EMBEDDED PC MONTHLY SECTION

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#87 OCTOBER 1997

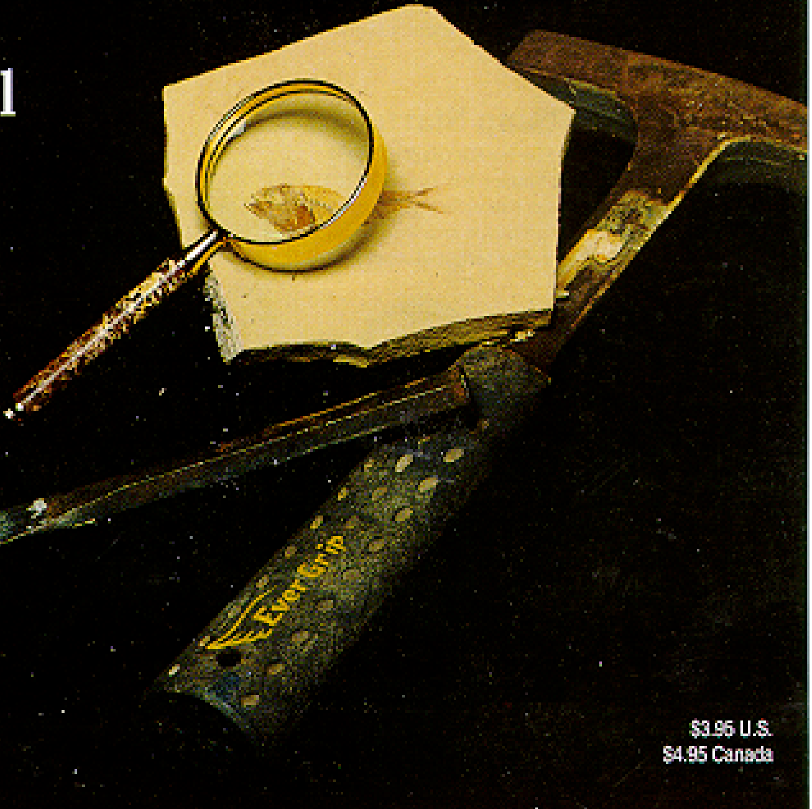
DATA ACQUISITION

DTMF Decoders for the Hearing Impaired

Forcing DOS Code onto a Windows GUI

New PIC—Doing It All with Eight Pins

EPC: Microsecond Timing from an Embedded PC



$3.95 U.S.
$4.95 Canada

TASK MANAGER

And the Survey says...

In our business, when someone talks about collecting data, we immediately start coming up with solutions for how we're going to collect the data. Battery operation, low-power sleep mode, wake-up timers, nonvolatile memory.. .the list goes on. However, when it comes right down to laying out the task, what we collect and what we do with it inevitably overshadows how we do it.

We recently conducted a survey of a random sampling of Circuit Cellar *INK* readers. The "how" was obvious: mail a sheet of paper to a list of names and wait for the results to come back. The "what" was a whole lot more difficult, and we spent many hours coming up with questions for the survey.

Once the raw data had come in, though, the task grew in size again. How do you meaningfully compile, combine, and interpret the results into something useful? It's far too easy to skew the numbers or compare apples with oranges and end up with meaningless answers.

After going through all these convolutions, what did we find? It turns out we know our readers pretty well, and most approve of what we're doing. For more information on some of what we learned about you, the reader, turn to Priority Interrupt and read what Steve has to say.

Moving from the "what" back to the "how," Damon Chu starts this Data Acquisition issue with a look at the newest members of the PIC family and how they are ideally suited to many low-power data-acquisition systems.

Next, Steven Kraft helps out a hard-of-hearing grandmother with a messaging device that works with ordinary phone lines using DTMF and without complicated equipment.

Craig Pataky next takes us on a trip through a wormhole to connect stalwart DOS code with a flashy Windows 95 front end. The result is reliable, well-tested code that meets the expectations of today's desktop user.

Finally, Bill Jackson and Reynaldo Archide show how a RISC processor can be superreduced, resulting in an even smaller instruction set and faster speeds.

In our columns, Ingo Cyliax continues the development of his MC68030 trainer board by going over the monitor and boot code. The board even boots over a network. Jeff explores the latest in low-cost prototype PC boards and finds the days of point-to-point wiring and wire-wrapping are drawing to a close. Lastly, Tom strolls into the analog camp again with a new programmable analog array device.

Embedded PC starts this month with Jim Blazer, Janos Levendovszky, and Robert Haris describing a distributed approach to data acquisition. It lets the main processor do the real computing, while the chores of collecting the data are handled by another processor. Steve Lisberger follows suit by unveiling a design that times events with microsecond precision and minimal impact on the main processor.

In the ***EPC* columns,** Dennis Liles and Rick Lehrbaum introduce EBX, a new open standard in high-power, compact, embeddable form factors. Fred Eady wraps up his NASA plant-growth chamber with a look at remotely collecting data and displaying it over the Internet.

[illegible]

editor@circuitcellar.com

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

EDITORIAL DIRECTOR/PUBLISHER
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

MANAGING EDITOR
Janice Hughes

TECHNICAL EDITOR
Elizabeth Laurençot

ENGINEERING STAFF
Jeff Bachiochi

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Fred Eady

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
KC Zienka

PRODUCTION STAFF
John Gorsky
James Soussounis

ASSOCIATE PUBLISHER
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION CONSULTANT
John S. Treworgy

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Valerie Luster

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0696.6965) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on US bank.

VISIT OUR WEB SITE FOR SUBSCRIPTION INFORMATION AT www.circuitcellar.com

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.

Cover photograph Ron Meadows – Meadows Marketing
PRINTED IN THE UNITED STATES

For information on authorized reprints of articles, contact Jeannette Walters (660) 675-2199.

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC	MIDWEST & SOUTHEAST	WEST COAST
Barbara (Best) Curley	Christa Collins	Barbara Jones &Shelley Rainey
(561) 694-2044	(954) 966-3939	(714) 540-3554
Fax: (561) 694-2051	Fax: (954) 965-6457	Fax: (714) 540-7103
B.Best-Hajar@worldnet.att.net	HajarChrista@worldnet.att.net	shelley.hajar@worldnet.att.net

Circuit Cellar BBS—24 Hrs., 2400/9600/14.4k bps, 6 bits, no parity, 1 stop bit, (860) 671-1966. For information, mail to info@circuitcellar.com. World Wide Web: www.circuitcellar.com

INSIDE ISSUE 87

12 Analog Data Acquisition
Damon Chu

18 DTMF Message Decoders
Telephone Aids for the Hearing Impaired
Steven Kraft

26 Interprocess Communication
Moving DOS Programs into Windows
Craig Pa taky

30 Compressed-Code **TinyRISC**
Bill Jackson & Reynaldo Archide

66 ❑ **MicroSeries**
MC68030 Workstation
Part 2: The Boot PROM Monitor & Device Drivers
Ingo Cyliax

74 ❑ From the Bench
Prototypes of the Rich and Famous-Not!
Jeff Bachiochi

78 ❑ Silicon Update
A(r)Ray of Analog Hope
Tom Cantrell

Task Manager 2
Ken Davidson
And the Survey Says...

Reader I/O 6

New Product News 8
edited by Harv Weiner

Advertiser's Index 95

Priority Interrupt 96
Steve Ciarcia
The People's Contest

EMBEDDED PC

34 Nouveau PC
edited by Harv Weiner

39 Intelligent Data Acquisition
Jim Blazer, Janos Levendovszky, & Robert Haris

44 Precision Timing and I/O
Steve Lisberger

56 PCQ **PC/104** Quarter
PC/104 Steps Out of the Box
Dennis Liles & Rick Lehrbaum

60 APC Applied PCS
Managing a NASA Plant-Growth Chamber
Part 2: Coordinating Sensors and Analyzing Data
Fred Eady

www.circuitcellar.com

READER I/O

OTHER ANGLES ON THE REMOTE EDGE

Daniel Patten and Michael Miller's "A Universal IR Remote-Control Receiver" (INK 82) was very useful. I'm building a TV remote by adding an IR LED and PIC16C54 to an old calculator. Since the calculator has a spare slide switch, I'm including my NEC VCR's functions. I measured NEC and nonNEC remote codes with a photodiode and amplifier connected to a bit on a PC printer port.

My NEC-remote code patterns agreed exactly with the timing values in the article. So, I was surprised that Joe Zhu (Reader I/O, ***INK 85)*** claims the NEC chip sends two code patterns per key press. I did not observe this.

I think the remote's "repeat" function is the issue. If a key is pressed and held down, each remote continuously sends a train of codes at 10 codes per second (100 ms between sync pulses). The nonNEC remote repeats the same code until the key is released.

On the NEC remote, the first code sent is the unique key code described in the article. All remaining codes sent are a fixed, nonstandard code that's the same for all keys and apparently means "repeat the last command." The code is nonstandard because it doesn't have the normal off time following the sync pulse.

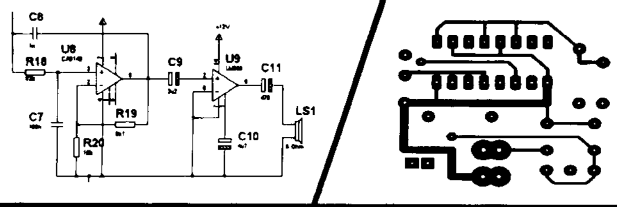
The repeat code is: 9 ms on, 2.3 ms off, 0.6 ms on, no additional bits following. The interval between the end of the first code and the beginning of the repeat code is 100 ms (length of key code). Since each bit in the key code is also transmitted as its complement, all bits take the same time to send (1.2 + 2.3 = 3.5 ms). The key-code length is then: 9 + 4.5 + 16(3.5) = 69.5, leaving a ~30-ms interval. The repeat period is a nominal 100 ms but was observed to be 104 and 108 ms in the two remotes, so Mr. Zhu's value of 40 is consistent with normal variations.

I also saw the stop pulse he refers to and originally interpreted it the same way he did. But, if bits are on times followed by off times, then a final stop pulse is needed to terminate the last off time.

But, suppose a bit is a variable off time followed by a fixed on time. The code sequence is: sync pulse (long on), mode pulse (long off followed by the first 0.6-ms on pulse), then 32 bits (off followed by on). The final on pulse is part of the last bit. Now, the mode bit distinguishes a normal data word (4.5 ms off) from a repeat code (2.3 ms off].

John N. Power
jpower@mail.bcpl.lib.md.us

THIS IS THE CAD FOR YOU!!!
"Easy to use, designed to help"



Now in
Both Windows and DOS

CA D PA K *Windows-$199, DOS - $159*

Ideal for New Users, Hobbyists, & Small Businesses!
Provides everthing for PCB Layout & Schematic Drawing

P R O T E U S *Starts at $425*

Most Powerful EDS System on the market!!!

* Schematic Capature * Circuit Simulation
• PCB Layout with Rip-Up & Retry Autorouting

R4 SYSTEMS INC.
1100 GORHAM ST. Suite 11B-332
NEWMARKET ONTARIO
CANADA L3Y7V1
905 898-0665 FAX 905 898-0683
BBS 905 898-0508 (9600,8,N,1)

TRY OUR FREE DEMO WRITE OR CALL TODAY

Download Demo - http://www.r4systems.on.ca
Internet email info@r4systems.on.ca

ALL POINTS BULLETIN !! BE ON ALERT FOR THE

PIC® KILLER! OEM (1K) PRICE $1.99 EVAL KIT 7.0

A LOWER COST, FASTER, EASIER TO PROGRAM SINGLE CHIP COMPUTER

COMPARE:	16C54	MV1200
DEM (1K) PRICE	$2.57	$1.99
PROGRAM DOWNLOAD	NO	YES
SINGLE CHIP OPERATION	NO	YES
BUILT-IN BASIC	NO	YES
EEPROM DATA MEMORY	NONE	64
PROGRAM MEMORY	768 OTP	1K FLASH
MATH REGISTERS	1	32
MAX INSTRUCTIONS/SEC	5M	ZOM
MAX COUNTER BITS	16	18
INPUT/OUTPUT BITS	12	15
A TO D COMPARATOR	NO	YES
HARDWARE INTERRUPTS	NONE	3

PINOUT:

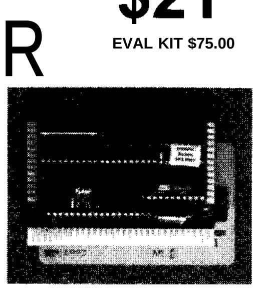
RESET	1	20	VCC
PDO	2	19	PB7
PD1	3	18	PB6
XOUT	4	17	PB5
XIN	5	16	PB4
PD2/INT	6	15	PB3
PD3	7	14	PB2
PD4/TMR	8	13	PB1/AD1
PD5	9	12	PB0/AD0
GND	10	11	PD6

® PIC IS A REGISTERED TRADEMARK OF MICROCHIP CORPORATION

CREDIT CARD COMPUTER

OEM (1K) PRICE
$21
EVAL KIT $75.00

-**SER** PROGRAM DOWNLOAD
-**256kb** NONVOLATILE MEM.
-LCD/KEYPAD INTERFACE
-ISA BUS (MODEM,MEM,ETC.)
-**RS232/485** SERIAL TO **115.2KB**
-ASSEMBLY/BUILT-IN BASIC
-100 **MICROAMP** OPERATION



MVS BOX 850
MERRIMACK,NH 03054
(508) 792-9507

MVS

5yr Limited Warranty
Free Shipping
Hrs: Mon-Fri 10-6 EST

#10

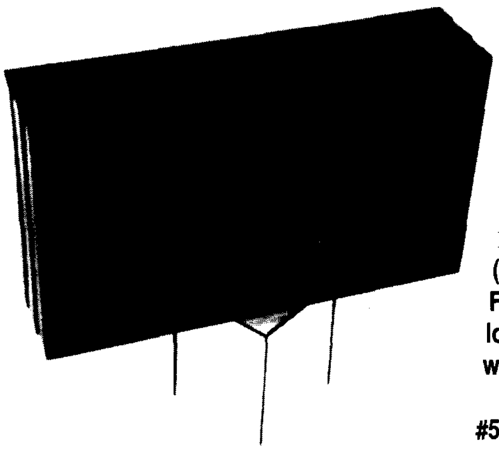
NEW PRODUCT NEWS

Edited by Harv Weiner

DATA LOGGING SYSTEM

The **ModuLogger** is a battery-powered, self-contained portable data-logging and alarming system designed for sampling and storing flow, pressure, temperature, current, power, and other process, vehicle, and utility signal data over time. After on-site collection, the ModuLogger's memory can be serially downloaded via modem or RS-232 link to a PC. With the provided HyperWare for Windows software, data can be further manipulated, plotted, and/or converted to various spreadsheet formats.

The ModuLogger accepts up to four universal analog-type inputs, which are software configurable for six thermocouple types, 15 ranges of DC voltage, or seven ranges of DC current. Another analog channel is used for cold-junction compensation for thermocouple applications. A self-calibration feature includes user-programmable self-calibration cycles. A single software-configurable general-purpose digital input logs events or counts digital outputs from flow meters, encoders, or other pulse-train sources. Isolated alarm relay outputs and a TTL alarm output are included.



The ModuLogger system starts at $995.

Logic Beach, Inc.
8363 Center Dr., Ste. 6F
La Mesa, CA 91942
(619) 698-3300
Fax: (619) 469-8604
loggers@logicbeach.com
www.logicbeach.com

#501

DATA-ACQUISITION CARD

UEI has introduced a family of data-acquisition cards featuring the **PowerDAQ** PCI interface. PowerDAQ is a DSP-enhanced PCI-interface subsystem incorporating a Motorola DSP56301 processor with an integrated PCI controller linked via internal data bus to system logic and onboard memory. Because the PowerDAQ interface functions as a multithreaded processor, a fatal error in one process does not terminate operations running simultaneously. This interface transfers data across the PCI bus at rates of 132 MHz-much faster than the data-acquisition card's maximum sampling speed.

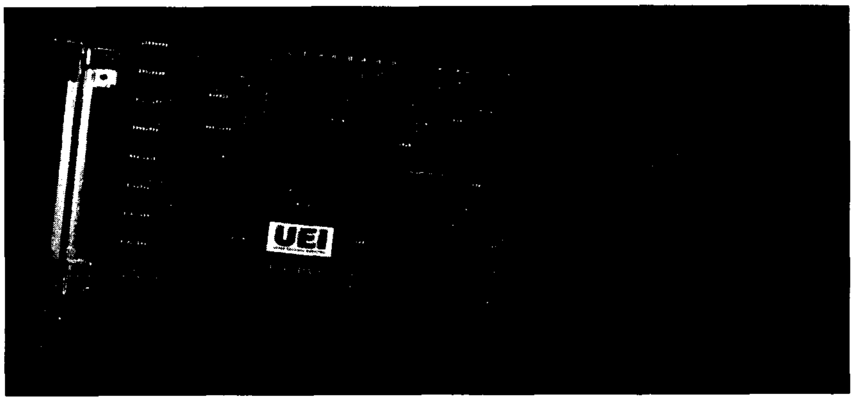
The first member of this data-acquisition family—the PCI-PD-1M16—offers 16 single-ended analog input channels sampling (continuously) at 1 MHz with 12-bit resolution, two 12-bit analog output channels, eight high-speed digital input lines, eight high-speed digital output lines, and three user-accessible counter/timers.

Each PowerDAQ card comes with a set of UEIDAQ software that includes a VxD for Windows 95 and kernel-mode drivers for Windows NT. These drivers support all relevant PCI features, including PCI-bus PowerDAQ detection, true plug-n-play support, bus-mastering block data transfers, multiple simultaneous command requests from concurrent Win32 application threads, and concurrent request processing (multiprocessor systems).

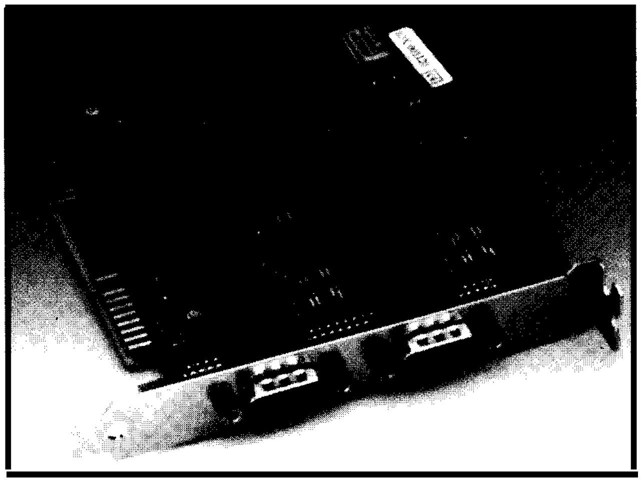
Pricing for the PowerDAQ starts at $1650.

United Electronic Industries, Inc.
10 Dexter Ave.
Watertown, MA 02172
(617) 924-1155
Fax: (617) 924-1441
sales@ueidaq.com
www.ueidaq.com

#502



NEW PRODUCT NEWS



ULTRA-HIGH-SPEED PC COM PORT

The Telebyte Model 480 dual-port, ultra-high-speed serial-I/O card enables high-speed serial data connectivity for ISDN and other technologies. The card can be used in either ISA- or EISA-bus-based PC systems in any application where the standard speed from COM1 or COM2 is insufficient. By using 16C650 UARTs, the Model 480 can support speeds up to 460 kbps on both serial ports. System overhead is reduced via 32-byte buffers. The card can be mapped to any location from COM1 to COM4 and use standard COM port drivers. Custom drivers are supplied to get the full benefit of the available performance.

Using the Model 480 in combination with software communications packages (e.g., pcAnywhere or Reach-Out), users can effectively access a remote PC and perform tasks on the remote machine as if it were a local PC. The card sells for $79.

Telebyte Technology, inc.
270 Pulaski Rd.
Greenlawn, NY 11740-1616
(516) 423-3232
Fax: (516) **385-8184**
telebyteusa.com

#503

TIME GENERATOR

The PCI-SG synchronized time generator provides precise time derived from internal or external references (with zero latency) to computers with PCI expansion slots. This system enables a PC to be an accurate and reliable time and synchronization source for many business, industrial, and scientific applications. The PCI-SG can also supply external timing to other PCs or devices requiring accurate time.

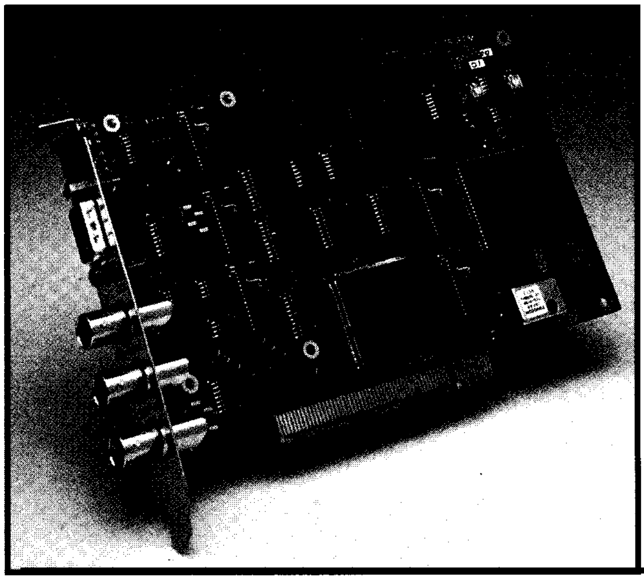
The PCI-SG derives time from industry-standard external sources (e.g., GPS or IRIG time codes). Using GPS as the reference source, the card provides timing accuracy to within 1 µs of Universal Coordinated Time. This order of accuracy is useful in time tagging data for comparison with data from another source.

