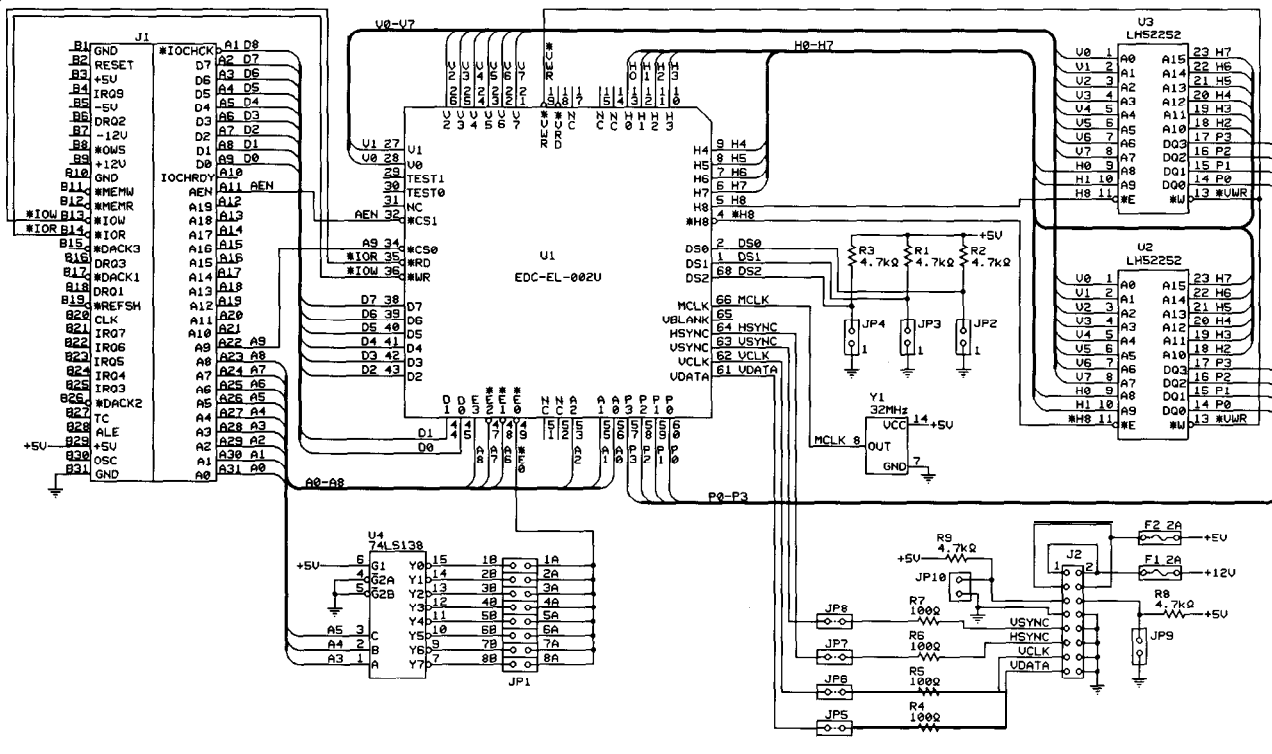
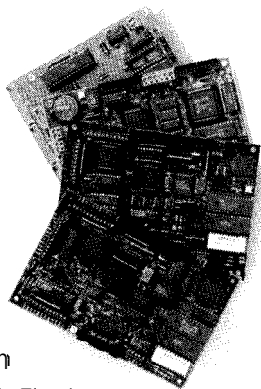


Figure 1: The **EDCELO2U** Embedded Display Controller chip *requires little more than MM, configuration jumpers, decoding, and an oscillator to directly drive a display.*

JAM
Packed

Embedded Controllers

- ◆ A/D inputs, 12-bit accuracy
- ◆ Analog outputs
- ◆ Relay control
- ◆ Counter/Quadrature encoder inputs
- ◆ Buffered RS-232/485 serial ports
- ◆ Operator interface via keypad and LCD display
- ◆ Program using a PC
- ◆ 512K program, 512K data memory
- ◆ 5V only operation
- ◆ Built-in BASIC supports all on-card hardware
- ◆ Floating point math
- ◆ From \$195 in 1's



REMOTE™
PROCESSING
The embedded control company

Call for more information and **FREE** Catalog of embedded controllers.

Ph: 303-690-1588, Fx: 303-690-1875

FLAT-PANEL DISPLAYS

Once we move from CRTs, we are naturally led to flat-panel technology. There are a number of competing technologies in the marketplace. In order of popularity, these are:

- Liquid Crystal Displays (LCD)
- Electroluminescent (EL)
- Gas Plasma (Plasma)
- Vacuum Fluorescent (VF)

Each technology has its own advantages and disadvantages, but the LCD technology, more than any other, has advanced significantly in recent years. As a whole, it offers the greatest number of advantages.

For the longest time, LCDs had one majordisadvantage: limitedoperating (and storage) temperature range. However, even this characteristic has been improving steadily.

The new active-matrix panels have increased the operating temperature range to 040°C. This range is very close to the 045°C achievable with EL and other technologies. I'll concentrate on LCD and Et displays since they represent the bulk of the

Base	Function
0	x-tow-Pointer Address
1	x-High-Pointer Address
2	y Pointer Address
3	Display enable
4	N/A-Reserved
5	Data Register (No Increment)
6	N/A-Reserved
7	Data Register (Auto Increment)

Table 2: The Embedded Display **Controller chip uses only eight registers (addresses) for communication.**

market and are the most rapidly developing and improving embedded displays.

Due to their brightness, ruggedness, and relatively low cost, VF displays were popular for a while in outdoor applications such as gasoline pumps and vending machines as well as in high-shock situations like elevators and automobiles.

However, now Et and even LCD technologies are giving VF a run for their money in these applications. (For more information on VF, see "Embedded Techniques," INK 32-33.)

Plasma displays were quite popular in smaller, character-only formats for many of the same reasons that VF displays were used. However, when applications shifted to larger color graphics displays, plasma panels have not been able to keep pace.

Before going on, I should point out that if your application requires a display that meets one of the popular PC video standards (i.e., CGA, EGA, VGA, etc.), many embedded system CPU boards and add-on video controller cards support flat-panel displays directly just as they do CRTs. If this is your case and you require only a single—or at most double-display(s), you can switch to flat-panel technology quite simply.

Remember though, such displays are not cheap and are overkill in many applications. This is where the smaller displays come in. All you really need to pay attention to are some unique interfacing requirements.

SMALL DISPLAYS FOR DISTRIBUTED INFORMATION

While desktop computing with PCs almost invariably involves some sort of CRT display, embedded applications present different display requirements.

Consider, for example, the embedded PC that drives all those small pricing dis-

plays on supermarket shelves. While the main system console might be a conventional monochrome or color CRT, the working end of the system is a vast array of small one-or-two-line, dot-matrix-character LCDs.

This type of display requirement is found in manufacturing, process control, inventory, shipping, security, and other settings. Yet, the interface between an embedded PC and a cluster of small LCDs might not be readily apparent.

If just one small display were needed, it is possible to use the PC's printer port to drive it. As Ed Nisley points out in *INK 8*, his LCDTEST program demonstrates how to drive an LCD with no additional hardware.

However, when a large number of displays is required—with each display having its own unique information—something more is needed.

Probably the simplest and most robust manner of supporting this need is to add a small amount of intelligence to each display. A single-chip microcontroller, such as one of Microchip's PICs, the Motorola 68HC705, or the Atmel AT89C105, fills the bill nicely. The controller's main function is to reduce the multiple parallel lines needed to drive the LCD to a more convenient asynchronous serial-communications format.

Robust communication between the embedded PC and intelligent displays scattered throughout a large store or factory calls for a noise-immune, differential, multidrop approach such as RS-485. While RS-485 is typically limited to only 32 receivers per bus, the use of new high-impedance receivers and low-capacitance cable extends this greatly!

Maxim's MAX1482 and 1483 transceivers present only one-eighth the load of conventional RS-485 devices and permit up to 256 nodes per bus. The MAX1487 runs faster—up to 2.5 Mbps—and still supports 128 devices per bus. These devices open up many possibilities for RS-485 multidrop applications.

With such an approach, each transmission begins with the address of the display receiving information and is followed by the information itself. Each intelligent display subsystem listens for its unique address and, when detected, grabs any data that follows and acts on it.

In this way, a single serial port on the embedded PC is able to communicate with

IF YOU DO
FUNCTIONAL TESTS
 YOU NEED
HOT SWAPPING ELECTRONIC EXTENDERS

For PCI, ISA, VXI, EISA, VESA, Micro Channel & NuBus. Custom Designs available.

Electronic ISA Extender / PC Mini Extender



Electronic PCI Extender / PCI Mini Extender



Insert/Remove Cards **With PC Power On!**
 Save Time Testing And Developing Card!
 Save Wear On Your PC From Rebooting
 Adjustable Overcurrent Sensing Circuitry
 NO Fuses, All Electronic For Reliability
 Single Switch Operation W/Auto RESET
 Optional Software Control Of All Feature
 Breadboard Area For Custom Circuitry
 And More...

Full line of passive extenders:
 PCI, ISA, VXI, EISA, VESA,
 Micro Channel & NuBus

Passive PCI Extender Passive EISA Extender



Passive MC32 Extender Passive ISA Extender



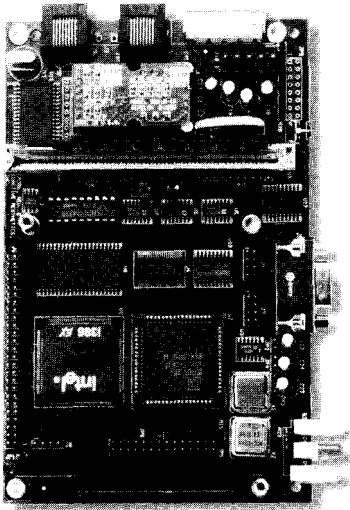
AZ-COM

3343 Vincent Rd., Ste. D,
 Pleasant Hill, CA 94523

TEL: 1-800-209-2418
 FAX: 510-947-1900

CALL OUR 24 HOUR
 FAX ON DEMAND SYSTEM
 510-947-1000

Embedded PC with Built-In Networking



The **CA386-N1** is an embedded PC (386EX) with a built-in LONWORKS® network interface.

LONWORKS networking technology supports a variety of physical media, including twisted pair, powerline and RF, and is supported by over 150 vendors. The **CA386-N1** features:

- **25 MHz** 386EX
- **512K Flash, up to 4MB RAM**
- **8/16 bit PC/104 Bus**
- **On-board I/O Includes 24 Digital I/O, 2 Serial Ports, and SPI**
- **DOS Preinstalled**
- **Download and Flash Utilities Included**
- **'C' API for Networking Available**

The **CA386-N1**, like the rest of Coactive's line of networked controllers and software tools, is powerful and easy to use, and can help ease your move to Intelligent Distributed Control.

Call us today to
discuss your application

1 (800) 699-8090

 **COACTIVE**
AESTHETICS, INC.

4000 Bridgeway, Suite 303 • Sausalito, CA 94965
Fax: (415) 289-1320 • Internet: coactive@coactive.com
http://www.coactive.com

a large number of intelligent LCDs. The micro might also handle local functions like alternating between unit and item price or scrolling an advertisement on the display's second line. Furthermore, communication loading might be reduced by storing fixed or downloadable canned messages within each microcontroller.

Getting power to remote displays is another problem. One straightforward solution would be to simply run another wire within the communication cable. If you choose this approach, I recommend a separate twisted pair for the power and that you not run regulated 5 V over long distances. Instead, run unregulated voltage (perhaps tapping off the PC's 12-V supply) and regulate it locally at each display or microcontroller site with an inexpensive 78105 linear regulator or a zener diode.

When only a few displays are on each bus line, you might take advantage of the extremely low-power requirements of the LCD and microcontroller (operating at a very low clock rate) and steal the power from the RS-485 data lines themselves! A pair of diodes, each coupling power from one of the differential data lines to a common filter capacitor, can yield an inexpensive 3-V power source in some instances.

CONTROLLER FOR MIDSIZE GRAPHICS DISPLAYS

When your display requirements demand more than a simple, dot-matrix LCD character module but less than standard VGA capabilities, you might consider a midsize Et graphic display.

Planar and Sharp are today's leaders in Et technology and many of their units are conveniently interchangeable. Table 1 lists the features of many of the currently available models.

Et displays feature bright, crisp, clear images and a very wide viewing angle. While they require significantly more power than micropower LCDs, it is often in the range of only a few watts. Color Et displays are available but still somewhat in the development stage. You usually choose Et displays for monochrome applications.

Until recently the main drawback to using midrange Et displays was a lack of off-the-shelf controller chips and interface cards. Unlike displays for PC standards (e.g., VGA), midrange Et displays have

different timing requirements and are incompatible with PC BIOS drivers and application software. However, as you'll see, lack of conformity to PC display standards is both an advantage and a disadvantage. Let me explain.

When using a display that is supported by the PC's BIOS (and BIOS extensions located on the video controller card), writing programs to display information is simple because the display serves as the system console. High-level print statements direct output right to the display.

However, the designers of the original PC only envisioned one (or possibly two) system consoles displaying identical information and operating only one at a time. PC users are often familiar with the problems and limitations encountered when trying to use both a monochrome and a color CRT monitor on the same computer at the same time. Embedded systems, on the other hand, often require multiple displays, each presenting unique information.

Consider, for example, an intensive care unit where separate displays monitor each patient's condition continuously and simultaneously. In an operating room, a single embedded PC drives different displays (often of different size and resolution) of patient status details.

An air-traffic control room has an intelligent radar system in an embedded PC application which presents different information to different users on multiple displays.

Military vehicles include a cluster of small graphics displays, each presenting different information to specialized personnel.

There's an almost endless number of embedded system applications in which multiple, midsize displays driven by a single embedded PC is the most suitable and desirable solution.

Recognizing this, General Digital developed the EDCELO2U, an LSI Embedded Display Controller (EDC) chip, which supports the Planar and Sharp midsize EL displays. It can be customized and adapted to nearly any small or midsize display. The chip is unique in that it is much simpler to use than the more familiar CGA, EGA, and VGA video controller found in desktop PCs.

While the chip supports full graphics capability, it possesses no character generator. The firmware must therefore pro-

vide the character fonts. This choice, however, is also an advantage. Customizable font sets, icons, operator control buttons, gauges, and so on can be defined in software and sent to the controller, which handles display refreshing.

General Digital provides a number of support programs and software libraries which aid the user in displaying BIOS fonts, custom fonts, and user-defined icons with the embedded display controller chip.

Let's see what it takes to apply this chip to embedded display applications.

Figure 1 is the schematic of an EDC/PC evaluation board. This board plugs into any PC, XT, or AT bus slot and drives any of the Et displays shown in Table 1. The design includes a decoder chip and jumper block for setting any arbitrary base address for the controller. This card operates independently and in parallel as an auxiliary video controller with the conventional system display. In fact, any number of EDC boards (each assigned to a unique display) may be used simultaneously within one PC.