The advantage of the PCI-SG over the PC clock is the accurate time made available to the host PC and external devices. Accuracy is maintained during any level of system activity, hardware interrupts, or low-level processes. External connectors on the card output IRIG B time codes, 1 pps, and a variety of programmable pulse rates. These outputs are useful for synchronizing other computers and peripheral devices as well as for passing time-code information to other computers.

The PCI-SG is priced at $1295.

TrueTime, Inc.
2835 Duke Ct.
Santa Rosa, CA 95407
(707) 528-I 230
Fax: (707) 527-6640
www.truetime.com

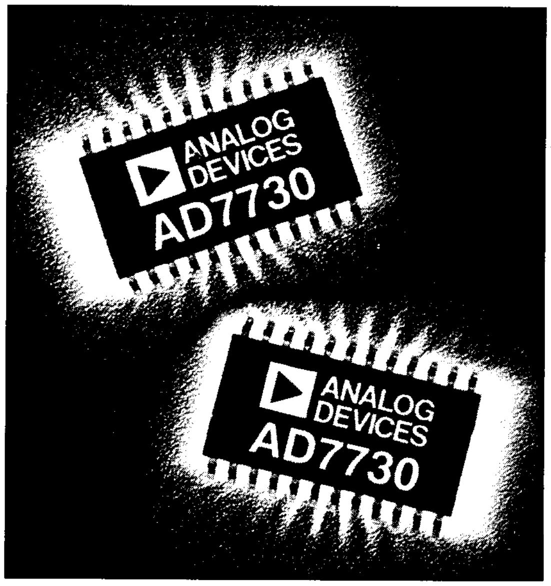
#504



NEW PRODUCT NEWS

TRANSDUCER A/D CONVERTER

The AD7730 is a high-resolution analog front end for weigh-scale and pressure measurement applications. Operating from a +5-V supply, this 24-bit sigma-delta ADC accepts low-level analog signals from a transducer and outputs digital words.



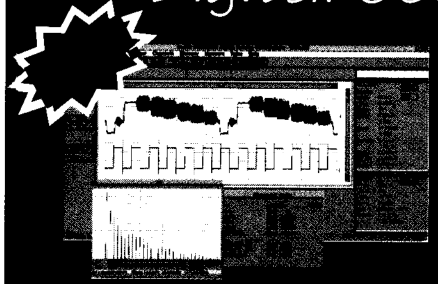
The AD7730 provides a resolution of 220,000 counts, peak-to-peak. It contains self- and system-calibration options and provides a unique chopping scheme that results in a typical offset drift of 5 nV/°C and a maximum gain drift of 2 ppm/°C. Featuring two buffered differential programmable-gain analog inputs, the chip accepts eight analog input ranges. An on-chip, 6-bit DAC for removing tare voltages and clock signals for synchronizing AC excitation of the bridge are also provided. If large weight changes occur on the load cell, the AD7730's FASTStep mode closely approximates (by switching between internal filters) the final result in one-eighth of the final output settling time. Its serial port can be configured for three-wire operation and is compatible with microcontrollers and DSPs.

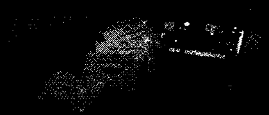
The AD7730 sells for $9.86 in 1000-piece quantities.

Analog Devices, Inc.
One Technology Way
Norwood, MA 02062-9106
(617) 937-1428
Fax: (617) 821-4273
www.analog.com

#505

Digital Oscilloscopes Logic Analyzers





- 2 Channel Digital Oscilloscope
- 100 MSa/s max single shot rate
- 32K samples per channel
- Advanced Triggering
- Only 9 oz and 6.3" x 3.75" x 1.25"
- Small, Lightweight, and Portable
- Parallel Port interface to Laptop or PC
- Advanced Math, TV Line Trigger and FFT Spectrum Analyzer options

$499

- 200 MSa/s max single shot rate
- 2 Oscilloscope channels
- 8 Logic Analyzer channels
- 10 Channels simultaneously
- 125 MHz Single Shot Bandwidth
- up to 128K Samples/Channel
- FFT and Spectrum Analyzer

DSO-2[illegible] $1999
DSO-2[illegible] $3799
All prices include Probes and Software

- 40 to 160 channels
- up to 500 MHz
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- up to 512K samples/ch

[illegible]	$1350
[illegible]	$2000
[illegible]	$1900
[illegible]	$2800
[illegible]	$7000

All prices include Pods and Software

For $499 you get the model DSO-2102 Scope, Probes, Interface Cable, AC Adapter, and Windows and DOS Software.

Link Instruments (201) 808-8990
369 Passaic Ave • Suite 100 • Fairfield, NJ 07004 • Fax (201) 808-8786
Web: http: www.LinkInstruments.com cci9 · Email: Sales@LinkInstruments.com

#104

NEW PRODUCT NEWS

INDUSTRIAL COMPUTER

The **IND-600** is a PC that combines a 10.4" TFT or STN color display with a plug-in industrial CPU board in a rugged rack/panel-mount enclosure. It can be equipped with a cost-effective '486DX, a midrange '586, or a Pentium microprocessor for maximum performance.

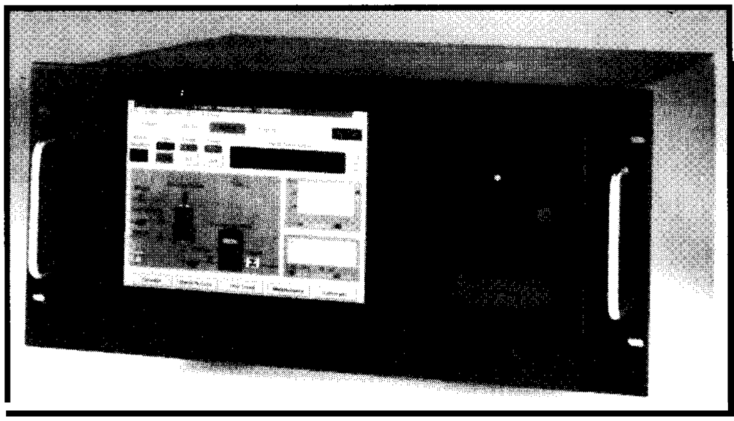
Standard features include a 1.2-GB hard drive, 1.44-MB floppy drive, and 4-MB DRAM (expandable to 128 MB). It also has two serial ports with 16550 UARTs, watchdog timer, keyboard interface, and a high-performance parallel port supporting SPP/EPP/ECP modes. A field-replaceable 250-W power supply is built in. Four ¾-length ISA-bus slots provide room to install network and data-acquisition boards. The front panel is fitted with an impact-resistant Lexan window to ensure reliable operation-even in harsh environments. Shock-mounted disk drives and adjustable board-hold-down brackets eliminate the effects of shock and vibration.

MS-DOS is preloaded on the hard drive, and Windows is available as a factory-installed option. This combination provides easy access to all industry-standard development tools and industrial or scientific application software packages.

IND-600 pricing starts at $2788.

Indocomp Systems, Inc.
5409 Perry Dr. • Waterford, MI 48329
(248) 673-7315 • Fax: (248) 673-8370
www.indocomp.com

#506



Looking to fill your embedded toolbox without breaking the bank???

DDS COMPLETE

Everything you need, all in one convenient package! Includes all of our Micro-C Compilers, Assemblers, Simulators, Disassemblers, Monitors, Project Plans, and Data Line Monitor. Many CPUs supported, including C-FLEA, a virtual CPU you can use anywhere.

Package price: $ xx? US ($ 650 CDN).

Call or write for our free catalogue .

FREE Demos available on BBS or WEB (or rend $5 for diskette)

*We accept **Cheque/MO/Purchase Orders/VISA***

Dunfield Development Systems

P.O. Box *31044* **Nepean,** Ont. **K2B 8S8** CANADA
Tel: 613-256-5820 Fax: 256-5821 BBS: 256-6289
info@dunfield.com, sales@dunfield.com,
http://www.dunfield.com

We know INK...

STANFORD • RYERSON POLYTECHNIC • ROCHESTER INST. OF TECHNOLOGY • TUFTS • HARVARD • PURDUE • DEVRY INST. OF TECHNOLOGY • PENN STATE • CORNELL • CALIFORNIA STATE • U.S. NAVAL ACADEMY • MILWAUKEE SCH. OF ENGR. • CASE WESTERN RESERVE • MERCER • AALBORG • AMARILLO C. • APPALACHIAN • MIT • STATE U. • U. OF ARIZONA

Do You?

If you're an engineering professor or teach qualified technical students, we'd like to give you and all of your students free subscriptions to Circuit Cellar INK. Contact:

Rose Mansella • Circulation Coordinator • Circuit Cellar INK
4 Park St. • Vernon, CT 06066
(860) 875-2199 • Fax: (860) 871-0411
rose.mansella@circuitcellar.com • www.circuitcellar.com

FEATURES

12 Analog Data Acquisition

18 DTMF Message Decoders

26 Interprocess Communication

30 Compressed-Code TinyRISC

FEATURE ARTICLE

Damon Chu

Analog Data Acquisition

As computer components evolve, chips get smaller, faster, and more integrated. Take the PIC12C672. In eight pins, Microchip brings us 2048 words of PROM, 128 bytes of user RAM, and advanced features like on-chip ADC and four analog channels.

Many systems now demand data-acquisition functionality. A microcontroller-based design enables the measurement, processing, control, and communication functions required by these applications, while keeping system costs low.

In this article, I discuss how a new microcontroller can create the foundation of a sophisticated data-acquisition system. Taking a home-security system as an example, I demonstrate how to use the MCU's feature set for maximum functionality.

But, let's start with a brief overview of this new 8-pin, 8-bit MCU.

ITS BITS AND BYTES

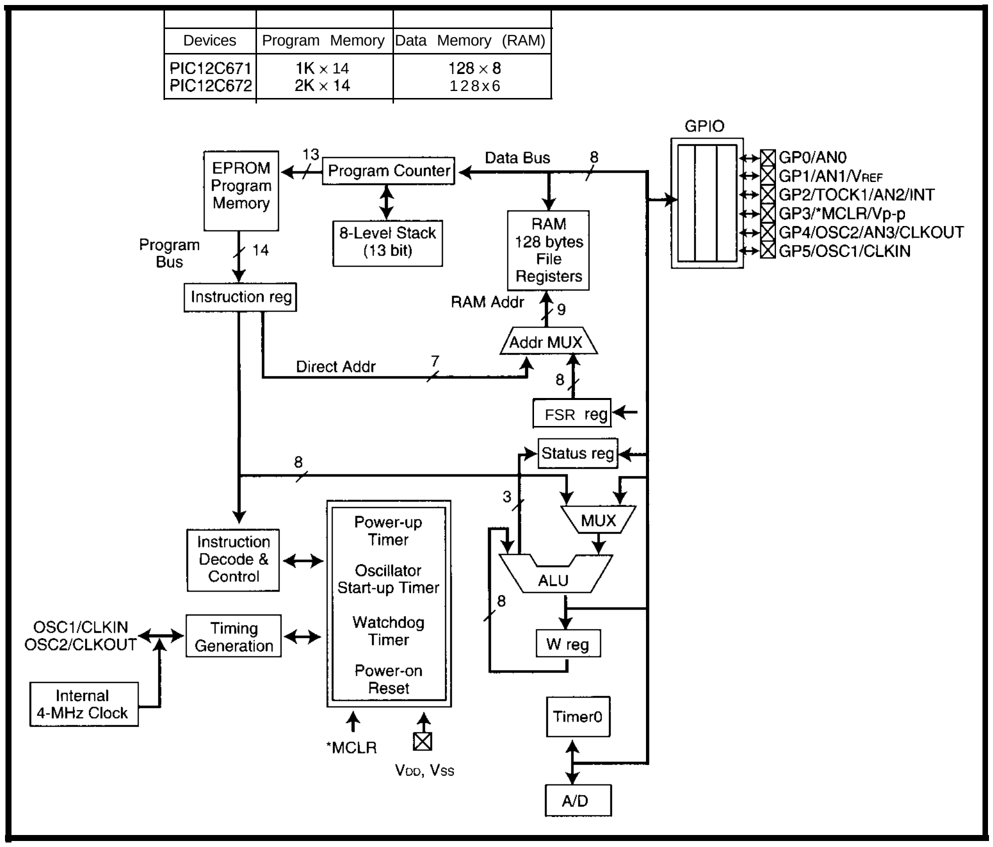
Depicted in Figure 1, the PIC12C672 features 2048 words of program memory along with 128 bytes of user RAM. Advanced analog features include an on-chip ADC and four analog channels, which can be used for measuring environmental conditions (e.g., temperature, pressure, motion, and voltage).

The device offers five multiplexed I/O pins (plus one input only) with on-chip clock oscillator (4 MHz), 35 single-word instructions, full-speed 1-ys instruction cycle at 4 MHz, and an eight-level-deep hardware stack.

It also includes an 8-bit clock/counter with 8-bit programmable prescaler, watchdog timer, direct LED drive, low 2.5-5.5-V operating voltage, and under 2 mA at 5-V, 4-MHz low-power consumption. In-circuit serial programming of the OTP controller offers a true, self-contained intelligent system on chip (see Photo 1).

Despite its small packaging, it offers high-performance RISC functionality.

This combination makes the PIC12C672 particularly appropriate for many data-acquisition applications in which the environment is being monitored and/or measured in a variety of products. The MCU can also provide a high-performance, low-cost replacement of many electromechanical applications.



Devices	Program Memory	Data Memory (RAM)
PIC12C671	1K x 14	128 x 8
PIC12C672	2K x 14	128x6

Figure 1 --The *PIC12C672 provides* advanced *analog* features, including an on-chip ADC *and four* ***analog*** *channels. The newest* family of ***8-pin*** B-bit microcontrollers *offers* 1024-2048 *words of program memory along with* ***128 bytes*** of user RAM. These *devices feature six multiplexed I/O pins with on-chip clock oscillator (4 MHz).* ***(Note*** *that higher order bits are from the STATUS register.)*

DESIGNING IT IN

Let's look at how the PIC12C672 can be used in a home-security system. Since my system is AC powered, its power supply provides the 5 V for the PIC12C672.

I chose the PIC's internal 4-MHz clock oscillator as the system clock. Its use not only eliminates the space and cost of an external clock oscillator but also frees pins 2 and 3 to be general-purpose I/O pins.

The internal clock oscillator is selected by setting bits FOSC2:0 in the configuration word to a binary value of 100. The configuration word can be set up during the in-circuit serial-programming process.

For measurement, the PIC 12C672 has an onboard 8-bit ADC. A resolution of 256 steps is sufficient for most sensor needs in a home system, especially given that the typical home thermostat has a range of 45–95°F, requiring measurement of 50 steps at a resolution of 1°F. With the ability to configure four channels of analog input, the ADC can measure four sensors while the processor is in the power-down sleep mode.

In my system, the PIC12C672 measures temperature and carbon-monoxide concentration. Two of the six I/O pins are configured as analog inputs by setting bits PCFG2:0 in the ADCON1 register to the binary value of 100, establishing pin 7 as analog input channel 0 (AN0) and pin 6 as analog input channel 1 **(AN1**), so now, analog measurements can be taken.

Configuring the ADCONO register to the hex value of Cl opens channel 0 for measurement, selects the ADC's own clock, and turns on the converter. Interrupts are enabled by setting the appropriate bits in the INTCON and PIR1 registers.

Analog conversion is started by setting the GO bit in ADCONO. Once a conversion completes, an interrupt signal is generated, and the device begins processing the input measurement. Listing 1 shows code for an A/D conversion.

APPLICATION TUNING

For power-conscious systems, the A/D conversion can take place while the rest of the PIC12C672 is asleep. To make this change to the code, simply add the S 1 eep instruction after starting the A/D conversion. The microcontroller wakes up once the conversion is complete and the A/D interrupt occurs.

For processing, the PIC12C672 is a well-equipped engine. With 2048 x 14 words of program memory storage and 128 bytes of user RAM, the security-systems designer has sufficient memory resources to implement averaging routines.

All system RAM locations reside in the register file and are available for every CPU cycle, so all registers are available for data manipulation on every cycle.

For sensor readings, the ADDW F and RR F instructions can perform addition and division by powers of 2. Each instruction executes in 1 µs when using the internal 4-MHz system clock oscillator. Its eight-level-deep hardware stack supports a number of nested loops.

If you're using the PICmicro architecture, it's a good idea to set aside a section of program memory to store look-up table values. These table values can be accessed via CA L L and RET LW instructions by sending the program counter into the table at a specified location and returning with the look-up value. Thus, the processed result can be matched to the look-up table value.

For example, that match might mean a certain toxic-gas ppm level is reached and action must be taken. Or, in the case of measuring IR radiation, after comparing the measured and processed data to the table look-up value, the heat level could determine whether human intruders are in the house or if it's just the cat arriving home.

After the chip has executed the processing algorithms and made its comparison to table look-up values, action may be required. Control may be as simple as turning on the air conditioner if the temperature is above the nominal setting or switching on a bright light if intruders are detected.

The PIC12C672's multiplexed pins can be configured through software to provide up to six digital lines-five bidirectional and one input only. Here, pins 6 and 7 are being used for analog measurements, and pins 2-S are available for general-purpose DIO functions. As the six pins are highly multiplexed, the GPIO register must be set up properly to establish whether the pins provide I/O or non-I/O functions.

Once the pins' functionality is chosen, the bidirectional I/O pins must have their direction defined via the TRIS register. A logic 1 from the TRIS register bit puts the corresponding output driver into high-impedance mode, allowing it to be a digital input. Conversely, a logic 0 puts the contents of the port's output latch on the selected pins, enabling the output buffer.

For outputs that are enabled, each of the output drivers provides 25 mA of drive current. That level is sufficient for turning on/off power transistors or triacs and for lighting bright LEDs.

This function can provide system recovery in the event of a software malfunction, and it can run during Sleep mode. It's set up during the programming process by setting WDTE (watchdog timer enable bit) in the configuration word.

In addition, pin 2 can be configured as a clock input (TOCKI), enabling synchronization with an external system clock using TMRO. TMRO is a 16-bit overflow counter with an 8-bit programmable prescaler. An overflow of TMRO can interrupt the processor during operation.

Finally, the PIC12C672 can be reset via an external RESET signal or by various timeouts built into the microcontroller. Of course, how you use these features depends on the complexity of your system.

For communication, the PIC12C672 incorporates the capabilities of other PICmicro families. For a stand-alone security system, LEDs are often sufficient in communicating status.

Each pin configured as an output can provide 25 mA of drive current. In a more complex system (e.g., communicating RS-232 to a PC's serial port), the fast instruction-execution rate of the MOVWF and TRIS commands enables the PIC12C672 to bit bang the RS-232 protocol.

The information can then be transmitted to a PC running the Windows Terminal program. Four pins are required-one for TOCKI, two outputs, and one input.

For transmit mode, TIMER0 generates the timing to send each bit of the serial stream. The value of the TIMER0 prescaler is determined by the input clock frequency and the data rate. The transmit pin (TX) can be any of the I/O pins set up as an output.

For receive mode, the receive pin (RX) must be connected to TOCKI to detect the asynchronous start bit of any transmission. The Option register is set up so TIMER0 is in counter mode and set to increment on the falling edge.

The chip's computational power supports parity generation. On reception of a packet, parity can be computed on the received byte and compared to the ninth bit received. Depending on system requirements, other protocols can be implemented (e.g., I^{2}C, 3-wire, or designer proprietary interface).

From the input standpoint, the PIC12C672 is equipped with interrupt on pin change. This capability lets push buttons, for example, be designed into a system, enabling direct control of the microcontroller. By pressing a button, the CPU is interrupted and goes off into a subroutine [e.g., checking the CO level) at that precise moment.

Its 8-pin architecture saves space, and its integration reduces component count. The on-chip 4-MHz system clock oscillator eliminates the need for an external oscillator. Its ability to store sensor calibration information in table look-up form in program memory space obviates the need for off-chip memory storage.

In some systems, pull-up resistors are used when connecting to open-collector transistors and similar circuits. The PIC12C672 under software control can select internal pull-up resistors at the I/O pins. In some manufacturing environments, the ability to uniquely program the contents of program memory for a number of systems inline is required.