From the pinout of the 68-pin PLCC EDC-EL-002U chip, it is obvious that a number of signals are involved. However, they fall into four categories, so it's a very simple interface to work with. The categories are:

- interface to the microprocessor bus
- interface to the refresh fast SRAM
- interface to the Et display
- miscellaneous power, clock, and mode select inputs

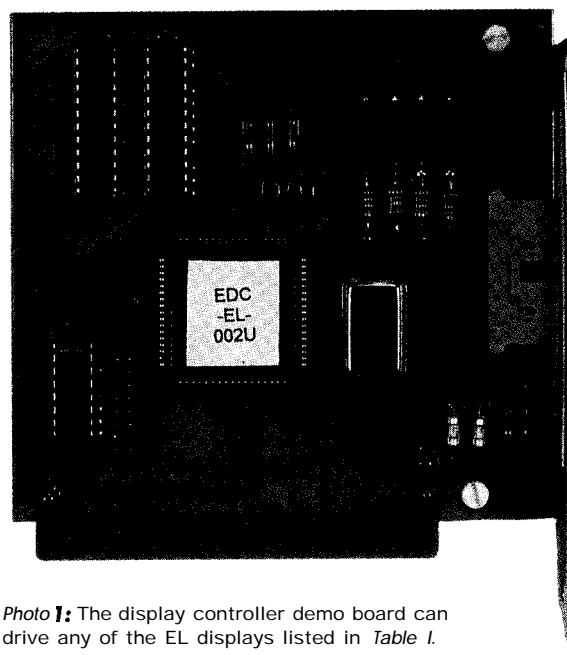


Photo 1: The display controller demo board can drive any of the EL displays listed in Table 1.

Communication with the embedded CPU—either directly or buffered via a standard bus interface such as XT, AT, STD, PC/104, and so on—is the job of the first group of signals. The *WR, *CS[0..I], *E[0-3], A[0-2], and D[0-7] lines make the display controller appear to the system as a bank of 8 byte-wide write only addresses.

Either I/O or memory-mapped address decoding and write signals may be used due to the small number of registers (addresses) involved. The multiplicity of chip select and enable lines reduces or, in some cases, completely eliminates the need for address decoders.

For instance, connection to a PC bus is possible with no external glue logic if a fixed address is acceptable. Or, as shown in the design of Figure 1, a single decoder chip and DIP switch or jumper block provides complete base address flexibility. This is particularly useful when a number of controller chips coexist in a single system, each at its own base address.

The big question becomes: How do you reference 10^4 – 10^5 pixels using only eight addresses?

The controller uses a novel pixel-pointer scheme in which a horizontal register (two bytes) and a vertical register (one byte) point to the display pixel being referenced. Four bits of attribute data are written to another data register to define the referenced pixel.

This referencing permits more than a simple on/off presentation. Each pixel may be individually and independently on, off, blinking, flashing, bright, or dim as desired. These attributes are useful for highlighting, alerting, and annunciating in many embedded display applications.

If an autoincrement data register is written to, the horizontal location is bumped to the next one after each pixel's data is written. At the end of a line of pixels, the new, full, horizontal, and vertical positions must be defined. It takes more time to describe what's involved than it does to write the code to accomplish this in either C or assembly language!



Your Complete x86 Solution!

Multitasking in Real and Protected Mode



smx? Full-featured, high-performance, preemptive kernel. Customized to x86 processors. Ideal for demanding applications. Real mode works stand-alone or with DOS. 16-bit, segmented protected mode works with *pmEasy16* or 286/1 DOS Extender? 32-bit, flat protected mode works with *pmEasy32* or TNT.™

Protected Mode Environment



pmEasy™ 16- or 32-bit protected mode entry, DPML services, application loaders, Periscope/32® and Soft-Scope® debugger support.

DOS-Compatible File System



smxFile.™ Full-featured file manager. IDE, floppy, and RAMdisk drivers available.

Dynamically Loadable Modules



smxDLM.™ Runs independent executables as tasks which may be downloaded or loaded from disk.

Networking



smxNet.™ TCP/IP stack. Fast UDP. Packet driver interface. NE2000, SMC, and SLIP drivers available.

Task-Level Debugging

4

smxProbe.™ Provides tracing and symbolic debugging. Works with or without code debuggers.

C++ Classes



smx++™ Class library built upon smx. Provides fully OOP-compatible kernel interface.

Extended Memory, Real Mode



smxEMS.™ Allows copying data between real memory and extended memory buffers or accessing extended memory via a window.

User Interface



smxWindows.™ Text windowing. Dresses up user interfaces.

Low Cost & Easy to Use



Royalty-free licenses. Supports common, low-cost C compilers and debuggers. Great manuals Source code available.

Reliable



On market 6 years. 100's of applications. Extensive error checking. 30-day free trial.

I-800-366-2491
mdi@earthlink.net

M I C R O D I G I T A L
ax 714-891-2363 Garden Grove, CA USA
<http://www.earthlink.net/~mdi>

OPTO-ISOLATED DATA ACQUISITION



-5KV optical isolation!
-Up to **1016 channels!**
- **12 bit A/D** conversion!

LAPTOP
DATA ACQUISITION

Simple to use
High speed
Long distance

FREE 'C' source code available

* OPTO-ISOLATED RS-232/RS-485 converter enclosed in a DB9 backshell. **(ID485-9C) \$52⁰⁰**

* ANALOG input module with 4 channels of 12 bit A/D conversion plus I/O base module with MPU. **(ID485-ADC & ID485-MOD) \$140⁰⁰**



Integrity Designs

800-450-2001 209 Grand, #403
Laramie, WY

AMX The Real-Time Multitasking Kernel

680x0, 683xx
50386 protected mode
iR3000, LR330x0

80x86188 real mode
i960[®] family
Z80, HD64180

NEW • DOS Compatible File System
• TCT/IP • 29K support

- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- InSight™ Debug Tool
- Comprehensive documentation
- No royalties, source code included

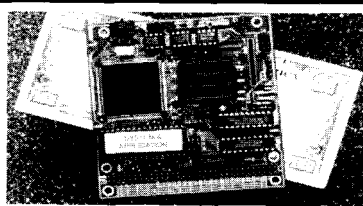
For a free Demo Disk and product description,
Phone: (604) 734.2796
Fax: (604) 734-8114



KADAK Products Ltd.
206 - 1847 West Broadway
Vancouver, BC, Canada V6J1Y5

#212

Power Miser Powerhouse!



5V at 100mA operating • SPI/I²C Port
(great for battery powered applications)

- PC/104 XT Engine
- * Realtime Clock
- Watchdog
- 512K or 2MB DRAM

- Flash ROM Module
- DOS in ROM
- EPROM

Only \$1 00/quantity 1000

vesta technology, inc.
303-422-8088 • FAX: 303-422-9800

#213

Table 2 identifies the eightwrite-only registers within the EDC.

The second group of signals interfaces the EDC chip to the refresh fast SRAM chip(s). For the smaller size displays which have fewer than 64k pixels, only a single SRAM chip is needed. Two chips handle all displays through 5 12 x 256 pixels.

The SRAM must be 25 ns or faster and is 4 bits wide. It is possible to use x8 SRAMs with an external multiplexer. The vertical V[0-7] and horizontal H[0-7] address lines connect to any unique set of SRAM address lines.

The exact assignment is not important since whatever location a given pixel is written into is also where it is read from. In addition, the H8 address line is available in both true and complemented form and serves as a chip select for the two SRAMs.

The SRAM serves as a dual-ported refresh memory for the display. As such, it is the heart of the EDC. A bidirectional 4-bit pixel data bus P[0-3] carries the refresh pixel data between memory and the EDC. During a portion of each pixel clock period, the SRAM is addressed by the EDC to obtain the current pixel data sent to the display for refresh.

During another portion of each pixel period, the SRAM is available for updating information sent by the host CPU. CPU data is latched and buffered within the EDC, so there are no wait states. You must guarantee that you don't write to the EDC any faster than the pixel clock rate. A single *VWR write-control line determines if the SRAM is being read or written.

The third set of signals consists of the four lines that connect to the EL display itself: HSYNC, VSYNC, VCLK, and VDATA. The signals contain all the data and timing information needed to drive the display. Connection is usually by simple ribbon cable.

The final set of signals includes V_{cc} (5 V), GND, MCLK (master 32-MHz clock from a crystal oscillator module), and three display-size select inputs. In some cases, if you have a suitable clock signal already available, the crystal oscillator module can be eliminated. The EDC chips can be customized to accommodate the clock rates commonly found in existing designs.

Having the display(s) totally independent of the system console simplifies a lot of things. There's no need for display driver software, compatibility with BIOS or DOS,

limitations to a single display, incompatibility of display modes, and other such complexities. You simply decide what you want to display and do it!

Photo 1 shows the completed EDC/PC demo board available from General Digital. However, I might point out that this EDC chip interfaces to nonPC microcontrollers like the 8051, 68HC11, and PICs just as easily. It is ideal for creating stand-alone, intelligent display subsystems.

SUMMARY

In this installment, I've tried to give some insight into techniques for interfacing smaller displays to embedded PCs. Hopefully, this discussion has removed some of the mystery from using flat panels with nonstandard display formats.

In future issues, I'll return to discuss other aspects of interfacing and programming displays of all sorts, for they are and probably always will be the foremost user interface we have. APC.EPC

Russ Reiss holds a Ph.D. in EE/CS and has been active in electronics for over 25 years as industry consultant, designer, college professor, entrepreneur, and company president. He may be reached at russ.reiss@circellar.com or 70054.1663@compuserve.com.

SOURCES

RS-485 bus interface chips
Maxim Integrated Products, Inc.
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737.7600
Fax: (408) 737.7194

EL Displays
Planar Systems, Inc.
1400 NW Compton Dr.
Beaverton, Oregon 97006
(503) 690-1100
Fax: (503) 6457024

Sharp Electronics Corp.
Microelectronics Group
5700 NW Pacific Rim Blvd., M/S 20
Camas, WA 98607
(360) 834-2500
Fax: (360) 834.8903

Embedded Display Controller chip
General Digital Corp.
198 Freshwater Blvd.
Enfield, CT 06082
(860) 741.7171
Fax: (860) 741.7071

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

DEPARTMENTS

72 Firmware Furnace

78 From the Bench

86 Silicon Update

91 ConnecTime

Vid-Link Characters

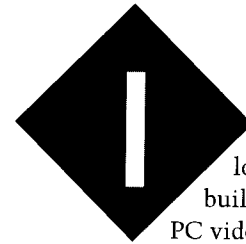
Part 2: Bits to Dots



After last month's discussion of the Vid-Link hardware, Ed sets out to squeeze the necessary firmware into an impossibly small space. Find out what tricks he has to pull to do it.

FIRMWARE FURNACE

Ed Nisley



Let's face it—no longer can you build a competitive PC video interface from a handful of chips and a yard of solder. Monochrome text begat color text, text begat bitmapped graphics, and plodding CPU bit-twiddling mutated into accelerated, true-color, full-motion video. Your monitor now presents two million simultaneous dots, each sporting a different color.

Building the hardware behind that video connector requires custom LSI, overlapped and pipelined memory, RF-terminated signal lines, and convoluted device drivers that recognize common benchmark programs.

We're not watching TV Typewriters any more!

The Vid-Link HCS video interface I introduced last month remains a TV Typewriter at heart because analog TV hasn't kept pace with the computer revolution. If you plan to watch an HCS status channel, you'll see it on a plain old TV with plain old 525-line NTSC video. S-video connectors do not transform a TV into a computer monitor.

As an aside, the local Best Buy home electronics store offers 25" stereo TVs for about \$600 while 21" computer monitors hover near \$2000. Even allowing for their different markets, that price differential puts far better hardware into the monitor. Spend some time watching high-bandwidth signals on TV, and you'll see what I mean.

This month, we'll take a look at the Vid-Link's character generator and serial network interface. Those of us writing microcontroller code confront

cramped memory, peculiar I/O, and stringent timing as a matter of course. Let's continue to do the best we can with what we've got.

MAPPING THE CHARACTERS

Vid-Link started life as the Image-Wise project, a grayscale display with no need for character output. Thus, if we want characters, we must draw them dot by dot from a bitmap stored in the same EPROM as the program code. That seems reasonable until you work through the numbers.

The smallest practical characters for a TV display fit into an 8 x 8 cell, with adjacent cells abutted tightly against each other. The next "nice" size uses a 16 x 16 cell, which allows considerably finer details. Using the hardware's grayscale capability generates nicely defined, surprisingly beautiful characters.

Steve and Ken agreed that Vid-Link needed the entire IBM PC CGA and VGA font to provide the line-drawing characters so beloved on status screens. Those characters lie outside the usual 128 ASCII codes, requiring 256 characters in the font table.

Harsh reality intrudes at this point, though. Each beautiful, antialiased

small-font character occupies 64 bytes, thus 256 of them require a 16-KB font table. The spectacular large font, four times larger, requires 64 KB. Fitting both into an 8-KB EPROM, along with the code that plunks them on the screen, would be a nice trick.

I opted for a low-budget solution by discarding the grayscales. Each dot becomes a single bit, reducing the small characters to eight bytes apiece. Without brightness information, the characters use a single brightness level, antialiasing goes out the window, and we're back to a TV Typewriter.

Anyone with even a smidge of typographic experience knows that you shouldn't generate a large typeface from a smaller one by simply magnifying the dots. Despite that, the code creates large characters by repeating each small font dot once in each direction.

The farther away you get, the better they look. Picture-in-picture windows viewed from across the room seem to be about right.

With those concessions to reality, the font table soaks up exactly one quarter of the EPROM: eight bytes per character times 256 characters fills 2 KB with dots. I experimented with

omitting some characters and remapping the remainder through a lookup table, but the overall savings didn't justify the effort.

Generating the font posed no problem. I rummaged around in my heap of font diskettes and CD-ROMs to find a freeware binary file of something that looked a lot like generic CGA characters. The original CGA font had only 128 characters, but the EGA and VGA included all 256 in their CGA compatibility modes.

I wrote a little REXX routine that ate the binary file and spat out the commented assembler source code shown in Listing 1. This may seem somewhat roundabout, but it eliminated the hassle of dealing with a separate binary file each time I created a new program hex file. I also twiddled a few of the characters for better looks.

With font in hand, we can now look at the gyrations required to drop dots on the screen. In case you lost count, a 2-KB font table leaves 6 KB for the rest of the program. The other HCS Links use Micro-C's Compact memory model, the smallest one weighing in at 16 KB. How would you squash that code by a factor of three?

No, you can't switch to C++ and gain the advantages of object orientation.

DERIVING THE DOTS

Although the font table contains only on-and-off information for each character dot, all is not lost. Vid-Link responds to the usual set of ANSI commands, one of which sets the video attributes for subsequent output. With very little effort, you may display text containing normal, bright, or reverse characters.

Normal characters appear as half-bright dots, hex 80, on a black background. Bright characters use full-bright dots, hex FF, also against a black backdrop. The reverse video ANSI command swaps the current intensities, giving black characters on either a half- or full-brightness background.

Even though the font table contains only one bit per dot, drawing a single, small character requires 64 byte writes that completely replace the previous buffer contents. There's no optimiza-

Listing 1—The Vid-Link's 8 x 8 font includes all the standard IBM PC CGA characters and occupies 2048 bytes of EPROM. This excerpt from the assembler source file shows a few typical characters.

```
* Font file      8X8FONT.BIN
* 256 chars
* 8 bytes/char
* 2048 total bytes

* Character 01
DB $7E      * ██████████
DB $81      * ████
DB $A5      * ████
DB $81      * ████
DB $BD      * ████ www ████
DB $99      * ████ ████
DB $81      * ████
DB $7E      * ██████████

* Character 41
DB $30      * ██████
DB $78      * ██████
DB $CC      * ██████
DB $CC      * ██████
DB $FC      * ██████
DB $CC      * ██████
DB $CC      * █ w █ W
DB $00      *
```

tion possible because each dot position can have any of the three values. We can't just update a single byte that produces an entire character, as we could if the video interface sported a hardware character generator.

The large font presents an even more formidable problem; each character requires 256 writes!