Many sensor applications require a calibration step to measure and store offsets, slopes, and configuration infor-

Listing 1 --Programming *code is used to configure four channels of analog input (four sensors) for the conversion process.*

```
BSF     STATUS, RPO          ; select page 1
CLRF    ADCON1               ; configure A/D inputs
BSF     PIE1,   ADIE         ; enable A/D interrupts
BCF     STATUS, RPO          ; select page 0
MOVLW   0XC1                 ; select RC clock, A/D on, Channel 0
MOVWF   ADCONO
BCF     PIR1,   ADIF         ; clear A/D interrupt flag bit
BSF     INTCON, PEIE         ; enable peripheral interrupts
BSF     INTCON, GIE          ; enable all interrupts

; Add sampling delay routine to ensure that required sampling
; time for selected input channel has elapsed. Conversion may be
; started after this delay.
;
BSF     ADCONO, GO           ; start A/D conversion
```

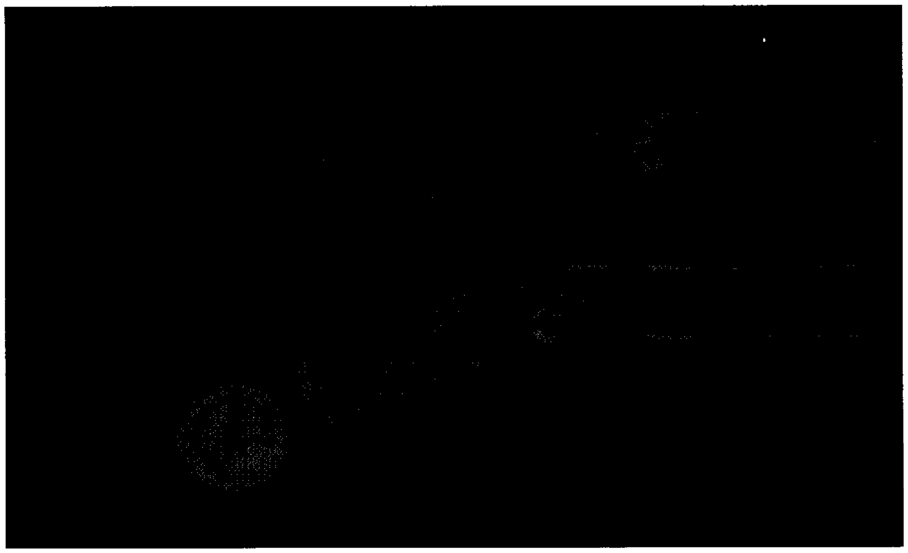


Photo I--The *PIC12C67x* targets *applications* in which the environment *(i.e., pressure, temperature, motion, acceleration, gas concentration, and sound) is being monitored and/or measured in a variety of products. The B-pin microcontrollers enable easy integration of first-time intelligent features into electromechanical designs and compete directly with 4-bit microcontroller products while providing enhanced performance.*

mation. Potentiometers or discrete serial EEPROM devices can set up and store this calibration information.

With the PIC12C672, the in-circuit serial programmer provides for a zero-component-count method. Using five pins, each sensor module can be programmed with a unique set of calibration parameters to bring any sensor tolerances to within the required accuracy.

And last but not least, the combination of an ADC with a PICmicro engine in an 8-pin package provides the most effective space and component savings. There aren't many discrete ADCs in an 8-pin package-let alone an ADC with an 8-bit RISC MCU.

IT'S A SMALL WORLD?

The PIC12C672 provides in a small form factor all the capabilities needed to measure and process sensor information as well as the control and communication driven by that information.

Everyday products such as battery chargers, rice cookers, toasters, thermometers, rheostats, thermostats, security systems, sensors systems—anything that measures the surrounding environment-now can have all the capabilities found in larger, complex systems. ❑

Damon Chu is strategic marketing manager for the Standard Microcontroller & ASSP Division at Microchip Technology. He has spent more than 15 years in chip design, DSP, and microcontroller systems with companies such as Microchip, Analog Devices, VLSI Technology, and RCA Advanced Technology Labs. You may reach him at *damon.chu@microchip.com.*

REFERENCES

J. Day, "AN656: In-Circuit Serial Programming of Calibration Parameters Using a PICmicro Microcontroller," ***Embedded Control Handbook,*** 1, Microchip Technology, Chandler, AZ, 1997.

S. Fink, "AN555: Software Implementation of Asynchronous Serial I/O," ***Embedded Control Handbook,*** 1, Microchip Technology, Chander, AZ, 1997.

SOURCE

PIC12C672
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-7200
Fax: (602) 786-7277
www.microchip.com

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

RELAY INTERFACE

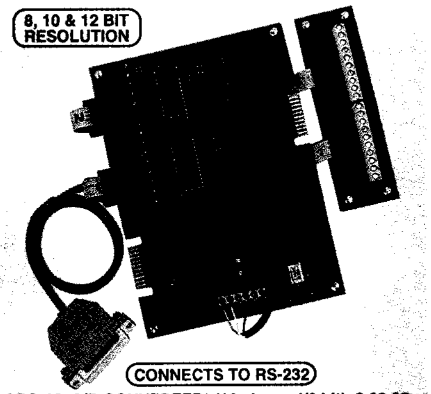
PROVIDES SOFTWARE CONTROL OF RELAYS

(CONNECTS TO RS-232)

AR-16 RELAY INTERFACE (16 channel) . . . . $ 69.66
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.
AR-2 RELAY INTERFACE (2 relays, 10 amp)....$ 44.63
RD-8 REED RELAY CARD (9 relays, 10 VA) . . . . . . $ 49.95
RH-8 RELAY CARD (10 amp SPDT, 277 VAC)...S 69.9%

ANALOG **TO DIGITAL**

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER* (16 channel/8 bit)..$ 99.95
ADC-8G A/D CONVERTER* **(8 channel/10 bit).$124.90**
Input voltage. amperage. pressure, energy usage, light, joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately). Includes Data Acquisition software for Windows 95 or 3.1
ADC-8E TEMPERATURE INTERFACE' (8 ch)..$ 139.95
Includes term. block & 8 temp. sensors (-40° lo 146' F).
STA-9 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) . . .. $ 39.95

***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 129 temperature inputs using the PS-4, E**X-16, ST-32 8 AD-16 expansion cards.

- **FULL** TECHNICAL **SUPPORT...provided** over **the telephone** by our staff. Technical reference **& disk including** test software & pro**gramming**examples in **QuickBasic,** GW Basic, Visua **Baic. 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 19,200 baud).**

FREE INFORMATION PACKET...use our 800 number.
Fax or E-mail to order, or visit our Internet on-line Catalog.
URL: http://www.eeci.com
Technical Support (614) 464-4470

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

Internet E-mail: eeci1 @ibm.net
International 8 Domestic FAX: (614) 464-9656
Use for information, technical support & orders.

ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 604
Columbus. **Ohio 43215-5491**

#106

FEATURE ARTICLE

DTMF Message Decoders

Telephone Aids for the Hearing Impaired

Steven Kraft

Talking on the telephone is an ordinary part of life—unless you can't hear. Steven shows how to build a DTMF message decoder to improve voice communication over the telephone for the hearing impaired.

Given today's fast-paced advancement in communications technology with its high-speed data fax and E-mail, we often take for granted the more personal experience afforded by simple voice communication on the telephone.

It's common for many of us to assume the party at the other end is equally equipped with the latest hardware, software, and technical knowledge. Unfortunately, not all are active participants in these advancements.

Many hearing-impaired individuals also are unable or unwilling to embrace these modern means of communication. Perhaps their hesitation is due to the cost of the hardware, or maybe they're unable to master using a computer keyboard.

Grandma's hearing may be getting worse as each year passes, but she doesn't need to miss out on your calls just because she cannot make out many of your spoken words over the phone. In many cases, all she needs is a simple communication aid to get across those words or phrases she can't seem to hear right.

A DTMF message decoder is just the ticket to improve the accuracy of voice communication. It restores the enjoyment of what has become a difficult communication experience.

This DTMF message-decoder project is based on the Motorola 68HC705K1 microcontroller and a 2-line x 40-character LCD readout. Although conceived independently for this application, it is quite similar to the HCS Message Man project that appeared in Jeff Bachiochi's "Talking on the Phone Without a Word" (*INK* 40).

It differs in two important respects. It has a more intuitive alphanumeric keypad decoding scheme. And, software hooks enable you to use an optional IBM PC parallel port interface to speed up alpha key entry.

A USEFUL INEXPENSIVE SOLUTION

The most common method of telephone communication with hearing-impaired persons essentially consists of a TTY or teletype device connection at each end of the phone line. It's possible to combine both voice and TTY communication on the same call, but the necessary procedure is somewhat inconvenient.

If your phone isn't equipped with this hardware, you must use a special relay service where a relay agent translates your words onto the TTY, compromising your conversation's privacy. You also have to consider the cost of this special hardware for both parties.

Modem communication is the only other option. Unfortunately, commonly available hardware precludes its use when voice communication is predominant during the call. Otherwise, you need either two dedicated lines for separate voice and data or the latest DSVD (digital simultaneous voice and data) modem.

These new modems allow you to multiplex both voice and data on a single phone line. However, you again have the cost of computer and modem hardware for both parties. And when you don't have your modem with you, the only way to communicate via telephone is by voice.

By instead relying on the universal DTMF encoding standard, it's possible to transmit words or short messages (albeit much more slowly) using any telephone anywhere with a standard touch-tone keypad.

You don't need any special hardware on the transmitting end and only the easily operated inexpensive DTMF message decoder on the receiving end. This arrangement conveniently allows the combination of both voice and DTMF-encoded messages on the same phone line.

DTMF Key Name	Primary Keypress (1st Alpha)	Repeat Key once (2nd Alpha)	Repeat Key twice (3rd Alpha)	Num Lock Mode On (Numeric)
1	,	Q	Z	1
2	A	B	C	2
3	D	E	F	3
4	G	H	I	4
5	J	K	L	5
6	M	N	0	6
7	P	R	S	7
8	T	U	V	8
9	W	X	Y	9
0	space	backspace	backspace	0
•		clear display	clear display	
#	num lock			

Table l--This *fable shows the correlation between* key presses *and* the resulting characters displayed oh *the* LCD. *A* "repeat key" *is pressed* again within 0.7 s of ifs *first* pressing.

CIRCUIT DESCRIPTION

The complete circuit schematic diagram for the DTMF message decoder is shown in Figure 1. The op-amp circuit consisting of U1 and associated resistors and capacitors is a unity-gain audio-signal amplifier that provides a high-impedance interface with the phone line.

Diodes D1 and D2 serve as ringer voltage clamp protection for U1. The 2.5-V reference U2 is used to bias U1 halfway between the O-5-V power-supply rails to achieve maximum output signal swing. A low-pass filter formed by R5 and C3 attenuates high-frequency noise.

U3 is an integrated DTMF decoder IC that converts standard touch tones into their 4-bit binary-encoded equivalent representation. When a valid DTMF signal is detected, this code is latched into the output port of U3 and the Data Valid signal at pin 14 becomes active.

This signal is inverted by Q1 which drives the external interrupt pin of microcontroller U4. When U4 detects an interrupt condition, it immediately reads the 4-bit DTMF data from U3 and then converts it to the appropriate code for controlling the LCD module. R13 controls LCD contrast, and LED1 serves as a flashing power-on indicator.

Power for this circuit is supplied by a 9-V battery regulated down to 5 V by voltage regulator U5.

OPERATION

Any standard touch-tone phone may be used to send brief messages (up to two lines of 40 characters) to the message-decoder unit connected at the other end of the line. Allowed characters include all capital letters of the alphabet, the numbers O-9, commas, periods, and spaces.

To send the first letter of your message, just press the desired key once (e.g., A is sent with a single press of the 2 key).

If your letter is the second one assigned to that key, you must press the key twice within a 0.5-s time period. B is sent with two rapid presses of the 2 key.

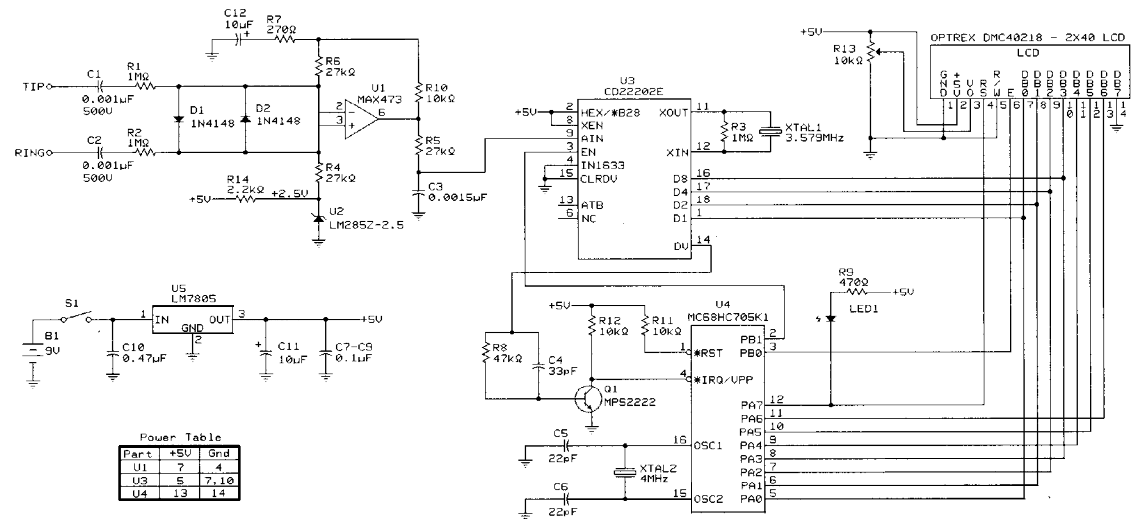
If you wait too long (i.e., more than 0.7 s) between presses, the decoder interprets the second press as the next letter of your message. You'd have sent the two letters "AA" instead of the single letter "B".

On the other hand, if you want to send two consecutive letters, both assigned to the same key, you should wait at least one full second after pressing this key for the first letter before pressing it again.

In addition to sending letters A-Z, you can also send a comma by pressing the "1" key once or a period by pressing the "*" key once. To put a space between words in *your* message, press the "0" key once.

To correct the previous character, backspace one character position by pressing the "0" key twice in rapid succession.

If you need to start your message from the beginning or to erase a previous message from the display, press the "*" key twice rapidly. This action clears the display and enables you to

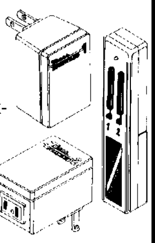


Part	+5V	Gnd
U1	7	4
U3	5	7,10
U4	13	14

Figure l--Received *DTMF signals are processed and* the *resulting characters sent fo* the *LCD module configured for B-bit data-transfer mode. A blinking LED "system active" indicator time shares processor output PA7* with the *data/command select pin of* the *LCD.*

ALL ELECTRONICS CORPORATION

2 Channel RF Remote Control



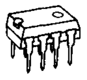
"Total Command"™ by Leviton Operate TV, lamps and other appliances from up to 100 feet away. Turn-on house lights from your car. Simple to hook-up and use. Includes two receivers on separate frequencies and a two button hand-held transmitter that controls them. Requires 9 volt battery (not included). Clam-shell display packaging. **$17.95 each**
CAT # TCR-1

Ultrabright LED

PAINFULLY BRIGHT RED LED
2500 to 4000 mcd @ 20 ma. These T 1 314 (5 mm diameter) red LEDs are significantly brighter than conventional LEDs. At close range, they are painful to look at. They are great for attention getting displays that can be seen from a distance. Water clear in off-state.
CAT # LED-42 **2 for $1.20**

10 for $5.00 • 100 for $45.00
1000 for $400.00

Low Power Audio Amp



Motorola MC341 19P
Low power audio amplifier suitable for speakerphones or talking picture frames. The 8 pin DIP package requires only a few additional parts, operates on 2 - 16 volts and drives speakers of 8 ohms or greater. Output power exceeds 250 mW with 32 ohm speaker. Power-down option saves power in battery driven applications. Hook-up sheet. Large quantity available.
CAT # MC341 19P
60¢ each **50 Pieces - $25.00**

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 COO. Prices subject to change without notice.

CALL, WRITE FAX or E-MAIL for our **FREE** 96 Page CATALOG
Outside the U.S.A. send $3.00 postage.

MAIL ORDERS TO:
ALL ELECTRONICS CORPORATION
P.O. Box 567
Van Nuys, CA 91408
FAX (818)781-2653
e-mail allcorp@allcorp.com
internet - http://www.allcorp.com/

#107

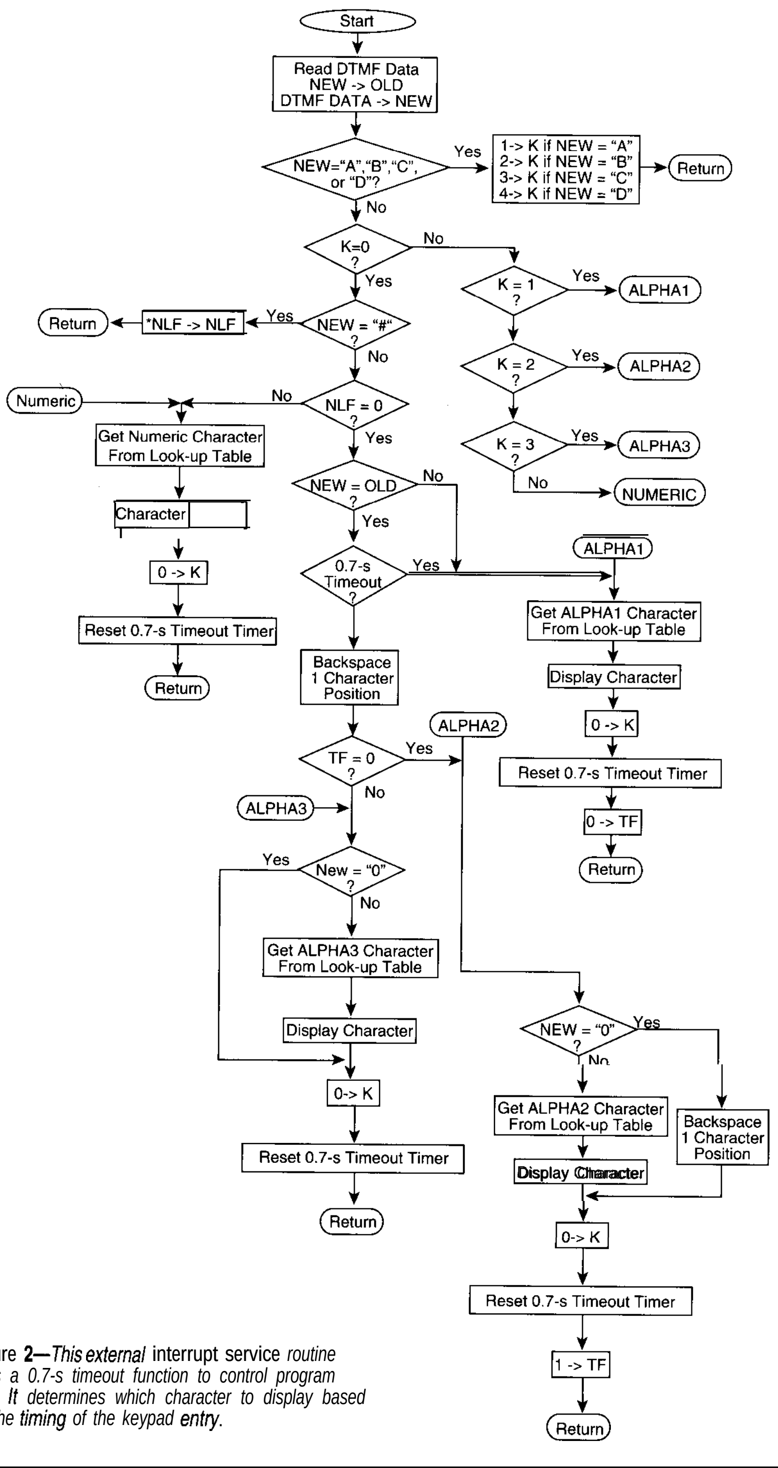


Figure 2—*This external interrupt service routine uses a 0.7-s timeout function to control program flow. It determines which character to display based on the timing of the keypad entry.*

start from the first character position of the first line.

Messages longer than 40 character positions automatically wrap to the second line of the display. If your message exceeds 80 characters total, you begin overwriting the first line of your message. So, be sure to clear the previous message before starting a new one.

To send a number, first press the "#" key once. This action puts you in the numeric mode, and all keys are interpreted by the decoder just the way the numbers appear on the key pad.

In numeric mode, you also can send a decimal point (period) character. To return to the letter mode, just press the "#" key again.

Listing 1—This simple *QuickBASIC* program is used *to* enter messages via PC keyboard *using* single-line character *input. It allows* up to *80* characters to be entered, edited, *and* automatically *transmitted as* encoded *DTMF signals.*

```
1   REM ****** DTMF MESSAGE ENCODER PROGRAM ************
2   CLS : DIM DATAOUT%(81): DIM LOOKUP%(40)
3   LOOKUP$ = " ,.0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
4   DATA 124,124, 12, 60,127, 15, 79,143, 31, 95,159, 47,111,175
5   DATA      76, 77, 78,140,141,142, 28, 29, 30, 92, 93, 94,156
6   DATA     157,158, 44, 13, 45, 46,108,109,110,172,173,174, 14
7   FOR I = 0 TO 39: READ LOOKUP%(I): NEXT
8   port = &H378: REM Address of parallel port used.
9   LOCATE 10, 25: PRINT "DTMF MESSAGE ENCODER PROGRAM"
10  LOCATE 15, 25: PRINT "COPYRIGHT (c) 1995 ELECTROKRAFT"
12  REM
14  REM Determine 60-ms DELAYCOUNT value for host computer.
16  TIMER ON
18  ON TIMER(6) GOSUB 900
20  FOR L& = 1 TO 10000000: NEXT L&
22  LOCATE 20, 30: PRINT "[ PRESS ANY KEY ]"
24  DO WHILE INKEY$ = "": LOOP
26  REM
28  REM Input up to 80-character message string and encode as DATAOUT.
30  CLS : DATAOUT%(0) = 61: REM Clear LCD before message.
32  FOR I = 1 TO 40: PRINT CHR$(196);: NEXT
34  FOR I = 1 TO 40: PRINT CHR$(223);: NEXT
36  LINE INPUT M$
38  M$ = UCASE$(LEFT$(M$, 80)): IF M$ = "Q" THEN END
40  FOR I = 1 TO LEN(M$)
42   DATAOUT%(I) = LOOKUP%(INSTR(LOOKUP$, MID$(M$, I, 1)))
44  NEXT I
46  REM
48  REM Output DTMF data codes to parallel port.
50  FOR I = 0 TO LEN(M$)
51   OUT port, DATAOUT%(I)
52   OUT port + 2, &H1
53   FOR L& = 1 TO DELAYCOUNT&: NEXT L&: REM 60-ms TONE ON.
54   OUT port + 2, &H0
55   FOR L& = 1 TO DELAYCOUNT&: NEXT L&: REM 60-ms TONE OFF.
56   OUT port, DATAOUT%(I) \ 16
57   OUT port + 2, &H1
58   FOR L& = 1 TO DELAYCOUNT&: NEXT L&: REM 60-ms TONE ON.
59   OUT port + 2, &H0
60   FOR L& = 1 TO DELAYCOUNT&: NEXT L&: REM 60-ms TONE OFF.
62   IF I = 0 THEN 70
64   R$ = MID$(M$, I, 1)
66   IF INSTR(LOOKUP$, R$) = 0 THEN PRINT " "; ELSE PRINT R$;
70  NEXT I
72  PRINT : BEEP: REM DTMF Message Transmission Completed.
74  GOTO 32: REM Ready for next message.
76   REM
900 REM ON 6-s timeout EVENT Calculate 60-ms DELAYCOUNT value.
901  TIMER OFF
902  DELAYCOUNT& = L& / 100 + 1
903  L& = 9999999
999  RETURN
```

PROGRAM DESCRIPTION

Performing these functions requires the internal 68HC705K1 PROM to be programmed with the DTMF message-decoder firmware.

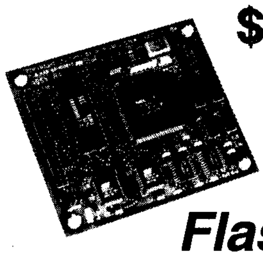
The program begins by initializing I/O ports A and B and defining all bits as outputs. All variables are cleared after enabling both real-time interrupt and external interrupt functions.

An internal timer is programmed to generate real-time interrupts every 65 ms. The program uses these primarily for the 715-ms repeat key timeout function.

After a 1-s power-up delay, the LCD module is initialized by sending a sequence of four 8-bit control codes to the module via port A. Following the output of each 8-bit code, a display subroutine waits 600 µs (to accommodate the LCD processing delay] and then strobes the LCD enable line via port bit PB0.

EMBEDDED DOS
CONTROLLERS AT 8051 PRICES

Use Your PC Development Took
No More Crash & Burn EPROM Technology



$195
Qty 1 Price

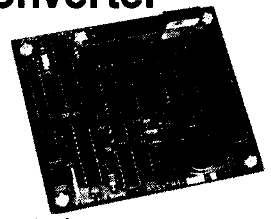
Flashlite
DOS Single Board Computer
with 572 *k* FLASH Memory *disk* drive

- ✓ 10 Mhz/8 Mhz CPU ✓ 2 Timers
- ✓ 512 k bytes RAM ✓ 4 Interrupt Lines
- ✓ 512 k/256 k FLASH ✓ 8 Analog Inputs
- ✓ 2 Serial Ports ✓ X-Modem File Transfer
- ✓ 24 Parallel I/O Lines

INCLUDES DOS & Utilities

USE YOUR TURBO C COMPILER OR QUICKBASIC COMPILER
SAVE TIME, MONEY AND HEADACHES

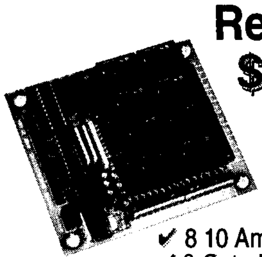
A/D Converter
$95
Qty 1 Price



- ✓ 8 Channels, 12 Bits
- ✓ 6 µs. Conversion Time
- ✓ Clock/Calendar Option
- ✓ Includes Drivers & Apps.

GET YOUR EMBEDDED CONTROLLER PROJECT RUNNING *FAST!* WITH THESE ACCESSORIES

Relay I/O
$139
Qty 1 Price



- ✓ 8 10 Amp Relays
- ✓ 8 Opto-Isolated Inputs

JK microsystems
Cost Effective Controllers for *Industry*
TO ORDER (510) 236-1151
FAX **(510) 236-2999—email: jkmicro@dsp.com**
Visit our WEB **site—www.dsp.com/jkmicro**
1275 Yuba Ave., San Pablo, CA 94806

#108

This same subroutine gets called each time a character is output to the LCD module. Port bit PA7 is set to logic zero for control data or logic one for character data.

After displaying a start-up message for 3 s, the LCD is cleared and the program enters a perpetual wait loop where it flashes the LED power-on indicator. It is ready to receive and decode DTMF messages sent over the phone line.

The IRQ input to U4 is pulled low each time U3 detects a valid DTMF signal. This external interrupt causes the program to temporarily suspend execution of the wait loop while servicing the interrupt.

Figure 2 shows the program flow for the external interrupt service routine. It starts by calling a subroutine to read the DTMF data from the output port of U3.

This subroutine first redefines port-A bits O-3 as inputs to the microcontroller and then enables the U3 output port. DTMF data is now read into one of the internal RAM locations of U4. Port-A bits O-3 are then set to function again as outputs.

Variable K is used for processing the auxiliary DTMF codes (separate keys labeled A, B, C, and D which do not appear on most standard 12-key phones). These auxiliary codes can be used by the program in conjunction with an optional IBM PC interface.

Bit flag NLF controls a num 1 ock mode via the "#" key. Timeout flag T F controls the program flow when keys are repeatedly pressed within the 0.7-s timeout period. Table 1 shows the correlation between key presses and the resulting displayed characters.

SPEED IT UP...A LITTLE

Some patience and practice is required to master effective communication from a touch-tone keypad. The slow rate of character entry limits real-time communication to at most a few words or short phrases per exchange.

If you happen to have a portable IBM PC handy, you can transmit messages to the same DTMF message decoder much more conveniently by using an optional IBM PC interface. Figure 3 shows the circuit used between parallel printer port and telephone line.

This telephone-line connection circuitry is intended for experimental use. Any commercial application would require substitution of an FCC-registered DAA module.

A complete QuickBASIC listing for this PC printer port interface is shown in Listing 1.

The PC keyboard is used to enter single-line messages up to 80 characters long-displayed on the PC monitor. Pressing Enter sends these characters (encoded as a sequence of two DTMF codes per character) to the DTMF message decoder at the other end of the. telephone line.

A crude but effective software delay loop generates an ~60-ms delay to satisfy the 50-ms minimum tone-duration requirement of the CD22202E receiver/decoder chip.

This DTMF message-encoder interface is much more convenient to use and achieves a relatively blazing transmission rate of four characters per second. Still, it's faster than manual entry on a touch-tone keypad.

BOXING IT UP

Almost any packaging scheme can be used for the DTMF message decoder. Since the LCD module dimensions are long and narrow, unlike readily available project boxes, I opted for a simple custom enclosure.

The 2" x 6" decoder circuit board and LCD module were mounted in a 2.5" x 2.5" x 10" metal enclosure formed by bending two precut pieces of 10" x 4" aluminum. A pair of triangular wooden endpieces complete the enclosure.

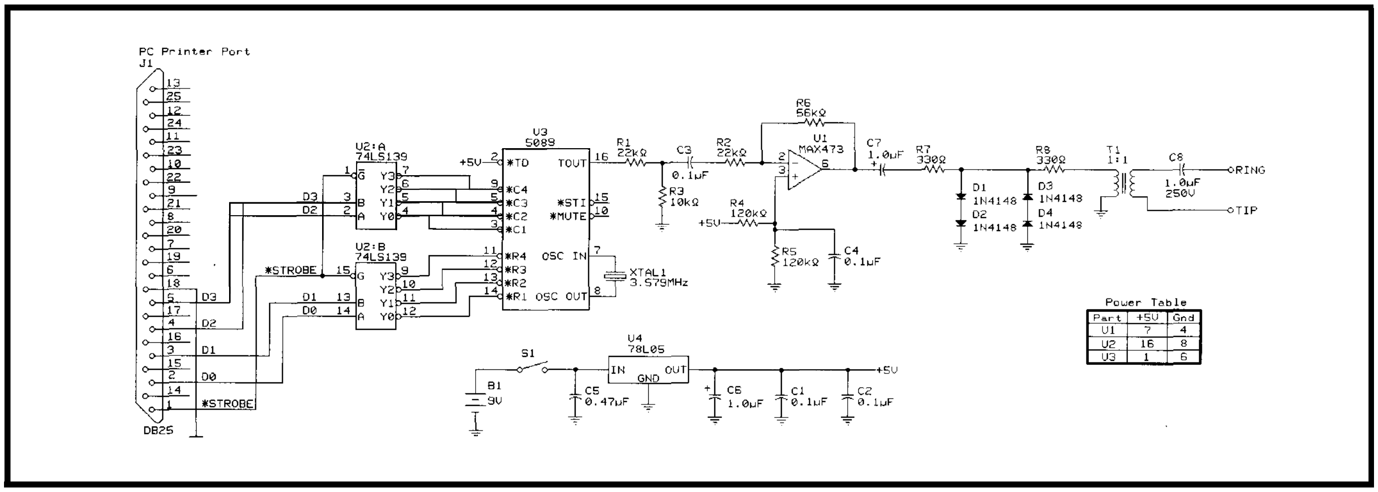
Install a fresh 9-V battery and turn on the power. You should see a start-up message displayed for -3 s, after which the display clears and the LED begins flashing at a l-Hz rate.

R13 controls display contrast for optimum viewing in ambient room light. To test the unit, connect it to any phone jack having an accessible touch-tone phone connected to the same line.

ON THE PHONE AGAIN

In all honesty, this project was as much an excuse to learn how to use an inexpensive microcontroller like the 68HC705K1 as it was an opportunity to create something personally useful.

Grandmother's hearing loss was serious enough to warrant some simple communication aid beyond an amplified telephone. I wanted a solution that partially preserved the voice com-



Part	+5V	Gnd
U1	7	4
U2	16	8
U3	1	6

Figure 3—*Simple hardware connected to a PC printer port uses the 5089 DTMF tone-generator chip to speed up message transmission. A QuickBASIC* program controls fhe *printer port.*

munication experience and yet was inexpensive, convenient, and easy to use from any location.

The DTMF messaging approach may seem unbearably slow-until you're communicating with a person who only understands 50% of your spoken words. Then, this turns out to be a wonderfully useful aid. ❑

Steven Kraft is an electrical engineer and holds an MSEE from the University of Colorado. He has worked on satellite control electronics and is currently a part-time consultant involved with electronic security-system sensor design. You may reach him at steven.kraft@circuitcellar.com.

SOFTWARE

The complete listing of DTMF6 . **ASM** is available on the Circuit Cellar Web site. **DTM** F6 . S **19** contains the object code.

SOURCE

DMC40218 (**Optrex LCD** module),
MAX473 op-amp
Digi-Key Corp.
701 Brooks Ave. S
Thief Falls, MN 56701-0677
(218) 681-6674
Fax: (218) 681-3380

CD22202E DTMF receiver/decoder
JDR Microdevices
1850 S. 10th St.
San Jose, CA 95112
(408) 494-1400
Fax: (408) 494-1420

TCM.5089 DTMF tone generator
Jameco
1355 Shoreway Rd.
Belmont, CA 94002-4100
(415) 592-8097
Fax: (415) 592-2503
www.jameco.com

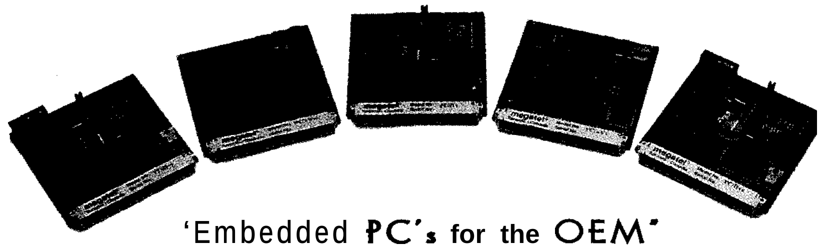
Preprogrammed **68HC705K1P** parts
Electrokraft
P.O. Box 598
Louisville, CO 80027
(303) 877-4532
Fax: (303) 665-857

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

Small Space Solutions

Megatel Offers a Wide Range of Embedded PC Architecture



'Embedded PC's for the OEM'

On-Board Features such as:

- 486DX,™ 386,™ 8088 or V40 Processors
- Local Bus Super VGA supporting LCD Panels
- Up to 16MB DRAM
- Ethernet Local Area Network
- On-board programmable Flash™
- Low Power Consumption
- Operation +5 volts only (typical)
- 2 IBM compatible RS-232C Serial Ports
- 8-bit general purpose Parallel I/O Port
- Only 4" x 4" form factor
- ISA or PC1104 Bus compatible options

Embed This

For more information call
Toll Free: 1•888•762•5572 or 416•245•2953
fax: 416•245•6505 email: sales@megatel.ca
Website: www.megatel.ca

megatel® 1•888•SMALL•PC

#110

Risc

32-bit controller

- 28 Programmable analog I/O, SE or differential, 4-20 mA
- Keypad & display operator interface
- 24-Plus digital I/O
- 4 RS-232/485 ports, optically isolated
- 4 Counter/quadrature encoder, pulse timing
- Flash EPROM program storage
- C Programmable
- Starting from $595*

Call for NEW catalog & information on more embedded controllers!

REMOTE™ PROCESSING
The embedded control company

Incredibly powerful

At 50 nS instruction time, our RPC-400 delivers the power that an 8 or 16-bit controller can't.

Inherently complete

We've included everything to handle even the most demanding industrial I/O requirements.

Instinctively simple

Easy to program, run, and get going!

Ph: 303-690-1588, Fax: 303-690-1875
Web Site www.remotep.com [illegible]

#111

FEATURE ARTICLE

Craig Pataky

Interprocess Communication

Moving DOS Programs into Windows

Sometimes you want your program running in DOS and you want a GUI front end. Impossible? Not according to Craig. Although DOS and Windows are a universe apart, all you need is a wormhole to bring them together. Here's how to create one.

In today's rapidly changing software landscape, it's getting increasingly difficult to apply old knowledge to new problems.

Because PCs changed little between 1985 and 1992, a lot of people developed a mind-set that simply doesn't apply to modern operating systems. These days, it's not reasonable to assume the computer is running only one program, nor is it acceptable to expect the keyboard to be the only input device.

However, since I'm reluctant to change any habit that works for me, I spend about as much time writing code for DOS virtual machines (VMs) as for Windows. After all, sometimes the application just doesn't need a nice GUI to get the job done.

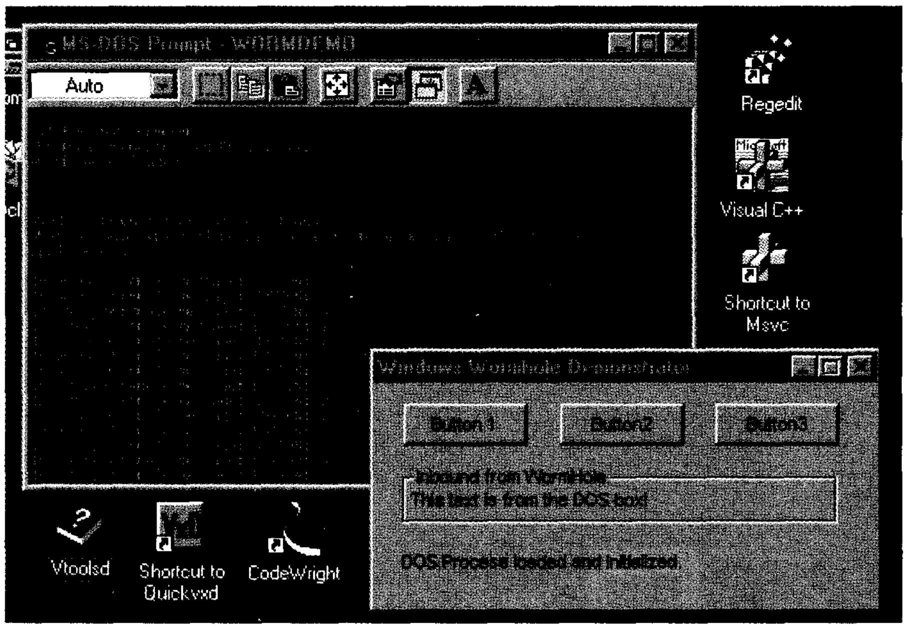
There comes a time, however, when implementing a GUI becomes important. To this end, I always wished I could just keep my old DOS code and put a Windows face on it. What I really needed was some way for an application in a hidden DOS box to communicate with a Windows application that handled the interface issues.

I read about DDE, but it seemed overly complex and hasn't worked across the VM boundary since the '286.

I also investigated creating a TSR to reserve some shared memory that DOS and the system VMs could use, but that seemed vulnerable and clumsy. Using the clipboard looked too slow.

I wanted something clean and encapsulated-like a network interface.

With DOS and Windows applications being a universe apart, I needed something major, something drastic.. . I needed a wormhole.


Inbound from WormHole
This text is from the DOS box

DOS Process loaded and initialized

Photo 1 -*As buttons are pressed, messages are senf to the DOS VM in the background and displayed. When keystrokes are entered into the DOS VM and it has fhe focus, keystroke messages are sent to the Windows GUI.*

VxDs SAVE THE DAY

Although DOS VMs and Windowed applications can't talk to each other directly, you can create helper programs that facilitate interprocess communications on the application's behalf. The easiest way to accomplish this is with a Virtual Device Driver (VxD).

In the article "Getting Beyond The Box With Windows 95" *(INK* 74), I depicted the relationship between applications (including DOS VMs), VxDs, and the hardware as a three-layer pyramid. The bottom-most layer represents physical hardware (e.g., parallel ports, serial ports, etc.)

The middle layer represents VxDs with the specific knowledge needed to control the associated hardware devices. The VxD also sports some form of API to communicate with the next layer up.

At the top is the applications layer, which is the realm of DOS VMs and GUI applications. The important idea here is that both DOS VMs and GUI applications can-and routinely do—interact with VxDs. More specifically, DOS VMs and GUI applications tend to interact with the ***same*** VxD when using a common hardware resource.

A useful realization is that a VxD doesn't necessarily have to be associated with any specific hardware. You can fashion a VxD so that it's common to DOS VMs and GUI applications but doesn't perform any hardware interaction whatsoever. With this in mind, it was a snap to create WORMHOLE.VXD and cross the VM boundary.

HOW IT WORKS

The golden nugget to WORMHOLE.VXD's operation is that it supports both the standard DeviceIOControl API common among Win32 applications as well as a nonstandard API in the form of a hook on int 14h so DOS applications can access the same VxD services.

Always remember that VxDs are global across all VMs, so every DOS box or Windows application that uses a wormhole's API is communicating to the same wormhole. This idea is what makes the bridge between DOS boxes possible.

The version of WORMHOLE.VXD in this article acts as a central post office where applications can send and retrieve messages. It can facilitate communications between up to 16 different applications across all VMs.

On startup, an application registers itself by name with the wormhole by using the applicable API (listed later) and receives a handle to its own message queue. An application can later use this handle like a P.O. box number when retrieving messages coming in through the wormhole from other applications. Similarly, an application can find another application's queue handle and use it to send messages to the other application.

Message sizes are limited to 250 bytes or less. Queue sizes are currently fixed at 1 KB, which is more than sufficient for most purposes.

SHOW ME!

If you're like me, this prattle means nothing without real source code. So, I created a DOS executable (WHDOSDEM.EXE) and a Win32 application (WHWINDEM.EXE) that use WORMHOLE.VXD to send messages to each other. The source code for these files will be your best reference, as these little applications exercise almost all the functionality of the wormhole.

Before running WHDOSDEM, load WORMHOLE.VXD and reboot your PC after adding the following entry to your SYSTEM.INI file:

```
device=Cpathl\wormhole.vxd
```

When WHDOSDEM has the input focus and a key is pressed, the character is forwarded to WHWINDEM in the form of a message sent through the wormhole (see Photo 1). When the message is received by the GUI, the character is displayed in the Incoming box.

API

At a higher level, the API for the wormhole is pretty much the same for both DOS and Windows applications. And, it supports commands that enable applications to register themselves and find other registered applications. The information gleaned from these commands can be used to post and retrieve messages to and from other applications.

The API is encapsulated as static functions in the file WORMHOLE.H. To use the Wormhole API, simply #define USES_WORMHOLE_WIN or USES_WORMHOLE_DOS (depending on your target environment) and #include "WORMHOLE.H".