Michael Abrash, of code-optimization fame, reiterates that "There's No Such Thing As The Fastest Code." You can always make a given chunk of code go faster, sometimes by direct opcode twiddling and other times by rethinking the entire problem. His Game-of-Life contest shows many different paths to a well-defined goal.

With that in mind, the code in Listing 2 extracts font bits, plumps them into bytes, and tucks a small character into the video buffer. Listing 3 performs the same tasks for large characters. I've tweaked these routines several times, but, well, as Abrash said, "TNSTATFC."

Writing a single 8 x 8 character takes 1.2 ms, about 19 μ s per byte. Writing one large 16 x 16 character, touching four times as many bytes, uses 2.8 ms, a mere 2.3 times longer. As you look through Listing 3, you'll find a middle loop duplicating the rows and an unrolled inner loop writing two successive horizontal bytes.

Because the Vid-Link hardware has only one data path from the video RAM to the D/A converter, writing bytes into the buffer during the active video time produces very visible hash. The standard solution restricts buffer writes to the vertical blanking interval, but, as you saw last month, the 8031 CPU has quite a lot to do while generating those pulses.

My initial sketches for this project showed quite clearly that it must write characters during the active video time, pretty or not. Unable to navigate a course between the Scylla of sparkles and the Charybdis of a blank screen, I faded to black.

Those few milliseconds per character loom large in a 16.67-ms video field. When the Vid-Link receives a string from the HCS Supervisory Controller, it blanks the screen before writing the first byte and holds it off

until after the last character hits the buffer. Even a short string blanks the screen for several fields.

You can now see why I didn't implement the ANSI character blinking command! Simply writing a few characters occupies nearly the entire visible part of the video field, leaving no time to view them. In effect, the whole field would blink, although not in any desirable manner.

Now that we can write characters on the screen, where do they come from?

STUFFING A 20-BYTE BAG

The original LCD-Link network status display supported LCD panels with up to four lines of 20 characters each, the maximum for an HD44780 controller. The longest practical message, including ANSI cursor control

strings, might be 100 bytes. Because the LCD-Link board has an 8-KB RAM chip, I allowed messages up to 256 bytes.

Because the Vid-Link hardware supports 28 lines of 30 characters, you might need four or five such messages to update the entire screen. That seems reasonable until you recall that the ImageWise board doesn't have any external RAM other than the video buffer and that you can't update the screen until you verify the message against its checksum.

Oh, yeah, remember those sparkles?

The 8031's 128-byte internal RAM already contains every variable, bit, register, and stack byte used by the firmware. When I finished adding up all those Tiny memory model items, I had about two dozen bytes left over for serial buffers. It takes a lot of 20-char-

Listing 2—Writing an 8 x 8 character cell requires about 1000 instruction cycles or 1.2 ms. Converting fontable bits into bytes and writing them into the video buffer occupies most of that time.

```

MOV R2,#CG_FONTHEIGHT /* we do a whole character here */
?CWC_1
MOV DPH,R5 /* fetch font byte */
MOV DPL,R6
CLR
MOVC A,[A+DPTR]
MOV R0,A /* save the bits */
MOV R4,A
INC DPTR /* tick the font pntr (16 bits) */
MOV R5,DPH
MOV R6,DPL

MOV DPH,#HIGH_VIDEO_BUS /* select video buf for writes */
JB B_CGDoubleSize,?CWC_double

/* --- handle single-size chars */
MOV R3,#CG_FONTWIDTH /* set up for all eight bits */
MOV DPL,R7 /* starting at this address */
?CWC_2
MOV A,R0 /* pick up the font bits */
RLC A
MOV R0,A /* save bits for next pass */
MOV A,CGLowIntensity /* figure out dot intensity */
JNC ?CWC_Off1
MOV A,CGHighIntensity
?CWC_Off1
MOVX [DPTR],A /* write the dot */
INC DPL /* step to next dot */

DJNZ R3,?CWC_2

INC P1 /* step to next buffer line */
DJNZ R2,?CWC_1

```

Listing 3—Even though a 16 x 16 character cell has four times more dots than an 8 x 8 cell, this code writes them in about 2.8 ms rather than the 4.8 ms you'd expect. The savings come from duplicating horizontally with in-line code and vertically with a middle loop that avoids much of the outer loop setup.

```
?CWC_double
    MOV    R1,#2                /* need two lines per font line */

?CWC_twolines
    MOV    R3,#CG_FONTWIDTH /* set up for all eight bits */
    MOV    DPL,R7              /* at this address */

?CWC_3
    MOV    A,R0                 /* pick up the font bits */
    RLC    A
    MOV    R0,A                /* save bits for next pass */
    MOV    A,CGLowIntensity /* figure out dot intensity */
    JNC    ?CWC_Off2
    MOV    A,CGHighIntensity

?CWC_Off2
    MOVX   [DPTR],A            /* .. write it */
    INC    DPL
    MOVX   [DPTR],A            /* ... twice! */
    INC    DPL                 /* step to next dot */
    DJNZ   R3,?CWC_3

    INC    P1                  /* step to next buffer line */
    MOV    ?R0,R4              /* restore the font bits */

    DJNZ   R1,?CWC_twolines /* duplicate the line */

    DJNZ   R2,?CWC_1          /* repeat for all font lines */
```

acter messages to fill that TV screen, even should Steve agree to such a hobbled interface. Somehow, the network messages must wind up in the video buffer because they simply won't fit anywhere else.

The HCS network runs at 9600 bps, producing a new byte every millisecond. A single video field lasts 16.67 ms. If I could buffer at least 16 characters in internal RAM, perhaps I could transfer them to the video buffer during the vertical retrace without sparkling.

A single video line holds 256 bytes, enough for a complete network message. The visible part of the display occupies only 224 lines. So, finding an unused, invisible line wasn't difficult.

Flip back to last month's Photo 1 (INK 66). Although the 8031 has little spare time during the equalizing and vertical sync pulses shown on the left third of the top trace, those perfectly blank lines in the middle third seem ripe for the plucking. As it turns out, I found just enough time to transfer all the data before the first character row.

Figure 3 in that column itemized the various events during the Field 1 vertical retrace interval. The Video Data Pump routine gets control in Line 12 and captures the next eight horizontal sync pulses. During that time, it moves up to 20 bytes of data between internal RAM and the video buffer.

The top of a standard video picture begins in Line 21 and continues until Line 262, a total of 242 lines. Vid-Link, however, displays only 28 rows of 8-line characters, occupying 224 lines. I pushed the start of the first character row down to Line 29, giving more time for the data pump and, not coincidentally, moving the characters out from behind the bezel on monitors with lots of overscan.

How do you stuff 256 bytes into a 20-byte bag? It's easy-glue on a bigger bag!

STRIPPING AND PUMPING DATA

Most of the other HCS Links using my code share a common set of network and serial interface routines. Buffering messages, verifying the ad-

Flexibility Rules!!

With product life shortened from years to months, you're dead if your designs can't adapt for new conditions. Field Programmable Gate Arrays let you change your product's circuitry on-the-fly, providing the flexibility you need to succeed. But how can you quickly learn FPGA design techniques?

The **epXkit** is your one-stop solution that contains everything you need to do FPGA design right now:

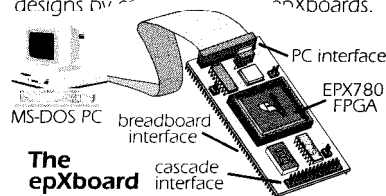
Our **PLDSHELL** text shows you how to design everything from simple combinational logic and state machines all the way up to a complete microcomputer with an FPGA.

The **PLDSHELL** also allows you to enter your designs as truth-tables, Boolean equations, or state-machine descriptions.

The **simulator** lets you examine how your design works before loading it into the FPGA hardware.

Our **platform** provides you with a simple, modular FPGA platform that can interface with other chips and expands to hold larger designs.

The **epXkit** is easy to use. First, you describe and simulate your design using the **windowed PLDSHELL** Interface Next, attach the **epXboard** to the printer port of your PC and download your circuit into the EPX780 FPGA. Then apply actual signals and see how your circuit does in a real-world test. You can test even large designs by using the **epXboards**.



The **epXkit** gets you up-to-speed fast. Start on your own design right away by modifying one of our existing circuits. Or go through the set of exercises in the **FPGA Workout** and get a complete understanding of FPGA design.

The **epXkit** lets you try out more design ideas with less effort. No more untangling wire-wrap or patching printed circuit boards. Just reprogram the static RAM in the FPGA and you're ready to try again. That will put a smile back on your face from 9-to-5!

epXkit
\$165.95
XESS Corp.

USA (800) 549-9377 Mastercard and
INTL (919) 387-0076 VISA accepted

Check our Web site for FPGA tutorials!
<http://www.xess.com>

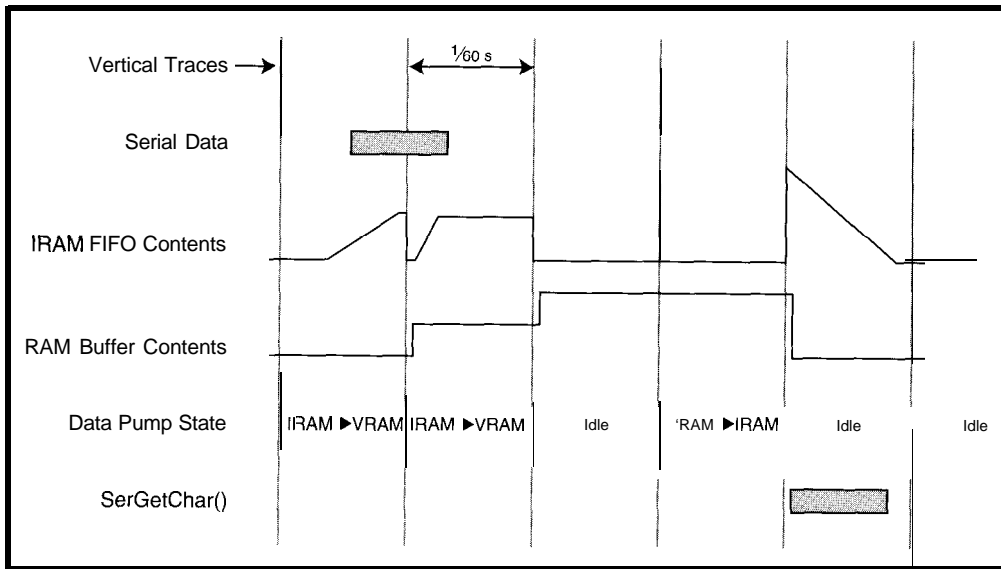


Figure 1—Messages from the HCS network can arrive at any time. A 20-byte internal RAM FIFO holds the bytes until the data pump moves them to a 255-byte buffer in video RAM. When the message ends, the pump returns the first bytes to the FIFO for the `SerGetChar()` routine.

This elaborate mechanism has two obvious disadvantages. First, `getch()` doesn't report a new character until about three vertical syncs after the message ends.

Worse, the firmware can't accept a new message until it finishes processing the old one, which, as we saw above, can take quite a while.

Because the Vid-Link receives every message for every network node, it would spend most of its time shuffling irrelevant bytes into and out of the video RAM buffer. To combat that, the serial interrupt handler decodes message headers and discards messages addressed to other nodes. The data pump sees only the text part of messages addressed to the Vid-Link, thus reducing the number of bytes going into the buffer.

The interrupt handler also accumulates a checksum and flushes invalid messages by simply resetting a few counters and pointers. Ken reports that checksum errors happen very infrequently, but keeping junk off the status display seems like a very good idea!

What happens if the Supervisory Controller sends a second message to the Vid-Link before it finishes displaying a previous one? I agonized over this problem for quite a while before deciding that the Vid-Link

must ignore the second message. There just wasn't enough internal RAM for a second FIFO and its control variables. With the one-and-only FIFO tied up handling a previous message, subsequent messages simply vanish into the ether.

address and checksum, and parsing the contents all depend, in part, on the common hardware design of those Links.

In an ideal world, firmware divides into stacked layers. The bottom layers handle hardware interfacing details, middle layers provide fundamental functions, and the upper layers should be recognizable to application programmers. That's what you read in software magazines, anyway.

Much of that goes by the board when the requirements also include a 6-KB total program space, cramped internal RAM, nearly inaccessible "bulk" data storage, and stringent timing. You can force-fit standard code onto screwball hardware only when another requirement cuts you some slack. Most often, that's not the case.

I took a long, hard look at how I'd done the serial and network interfaces in the other Links. Then, I took a long, deep breath and started from scratch. I'm a big fan of reusable code, but only when reuse makes sense for the project at hand. Reused code that doesn't do the job makes nobody happy!

Figure 1 shows how the timing works for a short message. The internal RAM FIFO holds arriving bytes until the next vertical retrace, when the data pump transfers them into the video RAM. The code

simply moves raw ASCII bytes from the serial port without converting them into dots.

When the message ends, the data pump transfers the final chunk and goes idle. The `SerGetChar()` routine, which has been waiting for precisely that event, switches the data pump's direction and tells it to begin transferring the message from video RAM back to the FIFO.

After the next vertical retrace, `SerGetChar()` returns the first message bytes from the FIFO through the standard C library `getch()` function. The data pump refills the FIFO during each vertical retrace until the video RAM buffer runs dry, at which point the pump goes idle again. `SerGetChar()` then drains the FIFO as the higher-level code finishes parsing the message.

As far as the parsing routines can tell, network messages arrive through `getch()` just like they do in a normal HCS Link. The extended video RAM buffer remains invisible, hidden inside `SerGetChar()` where it belongs.

Switch	Function	Off	On
1	Net name	TERMx	VIDx
2-4	Net address	0	1 (x = 0-7 in binary)
5	Inverse	Normal	Use inverse video on startup
6	Large	Normal	Use large font on startup
7	Bright	Normal	Chars bright on startup
8	Manual	Network	RS-232 "manual" mode

Table 1—Eight DIP switches set the Vid-Link startup options. You must reset the CPU after changing any of the switches, as the firmware reads the switches only once after a reset.

Outgoing messages follow a similar route: from the standard C library putch() to my SerPutChar() function, then into the FIFO.

When the FIFO fills up, SerPutChar() waits until the data pump empties the FIFO into the video RAM during the next vertical retrace. At the end of the message, the pump switches directions, refills the FIFO, and the serial interrupt handler begins dropping bytes into the transmitter.

The data pump delays output messages just as it does inputs. Given the lengthy delay before each poll response, Ken decided that the Vid-Link should be truly a write-only device and tweaked the SC code to suppress its normal status polling messages. You must carefully pace outgoing text because the SC cannot tell when the Vid-Link becomes ready again.

However, both he and Steve report that Vid-Link provides a convenient HCS status display for true home-control junkies. Now, if only it could blink!

COMMANDS AND CONTROLS

Table 1 shows the DIP switch settings that take effect whenever you reset the Vid-Link's CPU. You may have up to eight boards on a single HCS network and up to 16 in custom systems if you use both the TERMx and VIDx network addresses. If you like large, bright, reversed characters, just flip the switches!