TOOLS

To build the wormhole (WORMHOLE.VXD), I used Microsoft Visual C 2.0 (MSVC) and VtoolsD. The combination of VtoolsD and MSVC enables you to program VxDs in C, which as I soon learned, was preferable to using assembly language and the DDK. The "C Prototypes" sidebar has more details.

If you're serious about exploring the possibilities of the VxD, I can't more highly recommend a better tool combination. VtoolsD also comes with loads of documentation and several example programs that are directly applicable to most endeavors.

The Windows demonstration program (WHWINDEM.EXE) was created using MSVC 2.0. Versions of the MSVC development environment beyond 2.0 are radically different than their predecessors, so I'm reluctant to upgrade to the current revision (V.5).

MSVC 2.0 generates 32-bit code that is meant to be executed in the Windows 95 or NT environments. Though you can create what appears to be a genuine 16-bit DOS program using MSVC 2.0, it's not. Just try running it on your '386!

The WHDOSDEM application was built with MSVC 1.6. I've also built it with Borland C 3.0. These compilers generate 16-bit code, which means their executables can run in native DOS.

REAL-WORLD IMPLEMENTATION

My most recent application for the WORMHOLE came about a month ago. I needed to give a Windows application access to a slow (9600 bps) proprietary network.

It just so happened that I had already created the network drivers for a hand-held DOS portable eight months prior.

Not wanting to create an entirely new network driver for Windows, I simply repackaged my old DOS network drivers to take advantage of the wormhole. Within two days, the Windows application was up and running as a network participant.

POSSIBLE ENHANCEMENTS

Most of the limitations of WORMHOLE stem from my own arbitrary decisions. There's no real reason, for instance, that queue sizes have to be limited to 1 KB or that there be a maximum of 16 wormhole participants.

Also, rather than retrieving handles to message queues and using them to send/receive data, it may be more convenient to use process names exclusively. This technique does involve slightly more processor overhead, but speed isn't a genuine issue on a modern Pentium.

Because languages other than C derivatives have gained in popularity, it may be desirable to encapsulate the wormhole API within a DLL. That would allow Access, Visual Basic, and a host of MIS programmers to benefit from the wormhole.

HEADING ON OVER

When all is said and done, WORMHOLE turns out to be a convenient means of interprocess communication among Win32 applications whether you need to communicate with a DOS VM or not.

DDE, OLE, and the clipboard were originally meant to fill this need, but they're more complex than necessary.

I must mention that wherever I look in PC literature, I get the impression that the demise of DOS is imminent. Heck, even palmtops are running a version of Windows 95!

Of course, in a perfect world with no time constraints, writing software as full-blown multithreaded Win32 applications would almost always be preferable to cobbling together a bunch of DOS boxes.

The reality is, however, that time to market is more critical than elegance of implementation.

If DOS is what the bulk of your staff knows and a DOS/Win hybrid is good enough to go the distance, jump at it. I hope this wormhole makes it easier. ❑

C Prototypes

BOOL WORMHOLE_LoadVxD(void); is available only to Windows applications and ensures the wormhole is loaded and initialized. If the wormhole hasn't already been loaded by an entry in SYSTEM.INI, this function attempts to dynamically load the VxD. In dynamically loading, you must make sure the VxD is in either your application's start-up directory or in the \WINDOWS\SYSTEM directory. Either way, no other API functions will work until this function is called.

BOOL WORMHOLE_IsLoaded(void); is available only to DOS applications and ensures the wormhole has been loaded and initialized. The wormhole must have been previously loaded by an entry in SYSTEM.INI or by a Win32 app calling the LoadVxD API.

BYTE WORMHOLE_RegisterProcess(char* pProcName); registersthe given process in the process table and returns a handle to that process. If no room is available in the process table, it returns -1.

BYTE WORMHOLE_FindProcess(char* pProcName); looks for the given process in the process table and returns the handle to it if found. It returns -1 if the process isn't found.

BYTE WORMHOLE_RemoveProcess(char* pProcName); removesthegiven process and returns a value greater than -1 if successful.

BOOL WORMHOLE_PostMessage(BYTE Src,BYTE Dest,BYTE Type,BYTE* pBuf,BYTE Len); attempts to post a message to the destination-process message queue. Src should be the handle to the caller's process queue so the receiver may reply. You can use the Type variable to describe the application-dependent data being routed to the destination process. pBuf points to a buffer of data to be sent, and Len is the number of bytes to be sent. This function returns TRUE if the message posted OK or FALSE if the receiver's queue was full or invalid.

BOOL WORMHOLE_GetMessage(BYTE Dest,BYTE* pSrc,BYTE* pType, BYTE* pBuf, BYTE* pLen); attempts to retrieve a message addressed to the Dest process handle. If a message is available, this function returns TRUE with pSrc equal to the sender's process handle, pType equal to the application-dependent type of message, pBuf filled with the body of the message, and pLen equal to the number of bytes in the message body. If no message is available, it returns FALSE.

Craig Pataky is a systems engineer at Hi-Tech Transport Electronics. His eight years in software has included everything from simple embedded programming to operating-system design. At present, most of his time is spent designing and implementing network protocols for the trucking industry. You may reach him at CraigPtky@aol.com.

SOFTWARE

The complete source code (WORMHOLE.ZIP) for this article can be downloaded from the Circuit Cellar Web site.

SOURCES

Borland C
Borland Int'l.
100 Borland Way
Scotts Valley, CA 95066
(408) 431-1000
www.borland.com

MSVC 2.0, Windows OSs
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052
www.microsoft.com

VtoolsD
Vireo Software, Inc.
21 Half Moon Hill
Acton, MA 01720
(508) 264-9200
Fax: (508) 264-9205
sales@vireo.com

IRS

407 Very Useful
408 Moderately Useful
409 Not Useful

FEATURE ARTICLE

Bill Jackson & Reynaldo Archide

Compressed-Code TinyRISC

In the embedded world, it's a precarious balance between cost and performance. To maximize memory and reduce costs, Bill and Reynaldo suggest MIPS16, an extension of LSI Logic's TinyRISC family.

The growing demand for a variety of newer consumer and communications products is challenging the system designer to find innovative ways to design more efficiently and reduce cost. Performance at any cost is the one and only byword for system engineering focused on ultimate desktop computing power.

However, cost-sensitive embedded applications in products like cellular phones, Internet appliances, and routers are burgeoning. The engineering concentration in this particular scenario is on balancing performance and cost.

Higher integration, smaller device packaging, and advanced process technology are helping to reduce design cost. To date, though, the embedded microprocessor hasn't contributed a fair share.

Ideally, in this instance, the system designer should have a compact-code instruction set architecture [ISA) that can cut down on how much program memory is necessary in a system.

System memory has traditionally accounted for a major portion of the system design cost. Conventional PCs today require 8-16 MB of EDO DRAM, for instance, costing about $50-100. Network computers (NCs) use 8 MB of memory, which costs about $50.

An embedded processor based on a compressed-code ISA halves memory size. One example is the recently developed MIPS 16 applications-specific extension (ASE) embedded in LSI Logic's TinyRISC TR4100 family of 32-bit microprocessor cores.

MIPS16-enabled processors provide 32-bit RISC performance with an overall system cost more typical of a 16-bit processor. These microprocessor cores represent the first implementation of the MIPS 16 ISA developed by Silicon Graphics and optimized for price-sensitive applications.

The TR4100 family includes three initial members-TR4101, TR4102, and TR4120. Based on LSI Logic's 0.35pm process, the TR4101 has a clock frequency of 70 MHz and features about 60-Dhrystone MIPS performance running 16-bit code.

Running 32-bit code, the clock frequency increases to 81 MHz and 70-Dhrystone MIPS performance. The TR4101 measures 1.7 mm^2, operates at 3.3 V, and dissipates 1 mW/MHz. The TR4102 measures 1.5 mm^2, operates between 2.5 and 3.3 V, and dissipates 0.7 mW/MHz.

The TR4120, which includes a bus interface unit (BIU), is a high-end superscalar machine operating at 3.3 V and measuring 4.2 mm^2. Performance is expected to be 100-MHz clock frequency and 130-Dhrystone MIPS.

The TR4101 core provides embedded-systems designers with a cost-efficient building-block approach for

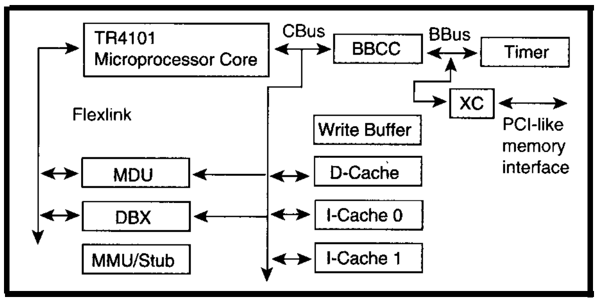


Figure 1 —*The system designer can use a variety of optional building blocks bolted on to the embedded TinyRISC microprocessor core. The blocks can be modified with customized logic.*

Instruction Group	MIPS-II	TinyRISC	A
Load/Store	13	12	–1
Moves	0	4	–4
ALU	14	12	-2
Logical	7	5	–2
Compare	4	6	–2
Shift	6	5	0
Jump	4		+1
Multiply/Divide	4	4	0
Special Instructions	0	2	+2
Branch	8	5	- 3
Coprocessor/Kernel	50+	0	
Total	106+	57	

Table 1 --MIPS 16 *reduces the number of* microprocessor instructions from the usual *106* found *in the MIPS-II* instruction set to *only* 57, reducing *the RISC* code size and *memory requirements by 40%.*

implementing highly integrated system-on-a-chip designs.

It can run either 16- or 32-bit instructions on the same processor. The Jump and Link Exchange (JALX) instruction enables the processor to switch between 32- and 16-bit code.

As shown in Figure 1, the TinyRISC processor core operates with a combination of LSI Logic blocks and modules, including a basic BIU and cache controller (BBCC), multiply/divide unit (MDU), extended bus interface unit controller (XC), timer, write buffer (WB), and debug module (DBX).

It also has a two-way set associative instruction or direct-mapped (I) cache and a data (D) cache. These optional building blocks can also be modified via customized logic.

The processor core reduces RISC code size by 40%, going from over 106 instructions in 32-bit MIPS to only 57 MIPS16 instructions (see Table 1). At the same time, it cuts system memory requirements up to 40%.

While the MIPS16 compressed code size is only 60% of the uncompressed MIPS code size, the TR4101's overall performance is equivalent to 85% of the full 32-bit uncompressed ISA.

MIPS16 OVERVIEW

Figure 2 overviews the instruction formats. MIPS16, which includes support for 64-bit operands, reduces the number of 32-bit MIPS instructions by about a third. A set of instruction variances in this format can be expressed in MIPS 32-bit code but not in the compressed code.

Consequently, a multiple-instruction sequence can achieve the same effect. The sequences are used sparingly, yet they contribute to overall code savings.

Instruction types between the 32-bit MIPS ISA and the MIPS 16 ASE do not change drastically, which exemplifies the approach taken to evolve the compressed code from the current MIPS architecture.

In the Immediate (I) type, one register field (rt) is omitted, leaving only the operation (OP) and source register (rs). The remaining portion of the instruction is devoted to the immediate field.

As well, the Jump (J) type format used for jumps and branches is narrowed down in the field specifying the target address.

The R-type is split into two related types-RR and RRR. In each case, fewer available bits in the 16-bit format are used to perform some functions previously done in the R-type.

In the MIPS ISA, 6 bits specify the operation. Two 5-bit fields specify two registers (rs and rt), which allow any pair of registers to take part in this instruction (see Figure 3a). The immediate field is 16 bits wide.

In MIPS16 ASE, the operand field is 5 bits. The rt field is omitted, and the rs is pared down to 3 bits. Therefore, one of only eight registers can be selected, whereas it was one of 32 in 32-bit MIPS.

Omitting the rt field means the previous contents of a register are now replaced with the result of the new operation. The same thing doesn't necessarily happen in 32-bit MIPS. For example, the value of one register is increased by the immediate value in MIPS and saved in a different register.

Operation code in the Jump/Branch instruction for MIPS16 is 5 bits compared to 6 in 32-bit MIPS (see Figure 3b). The target field, which specifies a jump or branch destination, is reduced to 11 bits from 26.

Consequently, the number of different kinds of branch specified in 32-bit MIPS is curtailed. With the six-bit OP in 32-bit MIPS, there can theoretically be up to 64 different types of branches. In MIPS16 with a five-bit OP, there are less than 32.

The 32-bit MIPS 26-bit target field permits a wide (228 bytes) range for jumps and branches. In MIPS 16, it's 213. If a jump needs to be further out than this in MIPS16, a multi-instruction sequence is needed to achieve the same effect.

As for register types, 32-bit MIPS has a 6-bit OP, three register fields (each could be any one of 32 registers), and five bits for the shift amount (SHAMT) field, which specifies instructions that perform arithmetic or logical shifts. And, as Figure 3c shows, a 6-bit function modifies the basic operation.

MIPS16 RR-type has a 5-bit OP, 3-bits each for rs and rt, and a 5-bit function. The RRR-type maintains the 5-bit OP and the 3 bits each for rs and rt. However, it adds a 3-bit rd register field, although the function field is restricted to only 2 bits for modification purposes.

Sometimes, instruction steps taking 32 bits in 32-bit MIPS can be directly encoded in either the RR- or RRR-type 16-bit formats. But, there will be cases in which a combination of instruction steps is attempted.

However, they won't be accommodated in either RR or RRR. When that occurs, a multi-instruction sequence must be implemented. RR and RRR types are used extensively for basic integer manipulation (e.g., ANDs, ORs, and XORs).

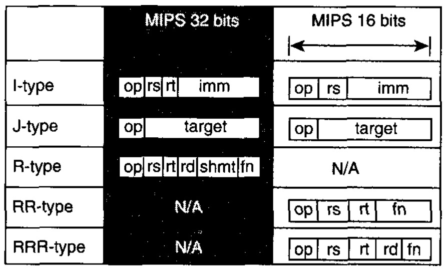


Figure 2—*Instruction types are similar between the 32-bit MIPS ISA bus and MIPS 16 ASE. For example, in the immediate-type instructions, only one register field is omitted from the normal 32-bit immediate-type instruction.*

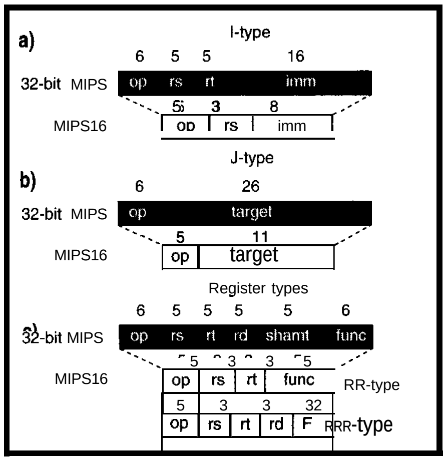


Figure **3a**—*Comparing the 32-bit MIPS* and *MIPS16 immediate-type instruction, 5-bit regisfers in 32-bit MIPS become one 3-bit register in MIPS16, while a 6-bit operation* **turns** *info a 5-bit operation in MIPS16. b-/n Jump/Branch instructions, a 26-bit target field becomes an 1 i-bit target field in MIPS16.* **c**—*In the* register *RR-type* instructions, *32-bit MIPS* has a *6-bit* operafion field and *three register fields, each of which could be any one of 32 regisfers.*

OTHER CONSIDERATIONS

Cutting back on system memory by using a compact-code CPU is only part of the overall solution. Other problematic areas are architectural flexibility as well as determining how reusable and scalable an ASIC processor core can be. A design may require additional or different functionality for competitive differentiation.

Here, the designer can choose from a variety of cores and building blocks. For example, a fast multiply/divide unit (MDU) can be added in a communications application to perform several modem tasks in firmware. Of course, adding a feature tacks on more cost savings to the design since it eliminates a separate DSP chip or modem chipset.

Other useful functional units include peripheral controllers like ROM or DRAM and/or video controllers and I/O. In effect, the designer can pack considerable functionality of various flavors onto a design, minimizing the amount of supporting components needed.

A reusable RISC microprocessor core is stripped down to only the IU and register filer. A flexlink interface enables systems designers to partially customize the CPU core, thus making it extendable.

Next, a coprocessor interface (COP) turbocharges the RISC core's computing power by enabling engineers to tightly hook up special-purpose processors.

As well, access to the CPU bus (CBus) enables different building blocks to be designed around the CPU. Consequently, considerable latitude is provided for designing in a variety of cores and building blocks to comply with specific NC designs.

The flexlink interface of the TR4101 TinyRISC microprocessor core opens the door to the reusability feature. This interface can extend the standard MIPS instruction set to many other different instructions.

Typical instructions that can be added to the MIPS core are multiply/add, multiply/subtract, find first set bit, find first clear bit, and saturate instructions, among others. It can also be used for multiple-cycle instructions, although it's more appropriate for single-cycle instructions.

The flexlink interface provides a way to select and bolt on to the CPU any number of designer-defined functions (e.g., a multiplier, barrel shifter, or pixel-manipulation engine).

This way, the systems engineer can open up the instruction set and modify it without changing the core processor logic.

TinyRISC's flexlink interface makes this all possible. Here's how it works.

The CPU presents two subfields of the instruction register and rs and rt operands at the flexlink interface when each instruction's execute stage starts. Flexlink decodes the instruction, executes it, and writes the results back onto the rd bus at the end of the cycle.

The CPU handles the register file update during the write-back stage. If the instruction needs more than one cycle, flexlink can either stall the CPU via the normal stall mechanism or write results into its own holding registers, which need to be read later with some user-defined flexlink instructions.

MAKING THE SWITCH

Cygnus Tools will be the first of several software manufacturers to support the MIPS16 ASE. These tools are based on the Open Software Foundation (OSF) GNU tool chain.

Cygnus intends to provide the assembler, compiler, linker, locator, debugger, and architectural simulator for the MIPS16/TinyRISC architecture. These tools will be made available free via the Internet in early 1998.

The transition from a 32-bit MIPS development environment to that of MIPS16 is made simple. The software developer can begin with a 100% MIPS application and convert the routines to MIPS16 step by step.

Compilers, linkers, and debuggers have full support for both instruction formats. Basically, a flag to the compiler controls whether it generates 32-bit MIPS or MIPS16 instructions. The remainder of the code-development process (i.e., linking and debugging) is the same for both MIPS, MIPS16, and mixed binaries. ▣

Bill** Jackson **is the marketing manager for embedded applications at Silicon Graphics-MIPS Technologies. You may reach Bill by E-mail at** jackson@mti.sgi.com **or** by phone **at (415) 933-7368.

Reynaldo (Rey)** Archide **is the product manager for the** MiniRISC/TinyRISC **line of microprocessor cores at LSI Logic. He has also held positions at Toshiba, NEC, Zilog, and Philips. You may reach Rey by E-mail at** archide@lsil.com **or** by phone **at (408) 433-7987.

SOURCES

TinyRISC TR4100 family
LSI Logic
155 1 McCarthy Blvd.
Milpitas, CA 95035
(408) 4338000
Fax: (800) 457-4286
www.lsilogic.com

MIPS16 **ASE** tools
Cygnus Solutions
1325 Chesapeak Terrace
Sunnyvale, CA 94084
(408) 542-9600
Fax: (408) 542-9699
www.cygnus.com

I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

EMBEDDED PC
OCTOBER 1997

34 Nouveau PC

39 Intelligent Data [illegible]
Jim [illegible], [illegible]szky,
& Ro[illegible]

44 Precision Timing and [illegible]
Steve Lubbers

56 PC/104 Quarter
PC/104 [illegible]
[illegible]

60 Applied PCs
Managing a NASA
Plant Growth Chamber
Part 2: Coordinating Sensors and
Processing Data
Fred Eady

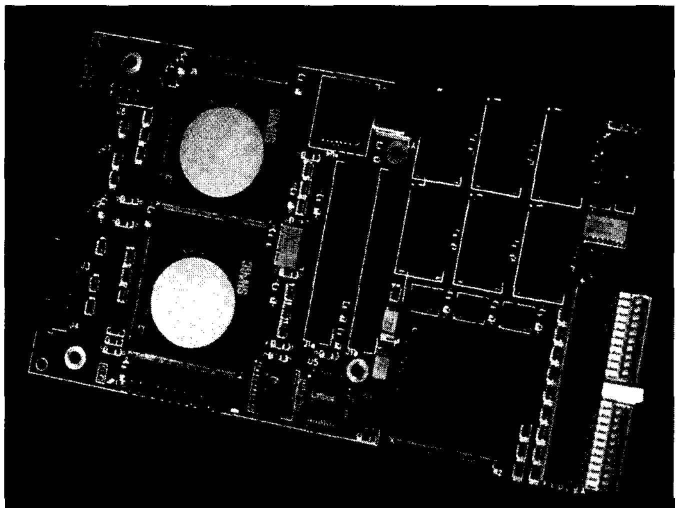
DSP BOARD

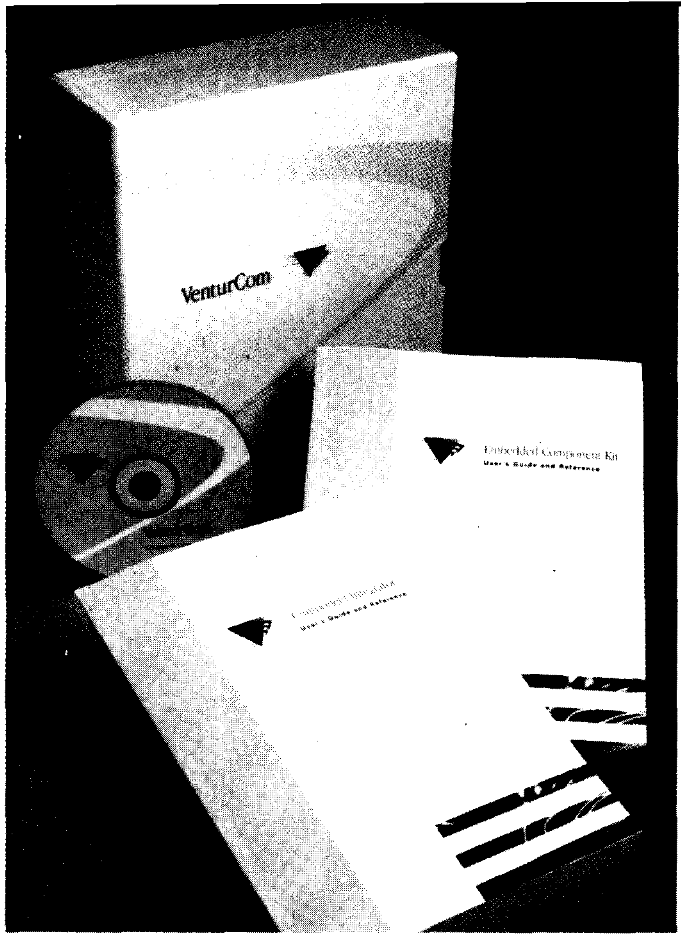
Snaggletooth **CompactPCI** is the industry's first product to combine the fast floating-point performance of Analog Devices SHARC DSP with a CompactPCI form factor. The 240-MFLOPS card features a pair of 40-MHz ADSP-2 106x SHARC processors, each including 2 or 4 Mb of on-chip dual-ported SRAM. The board also features up to 5 12K x 48 bits of zero-wait-state SRAM, four external link ports, and a BITSI mezzanine site for off-the-shelf I/O interfaces. The two SHARCs share a common processor bus, giving each processor access to the optional bank of SRAM, BITSI mezzanine I/O devices, and the internal SRAM of the other processor.