You need an external RS-485 converter that adapts the HCS network's

Command	Example	Function
Esc[#A	Esc[2A	Cursor up # rows (up 2)
Esc[#B	Esc[B	Cursor down # rows (down 1)
Esc[#C	Esc[10C	Cursor right # columns (right 10)
Esc[#D	Esc[5D	Cursor left # columns (left 5)
Esc[#;#H	Esc[H	Set cursor to row;column (to 1, 1)
Esc[#;#f	Esc[1;2f	Set cursor to row;column (to 1, 2)
Esc[#;#j	Esc[3j	Set cursor to row;column (to 3, 1)
Esc[s		Save current cursor location (1 level)
Esc[u		Restore saved cursor location
Esc[2J		Clear display and home cursor
Esc[K		Clear from cursor to end of row
Esc[#h		Set display mode
	Esc[Oh	large character set
	Esc[2h	normal character set
	Esc[7h	wrap at end of row
Esc[#l		Set display mode (that's an "ell")
	Esc[7l	force cr/lf at end of row
Esc[#m		Set display attributes
	Esc[Om	normal white on black
	Esc[1m	bright white on black
	Esc[7m	reverse (flips current attributes)

Table 2—The Vid-Link firmware recognizes a useful subset of the ANSI cursor and screen control commands. You may represent the Esc character with the C-style sequence "\e" to avoid unprintable characters in your program text. Note the three equivalent cursor movement commands: H, f, and j.

differential TTL levels to the ImageWise's RS-232 input. Because the Vid-Link does not send any messages or polls back to the Supervisory Controller, a simple TTL level-shifter will probably work in small systems. Send me a note on the BBS if you need help.

The Vid-Link responds to only two commands: S=t e x t writes a message on the screen and N n sets the network mode. The latter command has no use in HCS networks, but comes in handy with directly connected RS-232 units.

If you've used the LCD-Link, the Vid-Link's ANSI control codes in Table 2 look familiar. You must remember which character size you've selected. The firmware truncates the

row and column values to keep the characters on the screen.

Vid-Link uses the C-style escape sequences tabulated in Table 3 to send "unprintable" characters across the network. If you plan to write a C program driving a Vid-Link through PC's serial port, remember to double your backslashes: "\\e" sends "\e" and "\\\\" sends "\\".

And, yes, you can use a Vid-Link as a TV Typewriter. Hitch it to an ordinary PC serial port with a null-modem cable, fire up any terminal program, flip DIP switch 8 On, reset the Vid-Link, and start typing!

RELEASE NOTES

For those of you with an ImageWise board, just download the Vid-Link hex file from the BBS, burn an EPROM, and you're on the air! Sadly, the TML1852

video D/A converter has become hard to find and I don't have any leads for you. If you prefer a finished box, Vid-Links are available from CCI.

Next month, some mildly technical information that may save your project or someone's life. □

Ed Nisley (KE4ZNU), as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at ed.nisley@circellar.com or 74065.1363@compuserve.com.

SOURCE

Vid-Link, LCD-Link, ImageWise
Circuit Cellar, Inc.
4 Park St.
Vernon, CT 06066
(860) 875275 1

I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

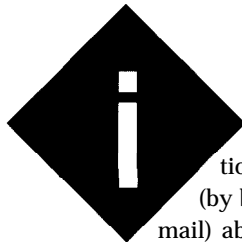
\cn	Control character n (\cZ = Ctrl-Z)
\e	Escape character, ASCII 27
\f	Form feed character, ASCII 12
\n	New line (linefeed and carriage return)
\r	Carriage return (to current line, leftmost column)
\t	Tab to next stop: column 4, 8, 12, 16, 20
\xnn	Send hex char nn directly to display Must have two digits: \x03
\\	Single backslash

Table 3—These character escape sequences simplify sending "unprintable" characters across the HCS network. Use ASCII 92, hex SC, for the backslash, just as you would in ordinary text.

Intel Hex to BASIC Data Statement Translator

FROM THE BENCH

Jeff Bachiochi



get a ton of questions each month (by both phone and E-mail) about using masked

BASIC-52 on the 8052 microcontroller. The ever-increasing interest supports my claim that BASIC offers a familiar and friendly platform to learn embedded control. To the seasoned veteran, it also provides an inexpensive development platform.

The whole thing started back in 1984 when Intel masked an 8-KB control-oriented BASIC interpreter, called BASIC-52, into an NMOS 8052AH DIP-style microcontroller. While Intel no longer sells the chip, Micromint continues to offer BASIC-52 masked into low-power 80C52 DIP and PLCC packages.

with on-chip BASIC-52, writing applications is a snap. No special compilers or assemblers are needed. You just attach a terminal (or PC terminal emulator) and type the lines of BASIC in directly. The results can be stored and executed immediately right there on the target system.

Debugging the application is also painless since all variables can be displayed and BASIC lines edited at any time. For the majority of applications, BASIC is all you need to collect, transform, or redirect data.

Of course, no single programming language fits all control applications. What a BASIC interpreter brings in ease of use and program development, it compromises in execution speed and hardware to BASIC interfacing.

THE HARD FACTS

The 8031 core processor has four 8-bit I/O ports. In an 8052 processor

with the masked BASIC, Port0 and Port2 are used for the external address/data bus. All eight bits on Port1 are available through direct BASIC commands. The bits on Port3 have multiple functions and are available, but only through assembly instructions or assembly routines called from BASIC.

Many applications don't need more than eight I/O bits. However, if you need more, you can add external I/O peripheral chips. These can be easily accessed using traditional PEEK and POKE-type BASIC commands.

Some peripherals require interrupts for tasks which need to take precedence over the BASIC program flow. To facilitate this, BASIC-52 can directly respond to one of the two 8031 core external interrupts. It also can support a 1-s tick clock for interrupts based on elapsed time. The interrupt servicing speed remains that of BASIC.

THE NEED FOR SPEED

When the execution speed of a BASIC application program becomes time-critical, consider supplementing it with lower-level assembly language for speed-sensitive tasks. The typical execution time for a line of BASIC-52 is 230 ms, depending on the command. FOR/NEXT loops are the fastest while P R I N T statements take considerably longer than the average.

Although assembly language executes in microseconds, it generally takes hundreds of lines of code to accomplish what a single line of BASIC can do.

On the other hand, task-specific assembly-language code (e.g., reading and storing A/D conversions) is much faster than interpreted BASIC (for a compiled BASIC the difference is not as significant).

CALL OF THE WILD

So, I contend that you should use a BASIC interpreter whenever and wherever it makes sense. When you need more execution speed, consider compiled BASIC or callable task-specific assembly language routines.

The BASIC-52 CALL 4200H statement saves a pointer to the next line of BASIC code on the stack and then jumps blindly to the address you give

Are you an 8052 user trying to mix assembly language and BASIC? If so, Jeff has a solution. He presents a translation file which automatically creates a BASIC data statement file from an Intel hex file.

it (in this case, 4200H). The processor now expects to fetch an assembly-language opcode to execute.

That's how your assembly routine gains control from BASIC. When your routine has finished, the **RETURN** opcode returns control to BASIC. The pointer to the next line of BASIC is popped off the stack and execution of the BASIC application continues.

Let's assume that all you need to do is set and clear an I/O bit normally unavailable to BASIC, like T1 (P3.5). First, you need a place in memory to store the routine. You might want to place the routine in ROM above the space where the BASIC program resides in autostart mode.

There's one problem with this solution. You now have two programs which must be loaded properly, one BASIC and the other assembly language. While this may not sound like much of a problem, it can be a bookkeeping nightmare for longer programs, especially if you forget to keep the files together for easy maintenance.

I suggest an alternative approach. Try keeping the assembly routine as part of the BASIC application program using **DATA** statements. While this approach involves an extra step to protect the necessary RAM space and poke the routine into memory each time the application is run, the process is quite straightforward. Just look.

When you power up the 80C52 platform, you start out with an allocated address space like that in Figure 1a. The processor has measured the amount of RAM you have in the system and assigns it to the variable **MTOP** (let's assume **MTOP** = 7FFFH for a 32-KB SRAM).

Begin by typing in (or downloading) your BASIC-52 application. It fills

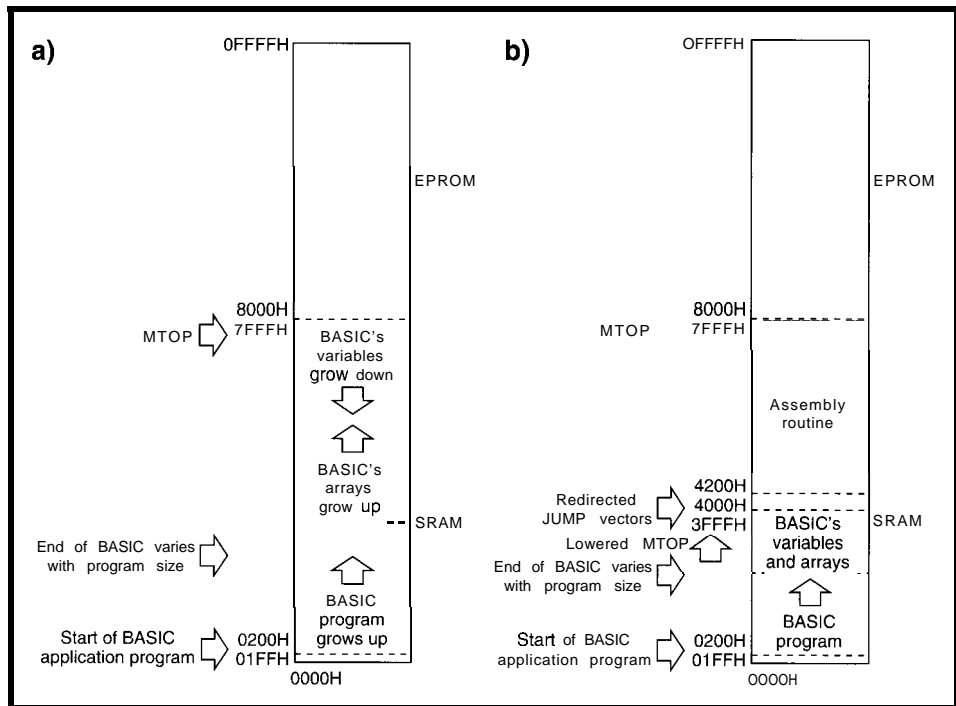


Figure 1-a) At powerup, BASIC puts variable storage as high in RAM as possible. **b)** Modifying **MTOP** protects a portion of memory for use by assembly language routines.

RAM from 200H upward. As the first statement, you need to add a line to protect some memory for the assembly-language routine.

This goal is accomplished by setting the **MTOP** variable to an address lower than that set in the power-up initialization. Let's use 3FFFH, to give you plenty of protected space.

10 MTOP=3FFFH

Notably, if your assembly-language routine were only three bytes long (and didn't require the use of an interrupt), you would only have to protect three bytes (**10 MTOP=7FFCH**).

With **MTOP** reassigned to 3FFFH, you now have the address space allocated as in Figure 1b. Although you may not require the interrupt jump vectors which start at 4000H, I always protect them but leave them free of code. You may need them eventually. (More on this later.) To stay clear of

these locations, I started my code at 4200H.

Let's try something simple like turning on or off bit B5H (P3.5 T1), which you can't do directly from BASIC. You don't need an assembler for something this simple. It only requires two opcodes: a **SETB** (or **CLR**) instruction and a **RETURN**.

By referring to the micro's data book, you can find the correct bytes for setting and clearing a bit. You can place them into **DATA** statements like this:

```
10000 REM Set I/O bit T1
10010 DATA 0D2H, 0B5H:
      REM SETB T1
10020 DATA 022H: REM Return
10030 REM Clear I/O bit T1
10040 DATA 0C2H, 0B5H:
      REM CLR T1
10050 DATA 022H: REM Return
```

The first data byte, D2H, is the assembly-language opcode for setting an I/O bit. The second byte, B5H, is the bit address where the operation is to be performed. The source code for this

```
a)
:20420000E4F590C2D5D2D47820C2D4759850758920758DFD858D8B75B81075A8 927588507A
b)
: 20 4200 00 E4F590C2D5D2D47820C2D4759850758920758DFD858D8B75B81075A892758850 7A
```

Figure 2-a) A raw line of Intel hex looks like a jumble of characters. **b)** The line separated into its six major parts—start character, data length, load address, mode, data, and checksum—becomes easier to deal with.

opcode follows in a remark statement for documentation purposes only.

In the second statement, 22H is an opcode which returns control (in this case, to BASIC). This hand-coding method can be used when there is little chance for error.

You're welcome to hand-code larger routines, but be advised that it's extremely easy to miscode a statement, especially one with relative jumps and such. Do it as a exercise, and back it up with output from an assembler. It's bad enough when your routine doesn't run due to an error in logic. Don't add coding errors to your debugging session!

Now, all you need to do is get these six bytes into protected RAM where they'll be ready for you to call them. I've suggested using 4200H as the starting address. So, you need a BASIC-52 routine which pokes the data bytes into RAM at 4200H using the X BY statement. You can use a routine like this:

```
20 FOR X = 4200H TO 4205H
30 READ V
40 XBY(X) = V
50 NEXT X
```

The **F0 R/N E X T** loop assigns 4200H to variable X. It reads a byte and places it into address location X. The address is incremented, and the read-and-store is repeated until X exceeds 4205H.

Once the data has been stored, it remains in RAM until something overwrites it or the power is cycled off and on. Your BASIC application can **CALL 4200H** to set T1 and **CALL 4203H** to clear T1.

You can even make the calls from the command-line prompt to test them. You quickly discover that if you make a call to a location which either has no routine or has a miscoded routine, anything can happen.

Anything can include totally locking up the system, so you may wish to both check your routine carefully and make sure it's there before you call it (at least the first time). At a minimum, at least ensure the first byte at the location you call is correct.

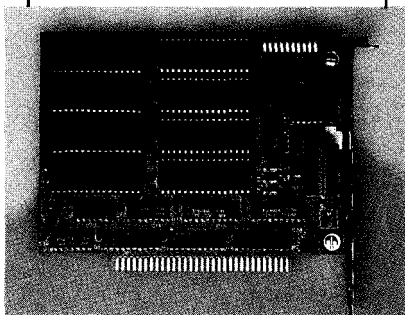
You can also sum all the code you placed in RAM and compare the total

Listing I-This program, written in a generic PC BASIC, translates an Intel hex file into an 80C52 BASIC program and loads the Intel hex data into SRAM.

```
10 CLS
20 FLAG=0
30 REM This program prompts for an Intel hex file name,
40 REM reads the file in, and creates an output file. The
50 REM file can be appended to a 80C52 BASIC program to load your
60 REM assembly routine into SRAM (located in combined
70 REM Data/Code space) for execution there.
80 INPUT "What is the Intel hex filename? ",A$
90 OPEN AS FOR INPUT AS #1
100 PRINT
110 PRINT "The output file will be called DATA.BAS."
120 INPUT "What line number should it begin with? ",LINENUMBER
130 B$ = "DATA.BAS"
140 OPEN B$ FOR OUTPUT AS #2
150 IF EOF(1) THEN O$ = "": TEMP=LINENUMBER: GOTO 970
160 ON ERROR GOTO 950
170 INPUT #1,I$
180 IF (MID$(I$,1,1) <> ".:."), THEN GOTO 930
190 PAIRCOUNT = VAL("&H"+MID$(I$,2,2))
200 LOADADDRESS = VAL("&H"+MID$(I$,4,4))
210 GOSUB 610
220 MODE = VAL("&H"+MID$(I$,8,2))
230 IF (MODE<>0 AND MODE<>1) THEN PRINT "Warning, mode must be 00"
240 IF (MODE=1) THEN PRINT "End of File"
250 FOR X=10 TO 10+(2*(PAIRCOUNT-1)) STEP 2
260 IF (BYTECOUNT>7) THEN BYTECOUNT = 0: GOSUB 890:
    LINENUMBER = LINENUMBER+10
270 IF (BYTECOUNT=0) THEN O$ = "": TEMP=LINENUMBER: GOSUB 390:
    GOSUB 580
280 O$ = O$ + " 0" + MID$(I$,X,2) + "H"
290 TOTALSUM = TOTALSUM + VAL("&H"+MID$(I$,X,2))
300 IF (BYTECOUNT<>7) THEN O$ = O$ + ","
310 BYTECOUNT = BYTECOUNT + 1
320 CHECKSUM = VAL("&H"+MID$(I$,10+2*(PAIRCOUNT-1),2))
330 FOR COUNT = 2 TO 10+(2*(PAIRCOUNT-1)) STEP 2
340 CHECKSUM = CHECKSUM + VAL("&H"+MID$(I$,COUNT,2))
350 NEXT COUNT
360 IF (CHECKSUM AND 255) <> 0 THEN PRINT "Checksum error"
370 NEXT X
380 GOTO 150
390 REM Place the line number digits into a string
400 BLANKFLAG = 0
410 IF (TEMP<10000) THEN GOTO 440
420 TEMPINTEGER = INT(TEMP/10000): O$ = O$ + CHR$(TEMPINTEGER+48)
430 TEMP = TEMP - TEMPINTEGER*10000: BLANKFLAG = 1
440 IF (TEMP<1000) THEN GOTO 470
450 TEMPINTEGER = INT(TEMP/1000): O$ = O$ + CHR$(TEMPINTEGER+48)
460 TEMP = TEMP - TEMPINTEGER*1000: BLANKFLAG = 1: GOTO 480
470 IF (BLANKFLAG=1) THEN O$ = O$ + "0"
480 IF (TEMP<100) THEN GOTO 510
490 TEMPINTEGER = INT(TEMP/100): O$ = O$ + CHR$(TEMPINTEGER+48)
500 TEMP = TEMP - TEMPINTEGER*100: BLANKFLAG = 1: GOTO 520
510 IF (BLANKFLAG=1) THEN O$ = O$ + "0"
520 IF (TEMP<10) THEN GOTO 550
530 TEMPINTEGER = INT(TEMP/10): O$ = O$ + CHR$(TEMPINTEGER+48)
540 TEMP = TEMP - TEMPINTEGER*10: BLANKFLAG = 1: GOTO 560
550 IF (BLANKFLAG=1) THEN O$ = O$ + "0"
560 O$ = O$ + CHR$(TEMP+48)
570 RETURN
580 REM Add the word "DATA" to the string
590 O$ = O$ + " DATA "
600 RETURN
610 REM Track the start and finish address for each segment
620 IF (FLAG<>0) THEN GOTO 650
630 START = LOADADDRESS: FINISH = LOADADDRESS + PAIRCOUNT 1:
    TOTALSUM = 0: FLAG = 1
```

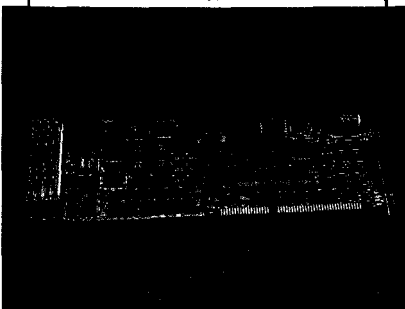
(continued)

VMAX[®]
QUALITY PRODUCTS
RESPONSIBLE SERVICE
RELIABLE DELIVERY



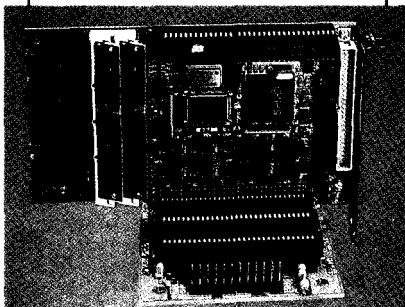
ENHANCED SOLID STATE DRIVE — \$164*

4M Total, Either Drive Bootable
 1/2 Card 2 Disk Emulator
 Flash System Software Included
FLASH & SRAM Customtoo



486 SLAVE PC — CALL

Add up to 4 Boards to One Host PC
Fast Data Transfer and I/O
PC-104 Port, IDE & Floppy Control
 Independent Processors on One Bus
No Special Compilers Needed



TURBO XT w/FLASH DISK — \$266*

To 2 FLASH Drives, 1M Total
 DRAM to 2M
 Pgm/Erase **FLASH On-Board**
CMS Surface Mount, 4.2"x6.7"
 2 Ser/1 Par, Watchdog Timer

All Tempustech VMAX[®] products are
PC Bus Compatible. Made in the
 U.S.A., 30 Day Money Back Guarantee
 *Qty 1, Qty breaks start at 5 pieces.

TEMPUSTECH, INC.

TEL: (800) 634-0701

FAX: (941) 643-4981

Fax for 295 Airport Road
 fast response! Naples, FL 33942

Listing P-continued

```

640 RETURN
650 IF (FINISH+1=LOADADDRESS) THEN FINISH = FINISH + PAIRCOUNT:
    RETURN
660 GOSUB 690
670 FLAG = 0
680 GOTO 610
690 REM Append the loading routine for the DATA statement segment
700 IF (RIGHT$(O$,1)=";") THEN O$ = MID$(O$,1,LEN(O$)-1)
710 IF (RIGHT$(O$,1)<>" ") THEN GOSUB 890:
    LINENUMBER = LINENUMBER + 10
720 O$ = "": TEMP = LINENUMBER: GOSUB 390
730 O$ = O$ + " ST=0: CT=": TEMP = TOTALSUM GOSUB 390
740 GOSUB 890
750 LINENUMBER = LINENUMBER + 10
760 O$ = "": TEMP = LINENUMBER: GOSUB 390
770 O$ = O$ + " FOR X = "
780 TEMP = START: GOSUB 390
790 O$ = O$ + " TO "
800 TEMP = FINISH: GOSUB 390
810 O$ = O$ + " : READ H : XBY(X)=H: ST=ST+H: NEXT X"
820 GOSUB 890
830 LINENUMBER = LINENUMBER + 10
840 O$ = "": TEMP = LINENUMBER: GOSUB 390
850 O$ = O$ + " IF (CT<>ST) THEN PRINT "
860 O$ = O$ + CHR$(34) + "DATA ERROR" + CHR$(34) + " : END":
    GOSUB 890
870 LINENUMBER = LINENUMBER + 10: BYTECOUNT = 0
880 RETURN
890 REM Display and save present BASIC line
900 PRINT #2,O$
910 PRINT O$
920 RETURN
930 REM First character error in Intel hex paragraph
940 PRINT"Error. First character must be a ':'":CLOSE: END
950 REM Character error within Intel hex paragraph
960 PRINT"Error in input file":CLOSE: END
970 GOSUB 390: O$ = O$ + " RETURN": GOSUB 890: CLOSE: END

```

to a known good total placed in the BASIC program:

```

60 S = 0: C = 834
70 FOR X = 4200H TO 4205H
80 S = S + XBY(X)
90 NEXT X
100 IF (C<>S) THEN PRINT"Data
    error": STOP

```

This is where I lose a bunch of people. "I'm not gonna type in all those DATA statements with the code from my assembled source. My Intel hex file is over 1 KB in size."

There isn't much I can say that would convince them it would be worth their while. So, this month I present a piece of code, written in a generic PC BASIC, which reads in an Intel hex file and translates it into BASIC-52 DATA statements. The out-

put can be appended to your BASIC-52 application program.

INTEL HEX FILES

When a source file is assembled into a binary file, it contains no address information and no way-other than the file size-of assuring that the file has not been corrupted. When the binary file is translated into an Intel hex file, it becomes protected, if you will. The binary data is cut into small chunks, called *lines* or *paragraphs*, and surrounded by additional information.

As you can see in Figure 2, each Intel hex line begins with a ":" start character followed by the number of data bytes in the chunk (two hex characters) OOH-FFH. (Note that the chunk must have data bytes which can be loaded into successive addresses.) To

keep the lines viewable on an 80-column screen, the size of a chunk is generally limited to 20H (that's 32 binary bytes or hexadecimal pairs).

The next four characters are the hexadecimal starting address for the first byte in the chunk OOOH-FFFFH. A two-character mode byte follows the load address. If the mode byte is OOH, then data follows.

If the mode byte is 01H, then it's an EOF marker. (Other mode bytes indicate extended addressing and are not used when the address space doesn't exceed 64 KB.) Assuming the mode byte is OOH, the chunk of data follows in hexadecimal format.

Finally, to keep errors from creeping in, a checksum byte is added. Since every line has a checksum byte, each is individually protected. And, since each chunk of data comes along with its load address, every data byte is placed exactly where it belongs, even if the lines somehow get out of sequence.

Code space address	Function
0000H	RESET
0003H	IE0 (external interrupt 0)
000BH	TFO (timer 0 overflow)
0013H	IE1 (external interrupt 1)
001BH	TF1 (timer 1 overflow)
0023H	RI & TI (serial-port interrupt)
002BH	TF2 & EXF2 (timer 2/capture) (80 x 2 only)

Table 1—The 8031 core interrupt vectors require code memory space starting at 0000H.

FILE TRANSLATION

The file HEX2BAS.BAS (Listing 1) was written in a generic PC BASIC and should be usable with any of the more powerful BASICs available today.

When run, it asks for the Intel hex file's name and what BASIC-52 line number to begin with. You should answer with the file name you'd like translated and a line number higher than any used in your BASIC-52 application (e.g., 10000). An output file called DATA.BAS is created, and both files are opened.

As each line is read in from the Intel hex file, it is dissected. All the

important information is extracted, like load address, number of data bytes, data, sum of the data, and legal checksum. Errors are flagged during processing.

An output string is formed using line-number information and the proper format for BASIC-52 DATA statements. When the output string reaches eight data values, it's written to the output file, and the line number counter increments.

Meanwhile, if the next input line's load address is not the next sequential address, the code is not sequential and the new data must be handled as such. So, the last data byte in the last line must be the end of the last block and therefore the block's ending load address. I now must create a FOR/NEXT loop load routine for that last block. Remember the routine which loads the data into RAM?

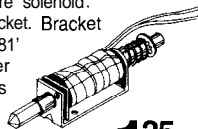
I've been keeping track of the sum of the data within this block. Therefore, I can also allow the load routine

ALL ELECTRONICS CORP.

QUALITY PARTS . DISCOUNT PRICES . HUGE SELECTION

MINI 12 VDC SOLENOID

12vdc, 54 ohm miniature solenoid. Coil mounted on U bracket. Bracket size: 0.36" x 0.43" x 0.81" ong. 1.65" long plunger extends from both sides of coil. Spring return. Can be used for both push and pull action.



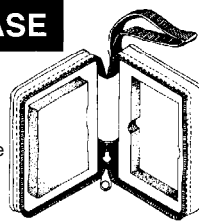
\$1²⁵ each

CAT # SOL-23

10 for \$10.00

CANVAS CASE

Good-looking, well constructed 7.5" X 6" X 1.95" canvas carrying case with zipper on three sides. Webbed nylon hand strap and ample foam padding inside.



Probably designed for transporting memory storage media. The black foam padding inside can easily be reconfigured to accommodate photographic, audio or video components. The removal of one large chunk of foam gives you a well padded interior volume of 5.9" x 4.3" x 0.75". All black with red and white embroidered "K-stor" (tm) logo on front.

CAT# CSE-3 \$3⁷⁵ each

10 for \$35.00

HEAVY DUTY HEAT SINK

Black anodized aluminum heatsink for heavy-duty heat dissipation. 15 cooling fins. Three threaded brass inserts embedded in ends of heatsink to facilitate mounting. 10.75" x 4.82" x 1.43".



\$9⁷⁵ each

CAT# HS-18

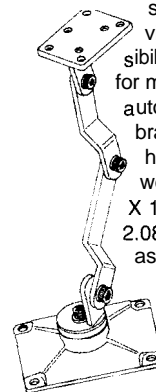
Visit Our World Wide Web Site...

<http://www.allcorp.com/allcorp/>

TILT/SWIVEL MOUNTING BRACKET

Ideal for video cameras, small speakers, video monitors and other equipment, these aluminum brackets feature locking multi-position ratchet joints and a 360 degree

swivel ratchet base to provide endless mounting possibilities. Originally designed for mounting cellular phones on automobile consoles, these brackets could be used for ham and CB applications as well. The base plate is 3.55" X 1.97". The end plate is 2.08" X 1.77" and has an assortment of pre-drilled holes for various applications. Arms are 1" X 0.2" die-cast aluminum. 10" reach fully extended. Flat black.



CAT # SMB-1

\$7⁰⁰ each

10 for \$65.00

ORDER TOLL FREE

1-800-826-5432

CHARGE ORDERS to Visa, Mastercard, American Express or Discover

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

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

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

E-Mail - allcorp@allcorp.com

IMMEDIATE RESPONSE!



When you've got to get that project done **NOW...**

Our off-the-shelf SBCs help you ship **FAST**

552SBC

80C552, a '51 Compatible Micro
40 Bits of Digital I/O
8 Channels of 10 Bit A/D
3 Serial Ports; 232 or 422/485
2 PWM channels
6 Capture/ Compare Inputs
Real Time Clock
(MKB SRAM; 1 UVPRM Socket
512B Serial EEPROM
Watchdog, PFI, Regulation
Expansion bus for more circuits
Development ROM Available

Our popular 552SBC-40 starts at just \$299, quantity 1. And we can create a version just for your needs, then pass the savings on!

C/C++ w/188SBC

80C188XL: x86 Family Micro
16 channel, 12-bit A/D
8 Channel, 12-bit D/A
24 Opto Rack Channels
2 Serial Ports; 232 or 422/485
Real Time Clock, Watchdog
512KB Static, batt-backed RAM
2 UVPRM or Flash Sockets
PFI, On-Board Power supply
PC/1048-bit Expansion bus
FPGA Socket and more!

Use our development tools with your MS & Borland compiler. Quantity 1 pricing from \$299 for the 188SBC-10 to \$749 for the -50!

Other products are available. Call Today!

We specialize in custom work too; as few as 25 units. CALL NOW!



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

Since 1983

(619) 566-1892



E-mail: info@hte.com - Ftp: ftp.hte.com
Web: http://www.hte.com

Table 2—80C52 BASIC remaps most of the interrupt vectors up to the 4000H area in code space.

Code space address	Function
4003H	IEO (external interrupt 0)
400BH	TFO (timer 0 overflow)
4013H	IE1 (external interrupt 1)
401 BH	TF1 (timer 1 overflow)
4023H	RI & TI (serial-port interrupt)
402BH	TF2 & EXF2 (timer 2/capture) (80 x 2 only)
4030H	UO1 (custom console output)
4033H	UI1 (custom console input)
4036H	(custom console status check)
403CH	PRINT@/LIST@
41 00H-41 FFH	CALL O-CALL 7FH

to do a health check on the data when it loads it into RAM from within my 80C52 application program.

Additional blocks are translated the same way. How, you might ask, can an Intel hex file contain more than one sequential block? For many files, it won't be. However, if you are using an interrupt [i.e., serial or timer), the processor has special preset locations it calls when an enabled interrupt occurs. Table 1 gives these interrupt-vector locations for the 8031 family.

In the 80C52, the masked BASIC has complete control of these locations (remember, code space 0000-1FFFH is internal). Knowing that users might want to access some of these interrupts, BASIC shadows the jump vectors to code space 4000H-41FFFH, where they are in external address space. Since I use overlapping data and code space (*PSENOR'd with RD), this location is smack in the middle of RAM.

BASIC-52 application programs which don't use interrupts can safely have BASIC program lines up through this address space since the processor won't ever be calling this area. However, if you provide an assembly-language routine using the interrupts, you must protect the vector area by lowering MIOP to 3FFFH. The interrupt jump vectors are found at the locations shown in Table 2.