Snaggletooth CompactPCI is supported by source-code development tools, including Analog Devices' SHARC-ANSI-compliant compiler, assembler, linker, simulator, and source-code debugger. True real-time in-circuit emulation is available with the optional EZ-ICE emulator. An IEEE 1 149.1 -compatibleJTAG connector enables nonintrusive DSP control and debug. The DSP21 k Toolkit also provides developers with C-callable host I/O functions, DSP functions, example code, and diagnostic utilities under Windows 95, Windows NT, and QNX operating systems. A DSP2 1 k Porting Kit is also available to facilitate Snaggletooth support on additional platforms and operating systems.

The Snaggletooth CompactPCI sells for $5595.

BittWare Research Systems
33 N. Main St.
Concord, NH 03301
(603) 226-0404
Fax: (603) 226-6667
info@bittware.com
www.bittware.com **#510**





APPLICATION DEVELOPMENT SOFTWARE

Component Integrator is an easy-to-use development tool for integrating, configuring, and building dedicated Windows NT target systems. Its optional OS extensions-Embedded Component Kit (ECK) and Real-Time Extension (RTX)-provide vital capabilities for real-time and embedded operations for Windows NT.

System developers can integrate custom applications, COTS software, ECK, and RTX extensions with Component Integrators' knowledge base of Windows NT components. An easy-to-use graphical interface simplifies selection and configuration of software components. A bootable target can be built on disk or flash media, or an installation image can be produced that can be loaded onto a target from a network server or CD-ROM.

Component Integrator reduces target-system size by enabling the developer to choose only the Windows NT components required for the application. Typical uses include industrial control, telecommunications, and medicine.

Pricing for Component Integrator starts at $9500.

VenturCom
215 First St. • Cambridge, MA 02142
(617) 661-1230 • Fax: (617) 577-1607
info@vci.com • www.vci.com **#511**

NouveauPC
edited by Harv Weiner

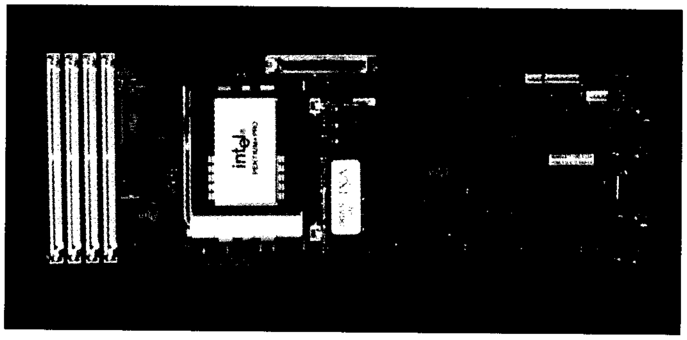
PENTIUM PRO SINGLE-BOARD COMPUTER

The **SB686P(V)** is a single-board computer that offers a complete array of onboard controllers, plug-n-play compatibility, and optional SVGA video. It offers an excellent price/performance value for embedded-systems designers and industrial-computer users in computer telephony, telecommunications, and factory-automation and process-control applications.

Two board versions are available—with or without onboard SVGA video. The version including SVGA features 1-MB RAM standard for PCI video performance. Both boards are PCI 2.1 compliant and offer a 200-MHz CPU clock speed, 256-KB cache, the ability to selectively disable all onboard peripherals, and watchdog timer. As well, it's compatible with Unix, Windows NT, and other 32-bit operating systems. Forward compatibility to Socket 8 ZIF processors is also provided for future upgradability.

Dual EIDE controllers permit two drives each on PCI Local bus. Other standard features include floppy-disk controller, Ultra, Ultra/Wide SCSI-3 controller, SVGA video, dual 16550-compatible serial ports, bidirectional parallel port, and ports for PS/2 mouse, keyboard, and speaker.

The SB686PV200 (SVGA video version) sells for **$2859**. The SB686P200 is priced at **$2709**.



Industrial Computer Source
6260 Sequence Dr.
San Diego, CA 92121
(619) 677-0877
Fax: (619) 677-0895
info@indcompsrc.com
www.indcompsrc.com

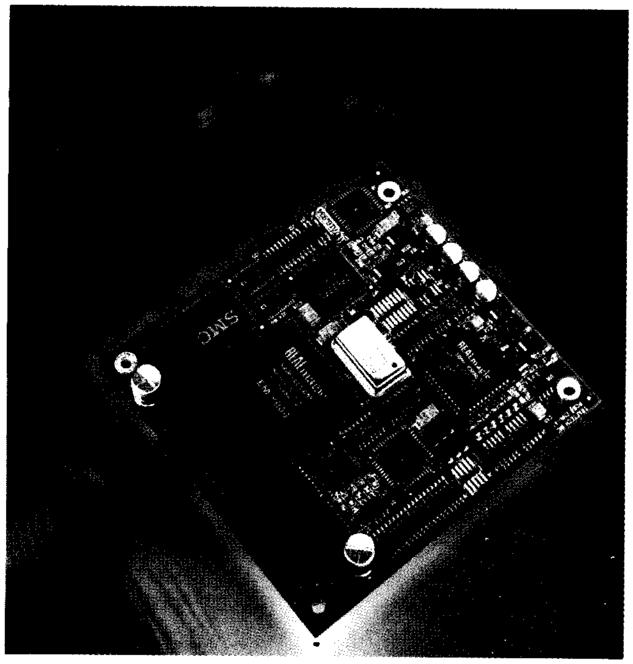
#512

EMBEDDED MULTIMEDIA CARD

Traftech's Chinook, a 3.6″ × 3.8″ PC/104 form-factor card, offers MPEG, Sound Blaster, and COM port all on one unit. It features full audio/video synchronization with MPEG-1 system-layer decoding using the REALmagic EM8000 decoder chipset. Smooth horizontal and vertical scaling functions are also included. Video-decompression functions supported include Huffman decoder, inverse quantizer and zig-zag, inverse DCT, and full-motion compensation. With REALmagic EM9000, the Chinook also supports full analog video overlay. As well, this embedded multimedia solution features a Crystal CS4236 Sound Blaster-compatible sound interface with two additional COM ports.

The Chinook is fully plug-n-play compatible, and it permits seamless insertion into Windows 95 and Windows NT applications. Given the compact PC/104 form factor, the Chinook is useful for video-on-demand, multimedia kiosk, and other space-constrained MPEG requirements. In addition, the card offers the full Windows 95 and Windows NT drivers set for both Sound Blaster and MPEG portions. Available options include SRS 3D sound, Q-sound wavetable, and the ability to blend Sound Blaster and MPEG audio outputs.

The PC/104 Chinook module has a list price of **$695**.



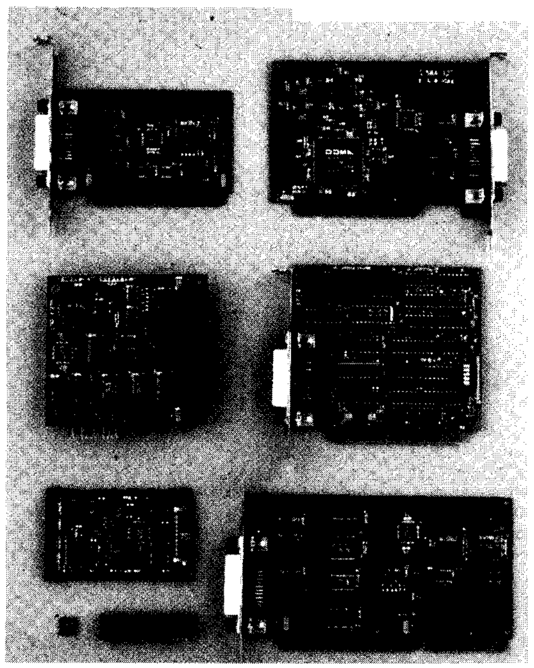
Traftech, Inc.
6981 Millcreek Dr., Ste. 30
Mississauga, ON
Canada L5N 6B8
(905) 814-1293
Fax: (905) 814-1292
info@traftech.com
www.traftech.com

#513

Nouveau PC

GPIB INTERFACE FAMILY

ComputerBoards hdS introduced a family of **GPIB** (IEEE-488 standard General-Purpose Instrument Bus) interface products based on their **CB7210.2** GPIB controller chip. The family includes the chip, three versions for ISA bus, one for PCI bus, a PCMCIA card, and a PC/104 module.



All versions use this new chip, which features a high-speed state machine and 1024-word FIFO to implement fast transfer rates. Maximum sustained data-transfer rates are greater than 1 MHz with 4-m cables and exceed 1.5 MHz with shorter cables.

The CB72 10.2 chip is one of the few IEEE-488.2~compliant talker/listener/ controller chips available. Designed entirely in VHDL code, the chip implements the GPIB state machine defined in the specification. Since it is backwards compatible with the NEC7210 (commonly found on older GPIB interface cards), replacing this chip with the CB72 10.2 brings that board up to new spec compliance. The chip is priced as low as $8 in quantity, and VHDL source code is available for purchase with low-cost license fees.

All boards are bundled with a complete NI488.2-compatible GPIB library for DOS, Windows 3.1, and Windows 95, ensuring functional compatibility with popular languages and applications. The boards are also supported by a variety of third-party software, including HP VEE and Test Point.

The boards each list for $299. A low-cost ISA version that eliminates the FIFO or high-speed state machine and has a maximum sustained data transfer rate of 300 kHz sells for $199.

ComputerBoards, Inc.
125 High St., Ste. 6
Mansfield, MA 02048
(508) 261-l 123
Fax: (508) **261-** 1094
info@computerboards.com
www.computerboards.com

#514

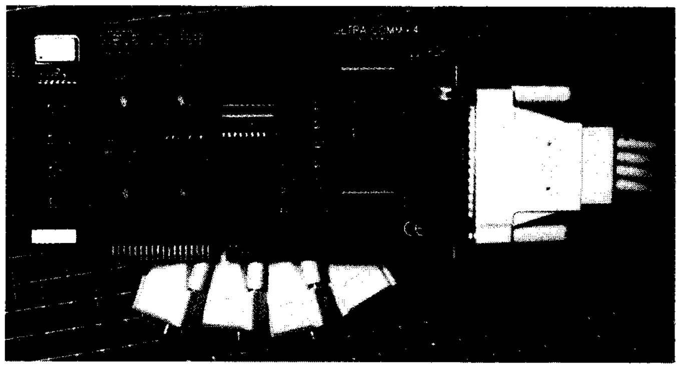
AUTOMATIC RS-485/-422 CARD

The **ULTRA-COMM+422** is a serial I/O adapter for the IBM PC and compatibles. It provides four RS-485/-422 DOS, Windows 3.*x*/95/NT, QNX, and OS/2-compatible serial ports with the ability to connect up to 3 1 RS485 devices to each port.

The OS views each DE-9 port as a COM port, so the standard COM driver can be used for RS-485 communications. Its auto-initialization circuit enables the RS-485 line driver during character transmission and disables it afterward. The adapter lets the RS485 line be managed transparently without user or software intervention and eliminates RS-485 network contention.

The ULTRA-COMM+422 supports the RS-422 specification, providing long-distance (4000') communications and noise immunity. User-selectable AT IRQs and an onboard interrupt status port let the user set each port to a separate IRQ or share a single IRQ over multiple ports. A versatile range of address settings provides seamless integration into existing systems.

The card supports standard PC data rates up to 460.8 kbps, providing support for ultra-high-speed serial applications. The 16550 UARTs are standard. The new 32-byte FIFO 16650 UARTs and 64-byte FIFO 16750 UARTs are available as options. One- and two-port versions are also available.



The ULTRA-COMM+422 sells for $369 ($319 without the automatic circuit).

Sealevel Systems, Inc.
P.O. Box 830 • liberty, SC 29657
(864) 843-4343 • Fax: (864) 842-3067
www.sealevel.com **#515**

NouveauPC

MULTICHIP EMBEDDED MODULE

The **SMX/386 OEModule** provides all the functionality of a single-board computer in a 240-pin surface-mount component, plus a preloaded BIOS, embedded ROM-DOS, and an internal resident flash disk. The module includes ail standard motherboard features, including a 40-MHz 80386SX CPU, 2-MB DRAM, IDE hard-disk controller, and floppy-drive controller. It also has two serial ports, a bidirectional parallel port, full AT keyboard, speaker interface, real-time **clock, watchdog timer, full** ISA (PC/I 04) bus, and 256 KB of user-available flash memory.

The SMX/386 is a multichip module containing more than 80 components. The 240-pin package is used to bring out the standard I/O connections. There are no licenses to negotiate. Powering up the SMX/386 results in a DOS prompt. A development board is available to facilitate testing software on a target CPU.

Pricing for the SMX/386 starts at $104 in 1 OOO-piece quantities.

ZF **MicroSystems,** Inc.
1052 **Elwell** Ct.
Palo Alto, CA 94303
(415) 965-3800
Fax: (415) 965-4050
www.zfmicro.com

#516

NouveauPC

Board with DOS..?

REMOTE
DATA
ACQUISITION

INDUSTRIAL
NETWORKING

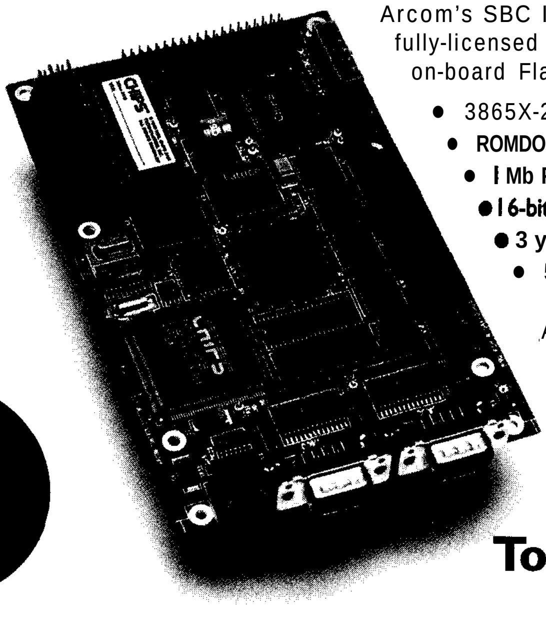
MACHINE
CONTROL

Arcom's SBC IO4 **single** board **computer is** complete with fully-licensed ROM-DOS 6.22 pre-loaded into on-board Flash memory.

- 3865X-25 **or 486SXLC2-50**
- ROMDOS Flash Filing System
- I Mb Flash & 2Mb DRAM standard
- I 6-bit PC/ I04 Interface
- 3 year Product Warranty
- 5 year Product availability (386SX)

Arcom's wide range of PC/ IO4 modules:

- Digital I/O
- Multi ADWDAC
- Multi-relay
- CANbus - DeviceNET



ToII**Free: 888-94 I-2224**

Fax: 8 16-94 I-7807 Fax-on-Demand: 800-747-I 097
E-mail: **sales@arcomcontrols.com** Web: **www.arcomcontrols.com**

Arcom CONTROL SYSTEMS

#116

Jim Blazer,
Janos Levendovszky,
& Robert Haris

Intelligent Data Acquisition

Strapped by a narrow bus and an overly taxed CPU, embedded PCs have traditionally been unable to compete with high-end data acquisition and processing. But, with a DSP coprocessor, the PC is breaking into this niche.

Ever since the first data-acquisition card was released, hardware and software engineers have been keen on producing data-acquisition systems with higher speed and more versatile features.

The processing ability of PCs, however, soon became a serious obstacle to the high-speed data acquisition and signal processing required by complex industrial applications. This limited performance was especially apparent in multitasking embedded applications, where the finite resources of a single-host CPU had to be distributed among several tasks.

Embedded-application designers have been waging a never-ending war to balance acquisition speed on the board side and processing power on the host side. From our perspective, however, the way to hammer out the discrepancies between board and host speed is through Distributed Intelligence (DI).

DI is a design philosophy that brings intelligence to the data-acquisition board. In this way, the host is freed from moving data around, managing memory, and performing signal processing.

With onboard intelligence (i.e., CPUs and DSPs), these tasks are carried out locally. The concept of DI therefore localizes logical and arithmetic power, removing it from the "divine monistic" reign of a single-host CPU.

The first implementation of this concept was to place a DSP, microcontroller, or microprocessor on a data-acquisition board with a FIFO for host communications. These boards either use a proprietary OS or none at all, leaving all programming to the user.

However, if we take into account what's missing and what can be improved, DI gains an ideal balance for embedded applications. To implement this idea, we focused on:

- adding intelligence by having an embedded PC with an easy-to-use OS and a DSP for localized signal processing.
- replacing the FIFO as a sequential communication medium with a shared memory guaranteeing a broad data highway between

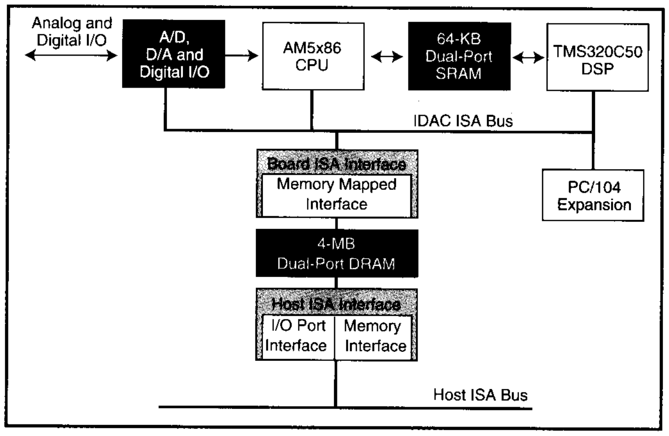


Figure 1—Onboard embedded PC and DSP, analog and digital I/O, and dual-port shared memory communications make the IDAC a perfect match for data-acquisition and process-control applications.

the board and host and letting the CPU read only required data
• using MS-DOS, Windows, QNX, or other PC-compatible RTOSs running on the board's CPU
• enabling hierarchical or parallel processing by having analog inputs available to all the system processors
• providing a direct communication path between all system processors
• enabling multiple boards in a single host with multiboard synchronous A/D conversions

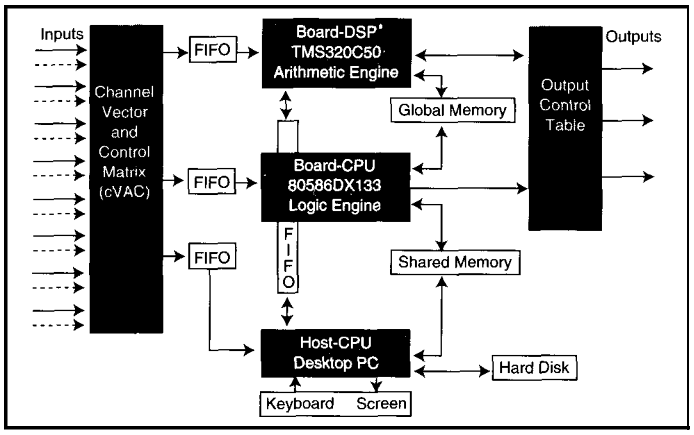


Figure **2—IDAC's** unique processing and communication architecture *gives three* ***CPUs access to*** acquired ***data*** in concert, providing parallel or hierarchical ***data*** processing. The ***cVAC*** mofrix sends *inputs* ***(on*** *the* ***left) to any or all processors, each*** performing its own algorithm. Outputs (on the right) are controlled by ***onboard*** processors.