The vectors are called *interrupt jump vectors* because the user is expected to place an LJMP XXXX at the vector location to redirect the program flow to the beginning of their routine (i.e., XXXX).

For this reason, I like to stay out of the 4000H-41FFFH area with my routines. Also, this use of vectors explains how assembled code can be non-sequential. If your assembly routine was

placed right at the jump vector location, it would overlap into successive jump vector locations.

Take a look at Figure 3 to see how this problem is handled in the D EMO 1. H E X file translation. Figure 3 shows what happens when you run it. The DATA. BAS file that is created can now be appended to your application. Remember to add the MTO P statement to your application to protect some RAM and add a GOSUB 10000 statement so this appended portion loads the routines into protected RAM.

Here's a quick overview of just what the translation program does. Skip over lines 10000-10030 for a moment. Notice lines 10040-10070. Three data bytes are loaded into 0023H-0025H, an LJMP 424DH. This is the serial-port interrupt. To allow BASIC to connect, you must change the load values from X=35 TO 37 (23H-25H) to X=16419 TO 16421 (4023H-4025H).

Next, a second jump vector is loaded (lines 10080-10110) at 000BH-000DH. This LJMP 4242H is the timer-0 overflow interrupt. Change line 10100 from X=11 TO 13 (000BH-000DH) to X=16395 TO 16397 (400BH-400DH) to allow BASIC the hooks for this routine.

The main body of data consists of the actual routines which reside at 4200H (line 290, X=16896). No changes need to be made here. In fact, if your routines do not use interrupts, this one block of code is all you most likely will see.

Now, back to the first few lines. This code was written to be in total control of the processor and, as such, would normally get control from the reset vector at 0000H. The first chunk

of code is the reset vector jump. Since BASIC-52 runs on powerup and this vector is not shadowed like the others, it can be discarded. Well, almost.

Instead, 4200H is the location BASIC would call to enter the routine. In this example, there is never a return to BASIC. The assembly-language routine completely takes over until power is shut down. Here, BASIC loads the jump vectors and routines—once it passes execution over to the routine, it is never heard from again.

This situation, of course, is the extreme. Why bother at all with BASIC once you are writing totally in another language?

And rightly so. I do not advocate the use of BASIC for every application. But, I do like the friendly development environment and the ease of getting an application up and running.

As you have seen here, it is very possible for BASIC-52 and assembly routines to coexist. I hope I have demonstrated a way you can use the 80C52 to have your cake (the power of BASIC) and eat it too (call on the speed of assembly language). Remember, when all you need to do is tap, don't use a sledge. □

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

IRS

428 Very Useful
429 Moderately Useful
430 Not Useful

What is the Intel hex filename? DEM01.HEX

The output file will be called DATA.BAS. What line number should it begin with? 10000

```
10000 DATA 002H, 042H, 000H
10010 ST=0: CT=68
10020 FOR X = 0 TO 2 : READ H : XBY(X)=H: ST=ST+H: NEXT X
10030 IF (CT<>ST) THEN PRINT "DATA ERROR": END
10040 DATA 002H, 042H, 04DH
10050 ST=0: CT=145
10060 FOR X = 35 TO 37 : READ H : XBY(X)=H : ST= ST+H: NEXT X
10070 IF (CT<>ST) THEN PRINT "DATA ERROR" : END
10080 DATA 002H, 042H, 042H
10090 ST=0: CH=134
10100 FOR X = 11 TO 13 : READ H : XBY(X)=H: ST= ST+H: NEXT X
10110 IF (CH<>ST) THEN PRINT "DATA ERROR": END
10120 DATA 0E4H, 0F5H, 090H, 0C2H, 0D5H, 0D2H, 0D4H, 078H
10130 DATA 020H, 0C2H, 0D4H, 075H, 098H, 050H, 075H, 089H
10140 DATA 020H, 075H, 08DH, 0FDH, 085H, 08DH, 08BH, 075H
10150 DATA 0B8H, 010H, 075H, 0A8H, 092H, 075H, 088H, 050H
10160 DATA 012H, 042H, 03FH, 0FAH, 0FBH, 0F5H, 090H, 0F4H
10170 DATA 0F5H, 08CH, 0EBH, 020H, 0D5H, 004H, 003H, 002H
10180 DATA 042H, 033H, 023H, 0FBH, 0F5H, 090H, 012H, 042H
10190 DATA 03FH, 06AH, 060H, 0EEH, 06AH, 080H, 0E4H, 0E5H
10200 DATA 0A0H, 022H, 0COH, 0EOH, 0B2H, 0D5H, 0EAH, 0F4H
10210 DATA 0F5H, 08CH, 0DOH, 0EOH, 032H, 0COH, 0EOH, 0COH
10220 DATA 0D0H, 0C2H, 098H, 075H, 0DOH, 010H, 0E5H, 099H
10230 DATA 0F5H, 0F0H, 064H, 0ODH, 070H, 0OFH, 074H, 01FH
10240 DATA 0C3H, 098H, 0F4H, 060H, 013H, 0FBH, 0E4H, 018H
10250 DATA 0F2H, 0DBH, 0FCH, 080H, 0OBH, 074H, 03FH, 0C3H
10260 DATA 098H, 0B3H, 050H, 004H, 0E5H, 0F0H, 0F2H, 008H
10270 DATA 0D0H, 0D0H, 0D0H, 0EOH, 032H
10280 ST=0 : CT=18439
10290 FOR X = 16896 to 17020 : READ H : XBY(X)=H : ST=ST+H :
NEXT X
10300 IF (CHECKTOTAL<>STORETOTAL) THEN PRINT "DATA ERROR" : END
```

End of file OK

Figure 3-A run of the translation program demonstrates the kind of output you can expect from an Intel hex file.

BIG THINGS COME

DOMINOTM

Microcontroller





DOMINOTM
Microcontroller
MODEL _____ SN _____

Starting at
\$79

Micromint's Domino-52 microcontroller is a "supercomputer" in less than 0.75 cubic inches. We've packed the most essential elements into one tiny package. Domino is a plug-and-go module, just attach +5V and a terminal or network. A simple keyed sequence saves an autostarting program in nonvolatile memory.

SPECIAL FEATURES

- 80C52 with ROM-resident, full floating-point BASIC
- 32K bytes SRAM and 32K bytes EEPROM
- Two PWM outputs, I²C bus
- Serial I/O: (up to 19,200 bps) RS-422, RS-485 & RS-232A
- Two interrupts and three timers
- Parallel I/O: 12 bits, 3 shared with ADC and I²C
- Power: +5V @ 15 mA;
- Size: 1.75" x 1.062" x 0.4" potted
- A/D converter: 2 channels, 12 bits, 10k samples/sec.
- Connections: via 2-10, 0.1" dual-row header
- -20°C to 75°C operating temperature
- Industrial temperature available



4 Park Street • Vernon, CT 06066

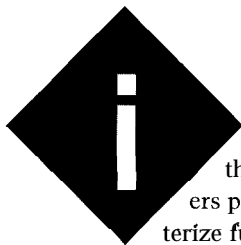
Call 1-800-635-3355

(860) 871-6170 • Fax (860) 872-2204

Fuzzy Buster

SILICON UPDATE

Tom Cantrell



It's not surprising that many designers perpetually characterize fuzzy logic as

Tomorrow's Technology Of Tomorrow. After all, if hype and hope were enough to guarantee success, I suppose we'd all be programming Ada on RISC MPPs. Instead, we're using 25-year-old languages on 20-year-old CISCs in 15-year-old boxes.

Fuzzy certainly hasn't been adopted as a mainstream design technique and the relatively few (though more than you might guess) successful applications are easy to overlook.

However, don't fall into the trap of prematurely dismissing a technology just because it isn't an instant hit. Yes, if you build a better mousetrap, the world will beat a path to your door, but sometimes it takes them a while to find it.

Some say the name itself is an obstacle, implying the flakiness of fuzzy thinking. I've heard the alternative of Zadehan Logic proposed, as if naming it after Lotfi Zadeh, one of the pioneers, would mimic the computer folks' success with Boole and Boolean.

I personally doubt changing horses is a good idea at this point. First of all, dismissing a decade's worth of brand awareness and name recognition isn't something that should be done casually. Also, I think the proposal that a name change compels customers to buy underestimates the customer.

I mean—we're talking about engineers, not consumoids that froth on each New and Improved command. Instead of worrying about fuzzy-thinking, I suggest proponents think along the lines of warm and fuzzy.

Indeed, the concept is apparently perceived as scary and intimidating by many. I trace some of this back to all the AI hoopla back in the '80s that lumped everything from neural networks to expert systems and fuzzy into the category of revolutionary technologies. However, the subtle implication of such positioning is that these technologies are only useful for revolutionary (i.e., difficult to design, expensive to build, and hard to sell) products.

Honestly, though, how many have the huevos to push the product envelope with an unproven technology—it's the K-age equivalent of the first moon shot? Despite big talk, I suspect most (me included) would be crying for mommy about the time they lit the fuse.

In fact, history more often shows that revolutionary technologies usually first appear in mundane products, a classic example being our beloved micros. After all, the engineers at Intel didn't intend to revolutionize the computer business when they set out to build a lowly calculator chip.

WHAT'S ALL THE FUZZ ABOUT?

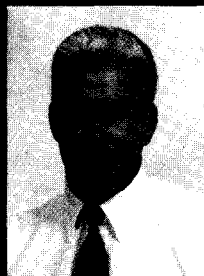
I don't have to explain the basics of fuzzy since those of you in need of a refresher need go no further than INK 56, which contained a number of good articles on the topic. Instead of examining a tree, I'd like to step back a little and talk about the forest.

Despite an excess of obfuscatory terminology, there's really nothing radical or weird about fuzzy. In fact, I contend you've been using fuzzy all along and just didn't know it!

For instance, IF/THEN statements and table lookups are little more than digital equivalents of the mysterious-sounding fuzzifier.

Same goes for anything with an A/D converter. Consider a hypothetical system that feeds two analog signals into two 0-5-V 8-bit A/D converter ports and their sum into a third. Now, feed exactly 1.254 V into each channel and check the sum. No one is shocked or outraged by the fact you end up with the IC equivalent of $64 + 64 = 129$.

Think your little programs are so precise and provably correct, eh! Try



Fuzzy logic is no longer quite so alien. As

Tom shows in this article, fuzzy logic has hit hardware. The AL220 fuzzy controller offers simple, small, and cheap solutions for management of analog input and output.

something like this...

```
x = 0.12345
x = x * x
x = SQRT(x)
IF x <> 0.12345
  THEN GOTO THE_MOON
```

Since everyone already has 1s and 0s on the brain, viewing fuzzy logic from a Boolean standpoint may be helpful. Simply replace AND with MIN and OR with MAX and you've got the picture.

Another way of looking at it is along kind of a variable-state continuum. With binary, there are only two states, so you need a lot of variables (i.e., HOT, WARM, COOL, COLD = 1 or 0). With fuzzy, there are lots of states, but you need fewer variables (i.e., TEMP = HOT, WARM, COOL, COLD).

In neither case are you allowed infinite variables or states, so what's the big deal?

Admittedly, fuzzy isn't good in the contrived world of numbers and symbols. Consider that good old BCD can track our national debt down to the centavo while if you piled up all the IOUs on a big scale and hooked it to an A/D converter, you'd surely be off by billions. Nobody needs a fuzzy check-book balancer to tell them they're kinda broke.

ONCE MORE, WITH FEELING

What success there's been with fuzzy has largely been based on software (i.e., libraries of code that imple-

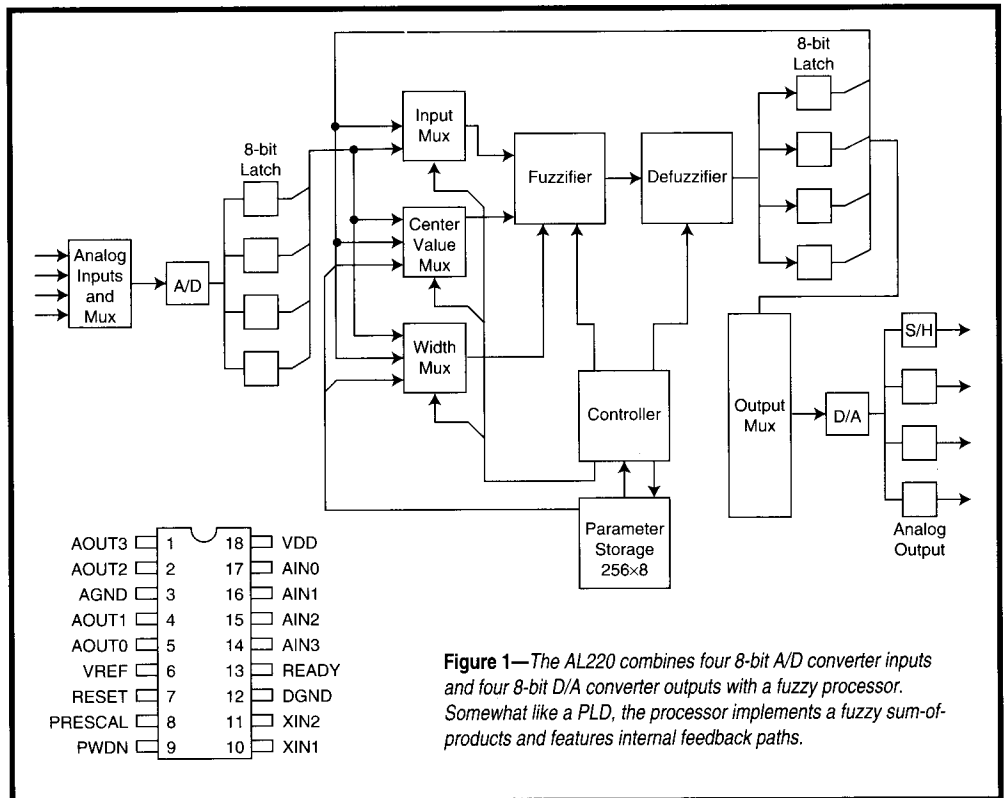


Figure 1—The AL220 combines four 8-bit A/D converter inputs and four 8-bit D/A converter outputs with a fuzzy processor. Somewhat like a PLD, the processor implements a fuzzy sum-of-products and features internal feedback paths.

ment the technique on standard micros). For instance, all the examples in *INK 56* (relying on a PC, 68HC11, and PIC) fall into this category. The fact that most fuzzy applications run on regular computers further highlights the six-of-one, half-dozen-of-another duality of the schemes.

The situation for fuzzy hardware is, pardon the expression, much more fuzzy. I first wrote about a chip called the Fuzzy Set Comparator (FSC) way back in *INK 15* in an article entitled "I've Seen The Future—And It Is Fuzzy." Clever title that, since skeptical readers may wonder—after five years—just how far in the future I meant?

Titles like this are pundits' favorites since the future can always mean "until you forget I wrote it."

The FSC wasn't hard to understand, and I even showed how

a page or so of code could emulate it. The fact the chip could run a thousand times faster than the emulation was neat, but also contributed to the revolutionary positioning problem.

For instance, the FSC evaluation kit consisted of a PC plug-in board full of logic with a high-speed video frame grabber, raising machine-vision expectations to unrealistic heights. Of course, the FSC couldn't live up to such highfalutin aspirations, reinforcing blue-sky stereotypes. Needless to say, the supplier of the FSC is no longer in the fuzzy biz.

With little happening on the fuzzy hardware front since then, I was intrigued by the recent announcement of the AL220 Fuzzy Controller from a local startup. In the Phoenix-like way of Silicon Valley, Adaptive Logic has risen from the ashes of the earlier efforts, and many of the same folks are involved. Persistence isn't everything (ask Forth aficionados), but it sure helps.

DÉJÀ VOODOO

Despite common roots, the AL220 is completely different in form and philosophy from the earlier fuzzy chips. Although the FSC was an expen-

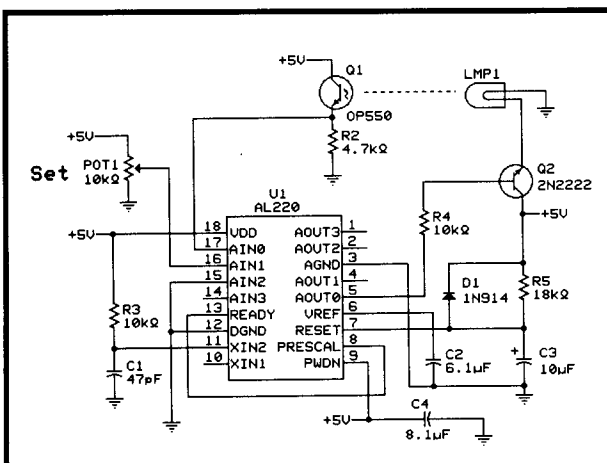


Figure 2—Though capable of digital I/O, the AL220 best serves standalone analog applications like this closed-loop dimmer.

sive (\$38), high-pin-count device targeted at grand challenge-class applications, the AL220 is simple, small (18-pin DIP, 20-pin SOIC) and, best of all, cheap (\$7.50 singles, under \$2.00 in high volume).

As shown in Figure 1, the AL220 sandwiches a small fuzzy processor between analog inputs and outputs. From the outside, it looks more like a simple analog chip than a micro, so the pinout is blessedly easy to dissect.

XIN1 and XIN2 are the clock inputs, accommodating a crystal or TTL input in the usual way. A version of the chip is also available that accepts an RC clock source for lowest cost. Certainly, the Reset pin doesn't need a heck of a lot of explanation—just punch it for eight or more clocks and off you go.

AIN and AOUT are the appropriately named analog I/O channels which feature 8-bit resolution across a 0–5-V span. As is, it follows the usual practice for mixed-signal chips. The analog section and digital sections have their own grounds (AGND and DGND). The internal voltage reference is also brought out (via VREF) for connection to AGND with a 0.1- μ F capacitor.

Power (5 V \pm 5%) is supplied on VDD, but not much of it since the AL220 only draws 30 mA max running at full throttle (10 MHz). The power-down input (PWDN) puts the chip to sleep, cutting power consumption to a trivial 20 μ A.

While it's natural to focus on how fast a chip goes, don't forget to ask how slow it can go too. The AL220 uses dynamic circuits, so the minimum clock rate is 1 MHz, and there are applications that not only can, but must, run slower.

The Prescale input establishes a more leisurely pace by enabling a programmable 8-bit clock divider, thus cutting the minimum effective clock rate to less than 4 kHz. Note that Prescale can be dynamically controlled during operation for slow-and-go situations.

The Ready pin clearly isn't the wait-state requester you might expect since it's an output. It simply is asserted after a short delay following the removal of Reset. As far as I can tell, its main role is to provide an easy way to invoke Prescale mode.

It turns out that the Prescale input spec requires a 4-clock delay after the removal of Reset before Prescale can be asserted. Connecting Ready to Prescale establishes the correct delay without the need for an external counter or flip-flops.

FISC?

Given the minimalism and analog I/O of the chip, an AL220 hardware design becomes little more than an exercise in signal conditioning. Let's look at a simple application: a dimmer that drives a lamp to a level set on a pot (Figure 2).

Actually, it's not as trivial as a typical dimmer which runs open loop. In this case, the loop is closed with a photosensor, which means the same amount of light is always delivered at a given setpoint, compensating for any lamp, wiring, or ambient-light variations.

About now is when any self-respecting fuzzy article gets into the bits and bytes of fuzzy, typically with a blizzard of membership graphs. I'll oblige with Figure 3, but that's all since I find them rather more distracting than illuminating.

Instead, let's just start pounding on the keyboard and voilà, out comes the program (Listing 1). Step through it a line (er, rule) at a time, and you'll see how easy it is.

The input and output sections name the I/O lines: Lamp (the drive voltage), Bright (the actual light level), and Set (the desired light level). The parentheses around (Lamp) in the input section indicate it is fed back internally in digital form (notice the feedback path from the output diagram on the block diagram), but it's conceptually the same as connecting the pins externally (i.e., AOUT to AIN).

The next section, Fuzzy Variables, admittedly looks a little odd, but it's easy to understand in words. What's happening is that the inputs are being classified into groups based on their value. The Left Inclusive and Symmetric Exclusive stuff sounds

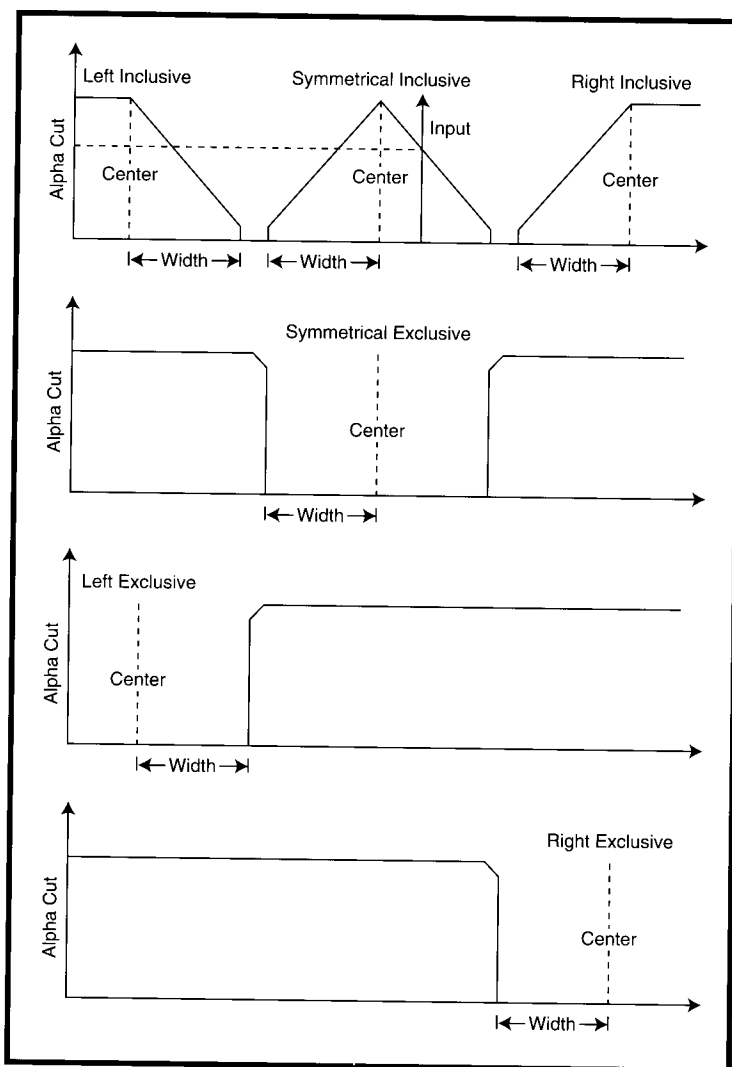


Figure 3—An input (e.g., temp) is classified (hot, cold, etc.) with combinations of membership diagrams, each defined by a center, width, and shape.

Listing 1—Though *trivially* simple, the dimmer *software* does *illustrate* some of the *AL220* features including *floating membership* (the *Set* center *point* for *Bright*) and *direct (=)* and *accumulating (+,-)* output.

Inputs
(Lamp)
Bright
Set

outputs
Lamp

Fuzzy Variables

Lamp is LowLimit (90, 0, Left Inclusive)
Lamp is HiLimit (200, 0, Right Inclusive)
Bright is target (Set, 2, Symmetric Inclusive)
Bright is low (Set, 2, Right Exclusive)
Bright is high (Set, 2, Left Exclusive)
Bright is vhigh (Set, 5, Left Exclusive)
Bright is vlow (Set, 5, Right Exclusive)

Rules

If Lamp is LowLimit
then Lamp = 91
If Lamp is HiLimit
then Lamp = 199
If Bright is target
then Lamp + 0
If Bright is vlow
then Lamp + 2
If Bright is low
then Lamp + 1
If Bright is vhigh
then Lamp + -2
If Bright is high
then Lamp + -1

IF <term> **THEN** <action>

The <action> is where the output gets set. The example shows both forms of action output: direct (=) and accumulating (+ or -). Note that the accumulating forms saturate (i.e., they won't overflow by incrementing past 255 or decrementing **below 0**).

If that's all there were to **it**, the AL220, despite low price and built-in analog I/O, wouldn't bring a whole lot more to the party than a penny-pinching micro or even a PAL. However, there are a number of embellishments beyond those apparent in this tutorial application.

Though the dimmer example only has seven rules, the AL220 can hold up to 54. Furthermore, while each rule in the example includes only a single term, the chip lets a number of terms be strung together with fuzzy ANDs. For instance, in a coin recognizer application, you might see:

```
IF <Diameter is 10Dia> AND
   <Thick is 10Thk> AND
   <Magnetic is 10Mag> THEN
   Di me=255
```

daunting, but it's really just the good old less than, greater than, in between, and so on that everyone's familiar with.

For instance, the line `B r i g h t i s target (Set, 2, Symmetric Inclusive)` is just fuzzy-speak for "The light level is at the target level when it's within 2 counts of the setpoint." Similarly, the `B r i g h t i s l o w` and `B r i g h t i s v l o w` statements classify light levels at 3-4 and 5+ counts below the setpoint, respectively. In other words, the part of the statement in parentheses defines the membership function's center, width, and shape.

Variables whose classification depends on a variable input (e.g., `Set`) are said to feature floating membership, a key feature for adaptive control. By contrast, the first fuzzy variables in the program, `LowLimit` and `HiLimit`, are hardwired with fixed values.

Finally, we get to the rules which define the action to take depending on evaluation of the fuzzy variables. The simplest form of a rule is:

*Under the hood of a stock
Cimetrics RS-485 Protocol Stack ...*

*... plenty of software power to drive
an embedded microprocessor network!*

Performance
Implementation

Exactly what
you would expect
from the leader in
microcontroller networks
for RS-485, BACnet™,

Ethernet™ and ARCNET™ environments. Have your RS-485 network up and running smoothly in just *hours* – at a fraction of the development and support cost – with full support from the experienced team at Cimetrics. Please call, or FAX us at 617-350-7552.

Cimetrics • Boston, MA • 617-350-7550

A key point to realize in a multivariable application like this is that membership functions for each variable (center, width, and shape) can overlap. For instance, the allowed thickness function of a nickel and quarter overlap, but the diameter and magnetic functions don't.

In the dimmer example, the values in the action clauses are all literals, but they could also specify another input or output. It's a little tricky, but exploiting this fact and the floating-membership feature provide a way to hack the fuzzy equivalent of a PID function.

The chip takes all the rules affecting a single output and evaluates them. The value of each rule is that of its lowest-value term. Then, all the rule values are compared and the highest one is declared the winner and rewarded with performance of its action clause.

A BIT OF A DILEMMA

Though the fuzzy code looks quite similar to that of a micro, it's important to note that the timing is completely different.

Where a micro operates instruction-by-instruction, the AL220 operates in kind of a batch mode in which all the inputs are sampled, rules evaluated, and outputs updated in a single 1024-clock pass.

This technique is kind of neat since it introduces a level of determinism hard to obtain with a micro. Changes to the fuzzy program have no timing impact since the AL220 always takes 1024 clocks no matter how numerous or complicated the rules.

A thousand clocks sounds like a lot, but keep in mind that at 10 MHz, the AL220 is sampling all the inputs, running the entire fuzzy program, and updating all the outputs at a 10-kHz rate. Depending on the complexity of the program, a low-cost micro would be hard pressed to keep up.

Though the AL220 is clearly best for purely analog, self-contained applications, the designers were nice-and-pragmatic-enough to realize that it's too late to pretend digital doesn't exist.

For instance, many motor or heater control tasks for which the AL220

Listing 2-The AL220 generates a PWM output by comparing a ramp voltage with an analog output and setting the digital output on or off accordingly.

```

INPUTS
(Ramp)           ;Pulse-width ramp voltage
(Analog_Out)    ;Analog-control output

OUTPUTS
Ramp
Analog_Out
PWM_Out         ;Pulse-width output corresponding to Analog-Out

VARIABLES
Ramp is Count (255,0,Right Exclusive)
Ramp is Reset (252,0,Right Inclusive)
Ramp is On (Analog_Out,0,Left Inclusive)
Ramp is Off (Analog_Out,0,Right Inclusive)

RULES
IF Ramp is Count THEN Ramp + 12 ;Generate ramp voltage
IF Ramp is Reset THEN Ramp = 0  ;Takes 21 cycles (21*12=252)

If Ramp is Off THEN PWM_Out=0   ;PWM off when Ramp > Analog_Out
If Ramp is On THEN PWM_Out=255  ;PWM on when Ramp < Analog_Out

```

seems a natural call for a PWM control signal. Furthermore, given the proliferation of micros and coprocessing design techniques, it might make perfect sense to combine an AL220 and micro in a single system.

To all these ends, the AL220 includes some hooks to connect with the digital world. For instance, the general rule that an output is updated once per 1024-clock pass has a caveat. If two sets of rules whose action clauses reference a common output are separated by a rule(s) referencing another, two updates occur. This provides an unobvious, though workable, way to perform bit manipulation and waveform generation.

Consider the combination of variables and rules in Listing 2 that converts an analog output (ANALOG_OUT) to a digital PWM (PWM_OUT). It works by generating a ramp voltage that's compared to the Analog_Out with the result (greater or less than) driven to the PWM_Out line.

One nice feature of matching an AL220 to digital logic is intrinsic handling of the mixed-voltage dilemma. For instance, connecting the 5-V AL220 to a 3-V IC is as simple as a code change along the lines of `TT L_H I = 180`.

MORE SOLDER, LESS TALK

When it comes to fuzzy hardware, there's certainly been no shortage of hot air (my own included). I figure it's about time for the technology to put up or shut up.

My gut feeling is the AL220 can do some neat stuff, but the only way to find out is to actually wire something up. So, that's what I plan to do. Stay tuned for a follow up soon. •j

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

CONTACT

Adaptive Logic, Inc.
800 Charcot Ave., Ste. 112
San Jose, CA 95131
(408) 383-7200
Fax: (408) 383-7201

431 Very Useful
432 Moderately Useful
433 Not Useful

CONNECTIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) 871-1988—Four incoming lines
Internet E-mail: sysop@circellar.com

We're going to start with a software thread this month. Anyone who's done any programming on the PC has encountered its ornery segmented memory architecture. The first thread covers how you deal with large storage buffers.

In the other thread, we look at a hardware design problem that seems simple, but isn't entirely obvious: controlling the amplitude of an AC waveform with a digital value. The solution turns out to be quite elegant.

C and DOS

Msg#:15123

From: Neil Cherry To: All Users

I'm having trouble with a C program I'm writing for my EPROM burner. It involves a 64-KB segment of memory I need to malloc. When I attempt to run this program it always locks up. I figure it has something to do with the memory model I'm using (large). The .exe is 72 KB in size and mallocs a 64-KB segment. Here are the code segments:

From ZAP.H (the colons indicate other code chunks removed for clarity):

```
#define NEEDED 16*1024

unsigned int far *storage; /* 64K for EPROM */

extern unsigned int far *storage;

From INIT_ZAP.H:

if (!(storage = (int far *) malloc(NEEDED))) {
    printf("Out of memory (%d)\n", errno);
    fini();
}
else {
    printf("%dk reserved\n", NEEDED/1024)
}
```

From FILL.C:

```
for (addr = begint; addr < endt;
     storage[addr++] = 0xFF);
```

If I compile this code into the program the program will lock up when I run the fill code. I'm using Turbo C++ 3.00 for DOS.