To integrate these components in versatile and intelligent data-acquisition and control systems, RTD designed the IDAC5250 with an onboard TMS320C50 fixed-point DSP unit and a 133-MHz Am5x86 CPU. The skeleton of the IDAC5250 is illustrated in Figure 1.

The processor's internal cache and floating-point processor is coupled with 8-MB DRAM, serial and parallel ports, watchdog timer, PC/104 expansion bus, and flash solid-state disk with Windows and DOS compatibility. Thus, it provides a versatile embedded PC for data acquisition.

Since different jobs can be assigned to the onboard CPU and DSP, the hardware structure is flexible enough to mirror the object-oriented structure of the software.

Photo 1 presents RTD's IDAC5250, a board implementing the DI concept. It's a highly intelligent signal-processing card that includes an embedded PC and DSP, shared memory, multiboard sync clocks and triggers, and an advanced analog unit. Its hardware potentials are extended by sophisticated algorithms. With these capabilities, it can serve at the core of a PC-based, high-speed industrial controller operating under Windows.

Let us illustrate more of the DI concept by examining the IDAC board in further **detail.**

WHAT IS IDAC?

DI brings intelligence closer to the places where data is acquired and real-time signal processing is needed. It increases system performance by sharing tasks optimally between the different signal-processing units. To implement intelligence on IDAC, several areas of technology were needed:

• leading-edge versatile analog front end (e.g., ADA3400)
• advanced embedded CPU technology (e.g., CMV586DX133) to serve as an onboard logic engine
• DSP coprocessor technology implemented as an arithmetic engine that provides the necessary computational power to perform sophisticated real-time digital signal processing on the collected data
• shared and global memory to help create large, flexible and parallel access from both the host and board sides. A direct-host FIFO is included for DSP communications.
• shared memory and direct serial communications for multiple IDAC installations in a single host running Windows

Texas Instruments' TMS320C50 board DSP-the arithmetic engine-provides 64K words of zero-wait-state program memory, 32K words of zero-wait-state local data, and 32K words of one-wait-state, hardware arbitrated global data memory.

All DSP memory, including DSP internal memory, is mapped into the board-CPU extended memory. Hardware-arbitrated access to the global data memory assures uninterrupted DSP operation, while the board CPU reads or writes the global memory.

The ADC and all three DACs have 1024-sample independent FIFO buffers to process bursty data. The onboard PC/I 04-bus connection between board CPU, DSP, and analog I/O unit enables the IDAC5250 to function as a stand-alone machine. However, its architectural strengths become clear when placed in a host computer.

In contrast to FIFO communications, the 4 MB of hardware-arbitrated shared memory enables simple and fast communication between the CPU and host. With its 8-MB DRAM, the CPU can run DOS, Windows, QNX, and other operating systems requiring DOS compatibility.

This relatively large memory also supports high-speed data-acquisition applications when the collected data is streamed into the memory. A 32-pin solid-state-disk socket supports flash, SRAM, or large flash drives like M-Systems' DiskOnChip.

The **analog** I/O features 16 single-ended and 8 differential input channels with 1.2MHz 12-bit resolution or 250-kHz 16-bit resolution and the 1024-sample FIFO. The programmable input ranges are ±5, ± 10, or O-l 0 V, while programmable channel

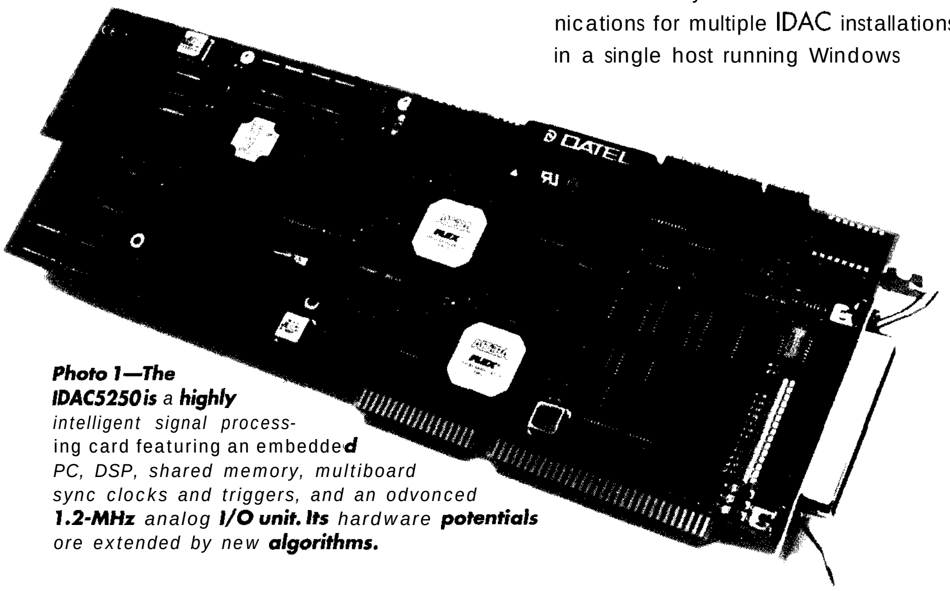


Photo 1—The IDAC5250 is *a* ***highly*** *intelligent signal processing card featuring an embedded PC, DSP, shared memory, multiboard sync clocks and triggers, and an odvonced* ***1.2-MHz*** *analog* ***I/O unit. Its*** *hardware* ***potentials*** *ore extended by new* ***algorithms.***

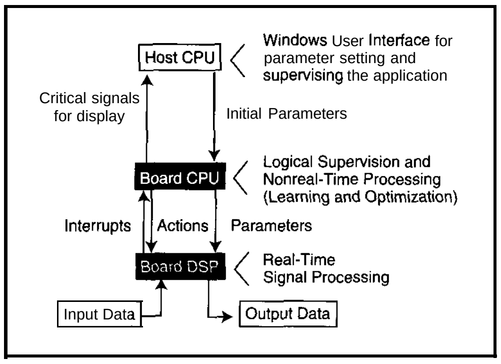

Host CPU
Windows User Interface for parameter setting and supervising the application
Critical signals for display
Initial Parameters
Board CPU
Logical Supervision and Nonreal-Time Processing (Learning and Optimization)
Interrupts
Actions
Parameters
Board DSP
Real-Time Signal Processing
Input Data
Output Data

Figure 3—IDAC** provides a division **of tasks** between **three** processors in **hierarchical data processing**, where the DSP **performs** real-time signal analysis, the embedded PC handles **learning and optimization, and the host provides a user interface.

gains are 1, 2, 4, 8, 16, 32, 64, and 128. The three analog voltage outputs with independent 1024-sample FIFO buffers provide high-speed analog feedback.

The digital control is implemented in an EPLD that has one 8-bit I/O port and eight bit-programmable DIO lines. These lines can generate an interrupt when a bit-or combination of bits-either changes value or matches a programmed value. The buffered lines can sink 24 mA and source 12 mA.

SIGNAL PROCESSING HIERARCHY

With locally implemented intelligence onboard, an intelligent data-acquisition board can multiply the performance of a single host.

This increased performance is most apparent when performing hierarchical signal processing (HSP) and automated reasoning while preserving real-time operations at high sampling rates.

With IDAC5250, these capabilities are ensured by the various signal paths summarized in Figure 2. As you see, there are several possibilities to assign processing units to the collected data. Given this flexibility, IDAC is capable of HSP.

HSP ranks one processing unit above another to perform automated reasoning. Information is passed from a lower layer to a higher one in a more condensed and abstract form.

This feature is a great aid to process-control engineering when a huge amount of observed data is to be converted into a few logical variables (regarding the quality of control). Based on these logical variables, a control action needs to be taken.

HSP ensures the mapping from raw data into abstract categories (e.g., the state of the system and quality of control) for these applications. The classic example is condition-based maintenance that monitors the state of complexelectromechanical systems.

SIGNAL PROCESSING

Signal-processing power often proves to be the bottleneck of embedded industrial applications. When intelligence is involved, optimal performance depends on how tasks are distributed among processing units.

Taking advantageof IDAC modular hardware structure, the board can work in the hierarchical order shown in Figure 3. First of all, the DSP is engaged with real-time signal processing, and if an event requires it, the DSP sends an interrupt to the board CPU.

The board CPU supervises the application and takes necessary actions when interrupts arrive from the board DSP. Besides supervising theapplication, the board CPU does nonreal-time processing (e.g., learning and parameter optimization) when fast computations aren't needed.

The host CPU offers an easy-to-use interface under Windows. The user can then set the parameters for a concrete application.

These parameters are loaded from the host to the board. In case a particular event occurs in the course of running the application, the host receives the collected or processed data from the analog or DSP unit, respectively, and displays it under Windows.

The user can then receive information explainingwhya particulareventoccurred. After detecting critical instability in PID control, for example, the signals of a control loop might be sent to the host CPU for further analysis.

OPERATING SYSTEM

Intelligent data-acquisition boards with DSPs alone or nonPC-compatible microprocessors exclude the use of DOS, QNX, or other RTOSs. But, DOS and other PC-compatible OSs offer some advantages since users can use their favorite desktop-PC compiler and debugging tools.

Running the same operating system on the target and development systems makes it easy to port the developed application program to the board. And, many RTOSs provide built-in networking and multitasking.

While it's possible to run Windows on the IDAC's CPU, it offers poor real-time multitasking performance. Real-time operation stipulates acting deterministicallywithin a given length of time.

Windows simply doesn't provide real-time performance when events follow each other in millisecond range with no assurance that they'll ever be processed. Obviously, the **fast** data acquisition and high sampling rate of real-time digital signal processing aren't conceivable under Windows.

Nevertheless, you can take advantage of Windows' graphical features. The host's user-friendly interface can supervise an application, set initial parameters, and display the obtained dato offline.

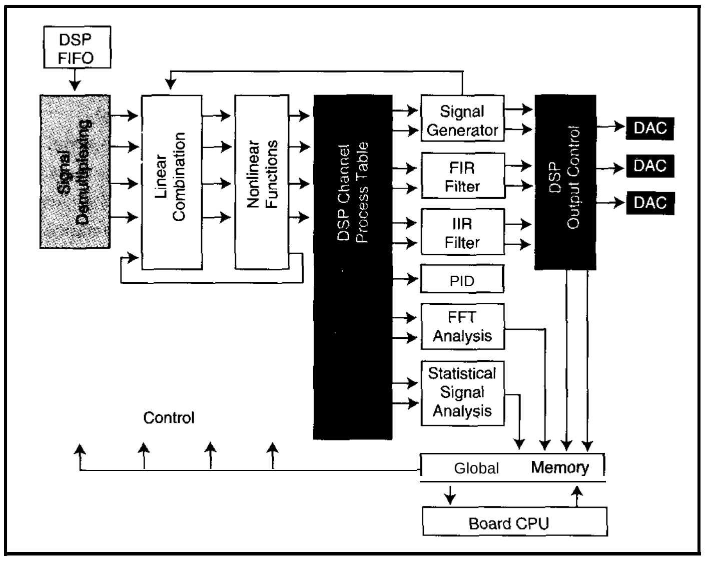


***Figure 4—The** multi-&king library enables the user to call DSP operations without programming the DSP and to perform multiple tasks.* This **principle** *is implemented by dynamic **transfers** of control that ensure the execution of the different tasks.*

REAL-TIME MACHINE

Missioncritical and complex applications frequently require real-time operations that usually lie beyond the capabilities of traditional **DOS** and single CPU embedded systems.

To turn IDAC into a real-time machine, the designers took advantage of the relatively high speed of the DSP compared to the CPU. In other words, the DSP carries out the real-time processing.

To enable the user to exploit the DSP's processing power without having to get into the details of assembly programming, two approaches were pursued.

First, single-task DSP operations let the user run single applications on the DSP (e.g., filtering, FFT, etc.). In this approach, DSP operations are activated by function calls in the user program. These C calls activate ready-made programs (written in assembly language), which are loaded into the DSP program memory with the proper parameters and executed.

As well, a multitasking function library was written for the board DSP. The user can therefore call DSP operations without programming the DSP and perform multiple tasks, if the tasks can be accommodated in a time frame.

This principle is implemented by dynamic transfers of control that ensure the execution of different tasks. Figure 4 depicts the structure of the multitasking approach.

To exploit the advantages of a full-fledged RTOS (i.e., notonlymanaging tasks, but also managing CPU, memory, timer, interrupts, DMA, and networking), run QNX or another real-time system on an IDAC. If you decide to emphasize networking, you can combine the signal-processing power of many IDACs.

DISTRIBUTED INTELLIGENCE NOW

Real-time data acquisition and control using the most popular GUI-Microsoft Windows-mandates special hardware adaptations. Traditional passive data-acquisition boards require constant and immediate host intervention. Faster buses, such as PCI, create an additional processing burden and do not improve real-time data acquisition and control.

This dichotomy brought us to the dawn of DI for the PC. Data-acquisition boards with onboard embedded PCs and open OSs that provide real-time processing power and easy-to-use algorithm libraries will dominate this environment. RTD's ultiMate Series IDAC5250 illustrates the possibilities for new data-acquisition products as we embrace DI and make Windows the optimum data-acquisition and control platform. EPC

The *international hardware and software design team included John Hazel, Dr. Istvan Koller, Dr. Kalman Elek, Dr. Josef Goal, Attila Farkas, and Paul Ganter. Financial assistance was provided by the Ben Franklin Technology Center, State College, PA.*

As director of engineering at Real Time *Devices USA,* Jim *Blazer manages a multi-*national engineering team **designing intelligent data-acquisition hardware and *software* and embedded DOS-based computers. He also serves as a director of the PC/104 Consortium, secretary of the IEEE P996.1 Technical Committee,** and **chair of the PC/104-Plus** *committee. You* **may** *reach* **Jim at** *jblazer@rtdusa.com.*

Dr. Janos levendovszky is director of *RTD* **USA BME laboratories in Budapest, Hungary, and Szechenyi Professor of Electrical Engineering at Technical University of Budapest. He teaches communication theory, adaptive** *signal* **processing, and neural** *network* **theory and conducts research on neuralnetworks, signaldetection theory,** *adaptive signal processing and control,* **statistical resource management, and** *CAC for* **ATM networks. You may reach Janos at** *levendov@hit.* **bme. hu.**

Robert Haris, president and CEO of Real Time Devices USA, *spent* 25 years *as a* Boeing **senior aerospace engineer and international business** *consultant before coming to* RTD *in* 1990. **He** *defined the IDAC architecture and structured the multinational task force.*

REFERENCES

Real Time Devices USA, *IDAC5250 Datasheet,* State College, PA, 1997.

Real Time Devices USA, Introduction to *Intelligent Data Acquisition and Control,* State College, PA, 1997.

SOURCES

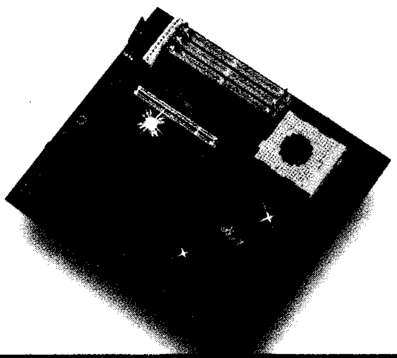
IDAC5250

Real Time Devices USA, Inc
200 Innovation Blvd.
State College, PA 16804-0906
(814) 2348087
Fax: (8 14) 234-52 18
sales@rtdusa.com
www.rtdusa.com

IRS

413 Very Useful
4 14 Moderately Useful
4 15 Not Useful

• 7 ISA slots
• 486 support



G486VPC Motherboard

The **G486VPC** motherboard combines quality and affordability with an industrial design to meet your needs. Seven full length ISA slots ensure expandability for the cards that you use.

Our engineering staff will gladly discuss custom motherboard designs. FCC & UL certified systems are also available.

- Six full length 16-bit ISA, One shared 8-bit ISA/PCI slot
- Intel, AMD & SGS-Thomson 486 CPU support
- VIA chipset
- Up to 64MB RAM, 256KB cache

(800) ASK-DFI1
www.dfiinc.com

DFI®

#118

Steve Lisberger

Precision Timing and I/O

When precise timing of events is required, *software* is not the way to go since each call to the clock a/so takes up time. Steve therefore moves this operation into the hardware and gains better than *10-µs* resolution.

We live in an asynchronous world. Things happen when they happen. And, now that we've abandoned the simplicities of DOS for operating systems like Windows NT, Windows 95, and Linux, asynchrony has taken over the world of computing, too.

Although we'vegained additional power with these operating systems, it comes with a loss of control over when things happen. And, it creates a new class of problems for timing asynchronous events.

THE TIMING PROBLEM

How can we measure the time intervals between events? For timing external events, the traditional strategy is to use a real-time clock. The clock is programmed to count with the desired resolution and interrupt the processor when a pulse occurs on an input line.

The interrupt service routine collects the current value of the hardware timer at an I/O port and stores it for later data analysis. I know of one board that includes a tiny FIFO so up to four events may be stored if the software responds slowly to the interrupts.

An elegant solution to the external-event timing problem is the PC board presented in Photo 1 and Figure 1 (complete schematics available on Web site). It precisely tracks the times of incoming TTL pulses without any intervention by application software, operating system, bus, or CPU. To make the board maximally independent, it hasenough memory to store the times of many pulses. I use this timing board in this article.

For timing software transactions, the task is a little moredifficult. It's simple to call `clock()` at the beginning and end of a transaction and measure time as the difference, but this technique provides only 10-ms resolution. Plus, Heisenberg's uncertainty principle is at work here.

Almost any method of timing software takes time. `clock()` has as little overhead as you can expect. On my 100-MHz Pentium, it takes an average of 320 µs under Windows 95 (range 307-420) and 71 µs under Windows NT 4.0 (range 65-158). Bettertemporal resolution and (in Windows 95) less interferencerequireanothersolution.

An ideal solution to the software-timing problem would present time in microseconds at a 6-byte memory-mapped port. This would allow microsecond timing for up to 1500 days withoutoverflowing thecounter (a 4-byte port gives only 72 min.).

If the software driver were written for parallel access by multiple processes, then every process could time

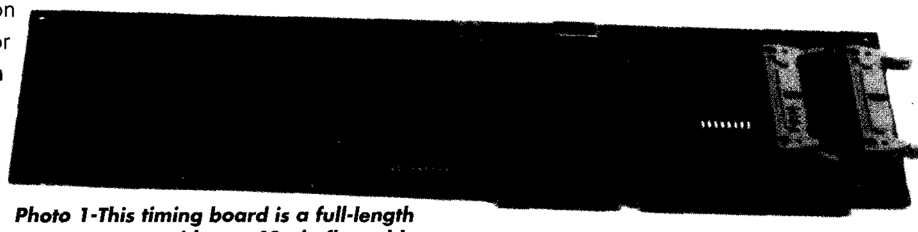


***Photo 1**-This timing board is a full-length ISA-bus boord **with** two **40-pin flat-cable** connectors for inputs and outputs.*

itself and report the results to a central data analysis program. The API could simply be:

```
get_me_a_time(TIME*
    pointer_to_time_data)
typedef struct{
  unsigned long low32;
  unsigned short high16; }
TIME;
```

The timing board enables a solution to a software timing problem that falls only slightly short of this ideal.

SOLUTION EVOLUTION

As a neuroscientist, I faced this general timing problem in 1981 when I needed 10-µs resolution in timing intervals between electrical events given off by brain cells.

In the lab, we were using a DEC 1 1/23 for real-time data acquisition and experiment control. The CPU's slowness was the main problem. I used a Schmitt trigger to convert the electrical activity of nerve cells into TTL pulses, but the pulses had to be timed by the computer.

The obvious solution of counting time in an interrupt handler for a real-time clock board was impractical because interrupts occurring at even 10 kHz (every 100 µs) dominated the CPU and precluded acquiring any analog data or using the computer to control the ongoing experiment.

In 1981, I solved this problem in a makeshift way, using a 16-bit parallel I/O board to communicate with an external logic circuit. Over the years, as we moved from the 1 1/23 to the 80x86 PC, the solution has evolved into a system that still implements the 198 1 strategy but is easy to use and readily available.

The system comprises a full-length ISA-bus timer board that uses 16 bytes of I/O port and one IRQ to communicate with 'x86- or Pentium-based PCs. Software libraries and drivers enable rapid development of applications for microsecond timing of external events or software transactions under DOS, Windows 95 or NT, and QNX.

THE HARDWARE

The design principle of the hardware is to keep precise time without any intervention from the CPU, bus, operating system, or application software. The timer board has 16 inputlines, each individuallytriggerable by the rising edge of a TTL-compatible pulse.

The hardware guarantees temporal precision by storing a 32-bit time whenever a pulse occurs on one of the input lines. It tracks which line or lines registered a pulse by associating a 16-bit mask with each 32-bit time. The timing is based on an onboard 32-bit counter, and timer resolution is software selectable in decades from 1 µs to 10 ms.

The board registers time without any intervention, but software is necessary to acquire the data from the timer board. For precise timing of many successive events without high-priority software intervention to read times, the timer hardware contains a 4096-event FIFO.

Thus, the software that reads times can, with some knowledge of the average and maximum event rates, check the hardware's status frequently enough to empty the FIFO, but infrequently enough so that system performance isn't compromised.