Msg#:15504

From: Ed Nisley To: Neil Cherry

Basically, you can't get there from here with the standard C runtime library..

The catch boils down to 16-bit memory addressing and the dreaded 64-KB segment limit. When malloc() and its ilk allocate a memory block for you, they tack on a small (typically 16-byte) header that identifies the block and links it to other allocated blocks. Both your block and the header must lie in the same 64-KB segment, which means you can't allocate precisely 64 KB in a single block.

Where is this documented? Beats me; there used to be a section in the back of the manual titled something like "Limits" with all the little gotchas. It's no longer there and I can't find anywhere else that mentions such. If you know what you're looking for, you can find the memory block definition (in mem.h, I believe) and figure out the header size..which surely doesn't count as good documentation!

I think the lack of documentation boils down to Borland's integrated manuals: they ship the same text in different wrappers for all their C/C++ compilers. If you compile the program for a 32-bit operating system, the 64-KB limitation Goes Away..so they just eliminated any mention of it.

The standard workaround involves allocating a bunch of smaller blocks and connecting them in a linked list or array of pointers. This solution is not nearly as nice and you get to handle data split between blocks all by yourself. Unfortunately, there's no nice way to allocate a complete 64-KB block without working around the C malloc() function..

Something of a motivation to move to 32 bits, isn't it?

Msg#:15511

From: Dave Tweed To: Neil Cherry

Ed's comments with respect to 64-KB segments notwithstanding, your problem is much more basic: you haven't

CONNECT TIME

allocated as much memory as you think! Your request to `malloc()` is for 16K bytes, not `ints`, so that when you fill the memory with 64 KB of data, you're overwriting stack, other heap, and/or code. Try:

```
if (!(storage = (int far *) malloc(NEEDED *
    sizeof(int)))) /* */
```

Msg#:16358

From: Rufus Smith To: Neil Cherry

I haven't have the opportunity to work in C very much lately, but I thought I'd throw in my two bits' worth:

```
NC> #define NEEDED 16*1024
```

This seems to indicate 16 KB needed.

```
NC> unsigned int far *storage;
```

Also implying `int`-sized, not byte-sized data.

```
NC> if (!(storage = (int far *)
    malloc(NEEDED)))
```

Does `malloc` recognize the data type? I thought `malloc` was just bytes.

```
NC> for (addr = begint; addr < endt;
    storage[addr++] = 0xFF);
```

The constant `0xFF` implies byte-sized data.

Are you sure your `begint` and `endt` values are correct?

Now, looking back at it..

Could that be the problem? It looks like you are assuming `addr` is incrementing by 1 and is a byte pointer. However, in actuality, it is a subscript to an integer array, which effectively multiplies it by 2 in the assignment statement.

Also, I have had the problem Ed mentions when working with Turbo Pascal as well. You can't get a full clean 64-KB block, if that is what you really want. I wrote some PROM utility code and worked with two 32-KB blocks.

Msg#:19010

From: Neil Cherry To: Rufus Smith

Let me address all the replies at once here.

```
RR> Not sure Neil, but check your C manuals for a
RR> function like MAKE_FAR() for creating a far pointer
RR> versus the method you are attempting.
```

The TC++ 3.0 manual seems to state that by using the large memory model all pointers are far pointers. But see my comments below to Ed.

```
EN> The catch boils down to 16-bit memory addressing and
EN> the dreaded 64-KB segment limit. When malloc() and
EN> its ilk allocate a memory block for you, they tack on a
EN> small (typically 16-byte) header that identifies the
EN> block and links it to other allocated blocks. Both your
EN> block and the header must lie in the same 64-KB
EN> segment, which means you can't allocate precisely
EN> 64 KB in a single block.
```

The TC++ 3.0 manual states two things in two unrelated places. First is that all pointers, when used with the `Large` memory library, are `far` pointers. Second that TC++ has an option which makes a pointer `far` if it exceeds a threshold. The default threshold is 32 KB. I didn't notice this option until a few minutes ago. Seems to contradict the first part of my comment doesn't it? I wonder if this means I can switch my memory model to `Compact`? Damned bank switching.. ;-}

```
EN> The standard workaround involves allocating a bunch
EN> of smaller blocks and connecting them in a linked list
EN> or array of pointers. Not nearly as nice and you get
EN> to handle data split between blocks all by yourself.
EN> Unfortunately, there's no nice way to allocate a
EN> complete 64-KB block without working around the
EN> malloc() function...
```

Ouch! I guess I've been spoiled by Motorola processors and UNIX. One of the reasons I haven't written the code under UNIX was that I had a heck of a time writing routines to handle the serial I/O. It recently dawned on me that I should structure my I/O routines like that of an interrupt handler under DOS.

Looks like it's also a good time to learn about linked list. This should help with some other features I'm hoping to add to my burner software. Got to love a problem that causes you to need to go out and learn new stuff. Now what was it that started this whole mess? ;-}

```
DW> Ed's comments with respect to 64-KB segments
DW> notwithstanding, your problem is much more basic:
DW> you haven't allocated as much memory as you think!
DW> Your request to malloc() is for 16K bytes, not ints,
DW> so that when you fill the memory with 64 KB of data,
DW> you're overwriting stack, other heap, and/or code.
```

See my comment to Rufus below, it seems that my coding/thinking was very sloppy. I really do have to be more careful.

```
RS> NC> #define NEEDED 16*1024
```

CONNECT TIME

RS>

RS> This seems to indicate 16 KB needed.

In an effort to trim down the code to various limitations (Intel's), I trimmed the size to 16 KB. I initially wrote it to handle 64 KB (at least that was the intention).

```
RS> NC> if (!(storage = (int far *)
                malloc(NEEDED))) /* */
```

RS>

RS> Does malloc recognize the data type? I thought

RS> malloc was just byte.

RS>

```
RS> NC> for (addr = begint; addr < endt;
            storage[addr++] = 0xFF);
```

RS>

RS> The constant 0xFF implies byte-sized data.

RS> Are you sure your begint and endt values are correct?

The data is byte oriented and begint and endt are, for now, 0 and NEED - 1, respectively. I think malloc only understands bytes, but you've pointed out one of my flaws in thinking. What I asked for was a pointer to integers, but what I wanted was a pointer to characters. I'll go correct that in the code.

The purpose for all this is so I can use a IO-year-old EPROM burner I bought as a kit for ~\$100. I originally wrote the code for my Atari ST in BASIC (no GOTOs ;-). My ST's hard drive isn't so healthy anymore and I still want to use the EPROM burner. So I wrote a GWBASIC program for the DOS box.

But, I also want a version that I can run under Linux (the reason for the C code). Since I didn't understand handling serial I/O under UNIX, I dropped back to DOS to burn EPROMs and test out my code. The reason I'll be using Linux is I have a dozen or so assemblers, disassemblers, and other tools for working with microprocessors. It's a rich environment with lots of free stuff. Besides, how am I going to learn anything if I take the easy way every time ;-}.

Thanks everyone.

Need AC source

Msg#: 4385

From: Richard Frantz To: All Users

I need a (simple) circuit that will take a digital signal as input and produce as output an AC power supply, range up to 18 V, for a small inductive load. Any suggestions?

Msg#: 4393

From: Russ Reiss To: Richard Frantz

Give the source of the digital signal (TTL, CMOS, ?) and frequency, duty cycle, and we might be able to help.

Msg#: 4444

From: Richard Frantz To: Russ Reiss

The digital input signal that determines the output voltage of the AC supply is TTL compatible, unless you want something else (I can always adapt). Frequency is 60 Hz, but anywhere in the vicinity is fine.

Msg#: 4463

From: Russ Reiss To: Richard Frantz

I'm starting to comprehend... slowly :)

From your other message, it appears that you have a bunch of bits that represent the magnitude of some voltage which is supposed to come from a power supply. I'm presuming that this PS is run from 60-Hz AC, but that it outputs DC? Or does it output AC? If it's the former, you need a D/A converter (chip) to convert the binary (digital) bits into a varying DC voltage (dependent on the number output). Then you need to interface this signal to a DC power supply.

REMOTE POWER CARD!

3 VERSIONS:

RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

PHONE

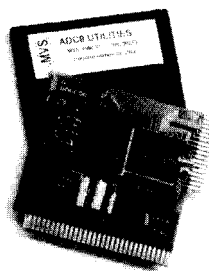
TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER

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

95\$ 27\$ OEM

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



8 CHAN ADC

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

95\$

2 CHAN DAC

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

75\$

MVS MVS BOX 850 MERRIMACK, NH (508) 742-9507

5 YEAR LIMITED WARRANTY P I N G I N U S A

CONNECT TIME

Perhaps the simplest (though not the most efficient) way is to use a pass transistor. Feed some voltage a few volts higher than the maximum you'd ever like to see at the output to the collector. Feed the buffered DAC output (buffer needed for drive) to the base. Take the output from the emitter. This arrangement is also known as an "emitter follower" configuration.

The output will be slightly less than the signal you feed to the base, due to base-emitter voltage of the pass transistor. This drop can be corrected by feeding the output voltage (from the emitter) back into a "correcting amplifier" (perhaps the same "buffer" I mentioned above) to raise the base voltage until the output equals the output of the DAC.

Well, that's the theory. If you were looking for a finished schematic or built-up unit, sorry I cannot help. But perhaps this will get you to first base, and you might learn a lot along the way. If you choose to go this route, I'm sure others will jump in and help with specific questions as they arise. Good luck!

Msg#: 4471

From: Richard Frantz To: Russ Reiss

My description is getting clearer. I need an AC output, approximately 60-Hz sine wave (but it can be incredibly noisy and still work) to drive a small AC motor (I don't get to pick the motor, it's already there; I just want to control it). I've got a binary number describing the output voltage I want. I'm looking for advice on how to convert binary to an AC signal. I'm not overjoyed with the designs I've come up with [I'm a good programmer, and I'm OK with digital logic, but I'm not very good with analog circuitry].

One solution I came up with produced a square wave at a programmable voltage which I could then filter (to near sine wave) and amplify for drive. I just wasn't very happy with the way it looked on paper and figured there was probably a better way.

Thank you for your suggestions.

Msg#: 4479

From: Russ Reiss To: Richard Frantz

OK! Now I understand what you need. Sorry. If you have multiple binary outputs, you can simply store a table of (say a quarter of) a sine wave, play these out with proper manipulation on the digital port, feed to a D/A converter, and obtain a nice sine wave with a little filtering. Then you'd have to amplify it to drive the motor—some sort of power op-amp of sorts. That's the direct approach.

You could also use a single output bit, as you mentioned, to produce a square wave at 60 Hz. This will contain quite a bit of 60-Hz energy, plus decaying third harmonics of that frequency, which can be filtered by a low-pass filter, amplified, and fed to the motor.

Alternatively, you can use PWM. Here, you generate a higher frequency (not too critical, whatever your software can handle) digital wave. You control the duty cycle (on/off ratio) and vary it in a sinusoidal manner. You increase the duty cycle to get a higher average voltage level, and decrease it to reduce the level. Using the right software, you can make the average level change sinusoidally at any frequency you chose, such as 60 Hz.

Now, with only 0- and 5-V levels, you would set the baseline at 2.5 V, or 50% duty cycle, and vary around there. You'd have to pass this through a capacitor to eliminate this 2.5-V bias from the signal, resulting in a 5-V p-p sine wave centered around 0 V. You'll need an op-amp with positive and negative supplies for that. You can set the gain of the amplifier to adjust the level to whatever you need, amplify it (or just use a power opamp), and feed that to the motor. Oh, and you should include some lowpass filtering to get rid of the PWM frequency. But this can be quite high, and is easily filtered, much more simply than the harmonics of the square wave.

It all depends on the degree of signal purity you desire and your software expertise. Some microcontrollers have built-in PWM channels, which makes life simpler. Why not try the square wave approach (you'll also need to eliminate the DC bias from it by AC coupling) and see if that's adequate? If not, try the PWM approach.

Msg#: 4483

From: Pete Chomak To: Richard Frantz

I'm not sure what you have for processing horsepower (if any) behind your digital output bits, so perhaps you want a solution that is self contained (i.e., controlled by a separate processor or dedicated hardware).

There are some inexpensive chips that are used for VCR capstan motor control that might be adaptable to your application, perhaps with a DAC to drive the analog speed input to the chip and suitable optoisolators and power devices from the outputs. You could also do something similar with a PIC processor or equivalent.

As far as reference materials go, one of the best would be the maintenance manual for one of the Fanuc AC servo drives (used on CNC machine tools). You might be able to photocopy one from a local machine shop. Those servo controllers drive AC motors from 1 hp to 30 hp and can position them to a minute fraction of a degree. You can learn quite a bit by looking at the manual.

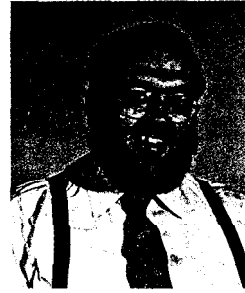
Msg#: 4496

From: James Meyer To: Richard Frantz

Pardon the interruption, but I think if you restate your objective, the solution will jump right out at you. It did for me, anyway.

PRIORITY INTERRUPT

It Just Frosts My Chops



Unlike a lot of magazine editorial directors, I don't generally use this space as a soapbox for political or social issues. I find it much more interesting just staying on the technical side of political center and keeping my spiel event driven. Unfortunately, a couple of unseemingly related issues have set off my alarm bells.

I've been coming to a slow boil listening to all the rhetoric about a flag-burning amendment. Now, the latest press about censoring the messages and graphics transmitted via the Internet really frosts my chops.

The problem is not that I'm some "free speech or die" advocate. But, the complete inability of our legal system to write a definitive law uncompromised by endless legal interpretations and personal ambitions should be a warning to all of us against believing that there is a simple solution-like passing a law. Isn't the freedom that gives us such liberty and independence of expression worth more than protecting a mere symbol of a diminished freedom?

Given the track record, any feel-good legislation presented by Washington would either be so restrictive that it would redefine the U.S. Constitution or so full of interpretation that it would be nothing more than a full employment program for lawyers.

Consider how certain flags and symbols would be protected. Is torching a 49-star flag illegal? Are embroidered flag patches now to be worn only above the waist or not at all? If a picture of the flag is proudly represented across the cover of a magazine, can that periodical only be destroyed by burning it while singing the national anthem?

The fact that there is now discussion among the same ranks that would bring you this absurd legislation about censoring Internet traffic should strike fear in the heart of every modem user in the country.

Unfortunately, obscenity is too easily used as the driving motive for legislation whose real long-term motive might merely be to eavesdrop on the community. Remember the clipper chip?

Any truly feasible censorship law would make it the network providers responsibility to monitor all data and graphics. In fact, some on-line services have already started the monitoring game. One provider's definition of obscene was apparently defined as "that which shocks an elderly grandmother" because one even disallowed the entry or display of the word *breast* throughout its service. Both food recipes and medical discussions took an interesting turn.

The end result is that unless we collectively head off the dim-witted thinking that government intervention and censorship are tools to preserve a free society, we are destined to lose a society and freedom worth preserving. Censorship, sooner or later, is only a weapon directed against freedom of thought.



steve.ciarcia@circellar.com