TIMING EXTERNAL EVENTS

One potential user of the board was trying to understand the source of noise on

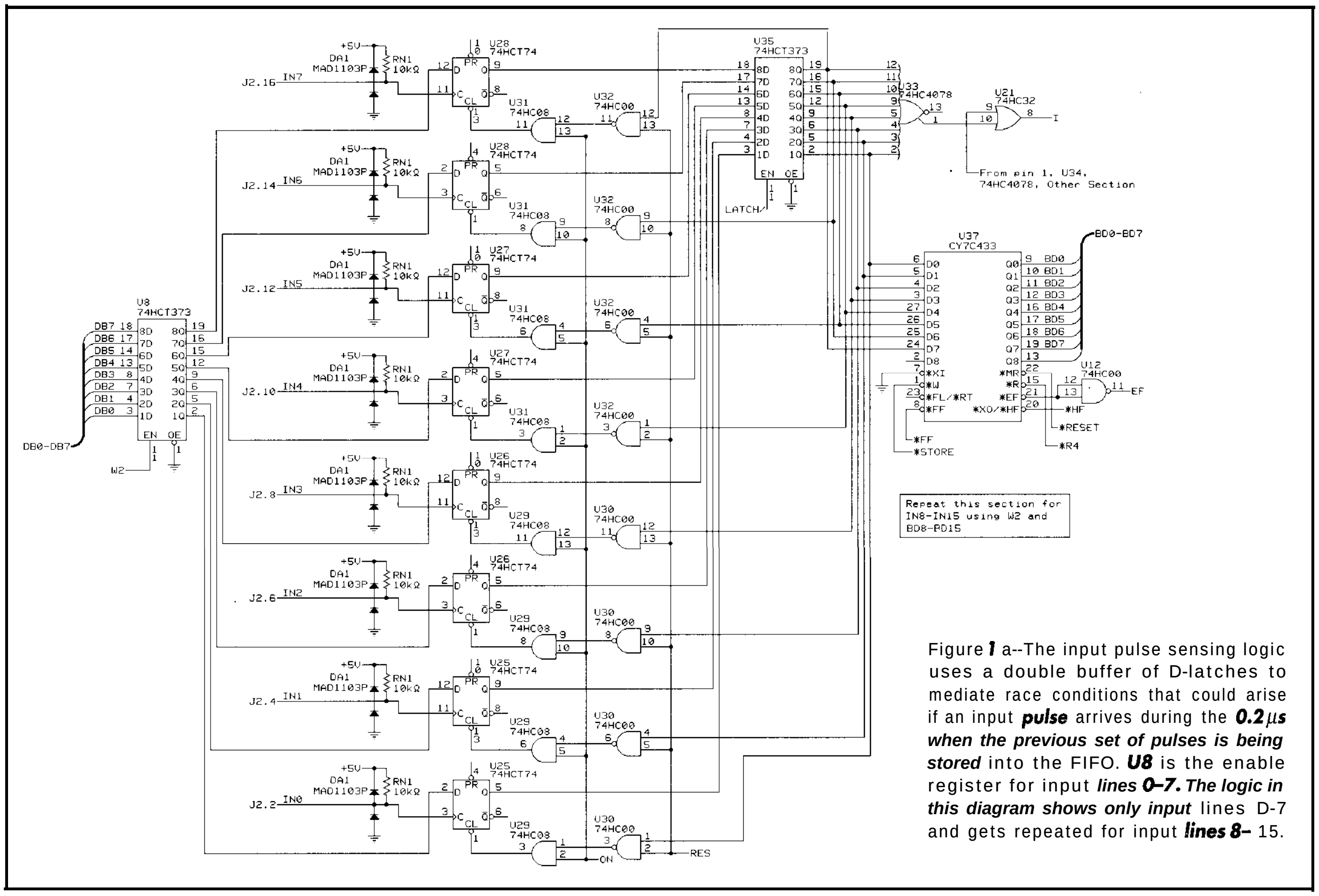


Figure 1 a--The input pulse sensing logic uses a double buffer of D-latches to mediate race conditions that could arise if an input ***pulse*** arrives during the ***0.2 µs when the previous set of pulses is being stored*** into the FIFO. ***U8*** is the enable register for input ***lines 0–7. The logic in this diagram shows only input*** lines D-7 and gets repeated for input ***lines 8–*** 15.

a set of 16 slip rings and wished to assess the reliability of the signals. He wanted to time pulses on 16 lines with microsecond precision for a period of 30 days or longer.

The application code in listing 1 is a thread he could use in Windows NT or 95 to set up the timer board and collect the data. The thread has three phases-timer setup, data collection, and timer shutdown.

Timer setup initializes the softwaredriver, issues a hardware reset to the board, selects the use of library mode 0 (the board-level interface), enables the requisite input lines (in thiscase, all 16), sets the timer resolution (to 1 µs here), and starts the timer.

To run under DOS, no driver is needed. Under Windows NT or 95,1 t_set up () returns a handle to the driver.

Many different application programs can have separate handles to the driver simultaneously, and a separate handle must be created for each thread in a single application. Onlysingleboard installations are currently supported, but this restriction can be relaxed with simple revisions of the software libraries.

Data collection occurs in a w h i 1 e loop that repeatedly sleeps for 1 s, checks the timer board status to determine whether the hardware counter has overflowed in the last second, and then attempts to retrieve events from the FIFO.

In the application outlined here, each iteration of the datacollection loop retrieves up to 100 events every second. The events are returned in twoarrays-unsigned shorts (p_mas k) for the masks and unsigned longs (p_t i me) for the times.

The data-acquisition function, lt_ empty(), returns the number of events that were available on the FIFO, up to the size of the request given as the last parameter of the calling sequence.

If the counter overflows, then a call to 1 t_ovf 1 ( ) returns TRUE. Internally, the overflow indicator is set to FALSE so it won't be TRUE until the counter overflows again.

This approach works because the hardware counter sets the overflow bit in the status register, resets the counter to zero, and keeps counting when it reaches OxFFFFFFFF. The need to count overflows depends on the duration of each timing run and the timing resolution used.

For example, with a timing resolution of 1 µs, the 32-bit counter on the timer board overflows about every 72 min. If overflows are counted in a 16-bit short, then time can be registered with microsecond precision for over 2^{16} h (i.e., 4 years).

Simple but careful programming is needed to determine whether the 32-bit times registered near the time of counter overflow occurred before or after the actual overflow. But, there are no ambiguities as long as the data-collection loop sleeps for only seconds or minutes between attempts to empty the timer board's FIFO.

Timer shutdown isaccomplished by stop ping the timer board, emptying the last events out of the FIFO, resetting the board, and closing the connection to the software driver.

In the **toy application** shown in Listing 1, the program exits the data-collection loop after 5000 events occur and executes the shut-down code sequence.

In a real application, thedatacollection loop may be allowed to continue indefinitely. Then, timer shutdown should occur in a routine that's called when the program is exited.

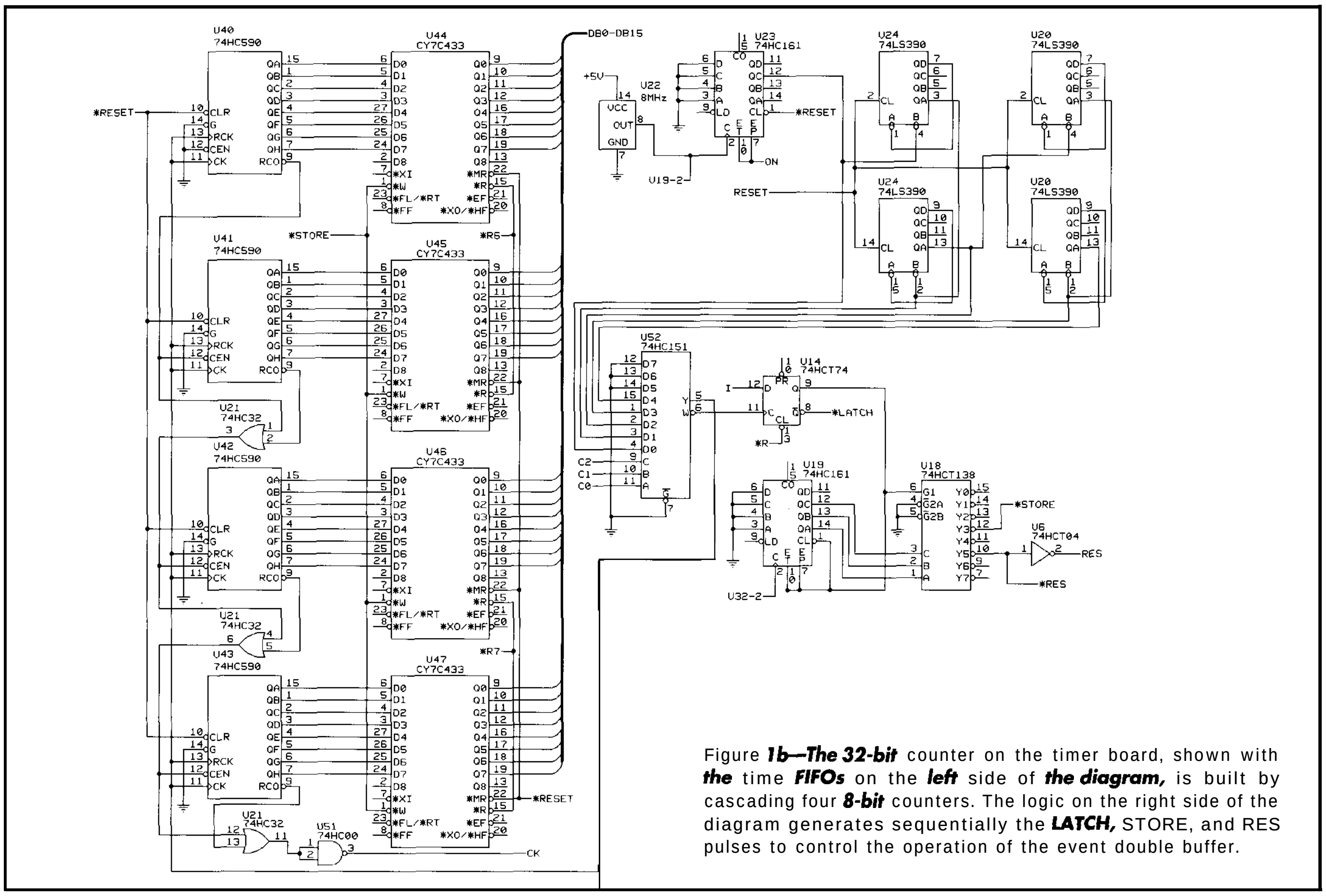
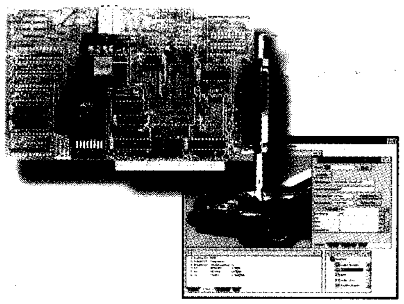


Figure **1b—The 32-bit** counter on the timer board, shown with ***the*** time ***FIFOs*** on the ***left*** side of ***the diagram,*** is built by cascading four ***8-bit*** counters. The logic on the right side of the diagram generates sequentially the ***LATCH,*** STORE, and RES pulses to control the operation of the event double buffer.



Lowest Cost
Data Acquisition

ADAC's new Value-line has uncompromising design features and high quality components at prices below the low cost guys!

Just check out the specs:

Lowest Cost
5500MF $195
8 **channels 12-bit A/D,**
16 digital I/O, **Counter/Timer**

High Speed
5508LC $245
8 channels IP-bit A/D,
100KHz, **DMA**

Multi-Function DMA
5516DMA $295
16 channels 12-bit A/D,
DMA, 16 digital I/O

High Resolution
5500HR $595
16 channels 16-bit A/D,
DMA, 8 digital I/O

learn more:

voice	800-648-6589
fax	617-938-6553
web	**www.adac.com**
email	**info@adac.com**

ADAC
American Data Acquisition Corporation
70 Tower Office Park, Woburn, **MA 01801 USA**

#121

Listing 1—*This* complete thread can be used with existing software libraries to get data from the timer board. *do_a na lys* is*()* is the *user's* function for analyzing and storing the timer data.

```
unsigned short p_mask[10000] = {0};
unsigned long p_utime[10000] = {0};
HANDLE hwinrt_0 = NULL:
long n_ovfl = 0:
long n-events = 0;
void uctest(VOID *pvoid)
{
  int nn;
  n_ovfl = 0;
  n-events = 0;

//Timer setup
  hwinrt_0 = lt_setup(0,0x300,10);   // get handle to driver
  lt_reset(hwinrt_0);       // reset timer board
  lt_mode0(hwinrt_0);       // set mode 0
  lt_ena(hwinrt_0,0xffff);  // enable all input lines
  lt_1m(hwinrt_0);          // set 1-µs resolution
  lt_start(hwinrt_0);       // start

// Collect data
  do {                      // sleep 1 s, ask for data, repeat 5000x
    Sleep((long) 1000);
    if (lt_ovfl(hwinrt_0)) // check for an overflow
      n_ovfl++;             // read up to 100 events from FIFO
    nn = lt_empty(hwinrt_0,p_mask,p_utime,100);
    n-events += nn;
    do_analysis(p_mask,p_utime,nn);
  } while (n-events < 5000);

//Timer shutdown
  lt_stop(hwinrt_0);        // stop board
  nn = lt_empty(hwinrt_0,p_mask,p_utime,5000); // get all events
  n-events += nn:
  do_analysis(p_mask,p_utime,nn);
  lt_reset(hwinrt_0);       // reset timer board
  lt_close(hwinrt_0);       // close driver
  hwinrt_0 = NULL;
  _endthread();
  return;
}
```

TIMING SOFTWARE TRANSACTIONS

Another user wanted to achieve 100-p resolution in timing both the client and server software transactions on a large-scale client-server system for Windows NT.

Figure 2 shows the organization of the hardware and software developed for this application. The components are the timer board, a dynamic linked library (DLL) with shared memory to assign each software client a unique ID number, and the data-acquisition application to retrieve the events.

Although one server and many clients write events to the timer board's output port, only one application reads data. The board and the Windows NT (and 95) drivers work well in this shared mode, but nonsense results if multiple processes read events.

The software timing application takes advantage of 16 lines of TTL output available at a connector on the timer board. To convert software events to hardware triggers, these output lines are connected to the timer input lines by o 40-pin flat-cable loop-backconnector. The state of the output lines is controlled by o 16-bit write-only I/O port within the timer board's address space.

A single call to `lt_pulse(HANDLE hwinrt, unsigned short x)` writes 0, x, 0 to theoutput port, causing brief pulses on the lines specified by the value of x. With the loopback connector in place, this triggers the timer's inputs and registers an event.

The code for the DLL appears in Listing 2. This implementation of the DLL uses the timer board's 16 bits in the following way:

- bits 0 and 1 -dedicated to timing transactions in the server software
- bits 2 and S-command bits to indicate the start and end of the client software
- bits 4-I 5—assign a unique 12-bit identifier for each client

The DLL has two functions. `1 t_Ge t I d ()` returns a unique identification number between 0x0010 and OxFFFO for each client process. Of course, `1 t_Ge t I d ()` deals with overflow by resetting the unique identifier to 0x0010 after it reaches 0xFFF0.

lt_IniId() turns on the DLL so subsequent calls to lt_GetId() return a nonzero ID number. If issued by the server, the call to lt_IniId() keeps the DLL resident as long as the server is resident, ensuring that each client receives a separate unique identifier.

For the client side, the C code is shown in Listing 3a. Listing 3b shows the code for the server.

HEISENBERG INTERFERES

It takes time to measure time. The Heisenberg principle applies when timing software transactions just as it does in other settings.

How much does the timing hardware and software interfere with the process it's trying to time? The board's design guarantees that software doesn't interfere with the registration of events reaching the input lines. Thus, for external events, the timer software doesn't affect the timing precision.

This board's only limitation for timing external events is the +0.01% precision of

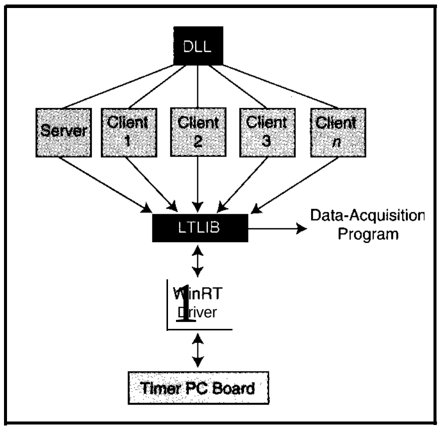


Figure 2-Here are *the* interactions among components far timing ***transactions*** in a *client-server **software** system.*

the 8-MHz crystal providing the time base for the counter. It can cause inaccuracies of 10 µs for each minute the timer is run.

For software timing, some amount of interference is unavoidable. First, it takes a finite amount of time to write to the timer board using `lt_pulse (hwinrt,x)`. I evaluated this time by delivering two successive pulses using the code:

```
lt_pulse(hwinrt,0x1);
lt_pulse(hwinrt,0x2);
```

The duration of one call to `lt_pulse()` can be estimated by computing the difference between the times registered as a consequence of these two function calls. On a 100-MHz Pentium, `lt_pulse()` takes an average of 45 µs under Windows 95 (range 43-l 19) and 88 µs under Windows NT (range 86-448).

For 1250 samples under Windows NT, 1246 of the durations were between 86 and 90 µs, and 4 were much longer. The long-duration intervals represent instances when the application's timeslice ended between the two calls to `lt_pulse()`.

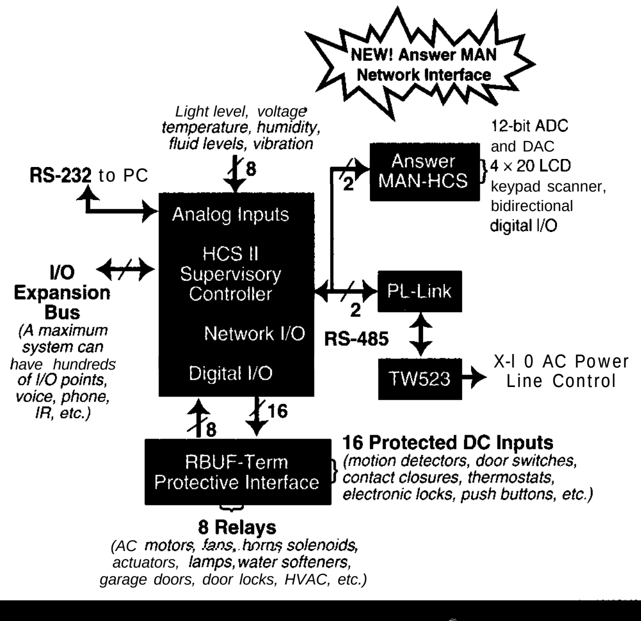
The delay introduced by `lt_pulse()` can be reduced to one bus I/O cycle by redesigning the output logic on the board to be driven from a memory-mapped location rather than an I/O port. Or, the same result can be achieved by driving the timer board's inputs with a digital output board already designed this way.

Second, the data-acquisition program takes finite time and makes all software functions appear to take longer, on average, than they **really** do. I used the software timing strategy outlined in this article to measure how long it takes to drain 100

CIRCUIT CELLAR HCS II

Affordable Home Automation and Security

Our concept of building a home control system is based on the principles embodied in building an original Heathkit. Besides giving you a project which details everything from the theory of operation to instructions for fabricating a functioning system, we give you the sense of confidence that comes from success.


NEW! Answer MAN Network Interface
Light level, voltage temperature, humidity, fluid levels, vibration
8
RS-232 to PC
Analog Inputs
HCS II Supervisory Controller
Network I/O
Digital I/O
I/O Expansion Bus
(A maximum system can have hundreds of I/O points, voice, phone, IR, etc.)
2
Answer MAN-HCS
12-bit ADC and DAC
4 x 20 LCD
keypad scanner, bidirectional digital I/O
PL-Link
2
RS-485
TW523
X-10 AC Power Line Control
8
16
RBUF-Term Protective Interface
16 Protected DC Inputs
(motion detectors, door switches, contact closures, thermostats, electronic locks, push buttons, etc.)
8 Relays
(AC motors, fans, horns, solenoids, actuators, lamps, water softeners, garage doors, door locks, HVAC, etc.)

The HCS II is designed with an economical central core and an expandable network. A basic system consists of the HCS II central processor, which incorporates direct analog and digital I/O with real-time event triggering; a PL-Link wireless X-10 AC power-line interface which has sixteen contact-closure/voltage inputs and eight 3-A AC/DC relay outputs for solenoids, motors, lamps, alarm horns, and so forth.

The house event control sequence is written on a PC in a unique, user-friendly control language called XPRESS and stored on the HCS in nonvolatile memory. Building them is as easy as 1,2,3.

Of course, we have lots of extra stuff like our new Answer MAN network module when you want to expand the basic system. The full HCS has phone and modem interfaces, infrared remote control, voice synthesizers, LCDs, and much more.

A Circuit Cellar HCS II is not for passive people and definitely not for the masses. It is for a special technical breed that wants a challenge with a tangible reward. Discover the world of manageable and affordable home automation and security.

Circuit Cellar, Inc.

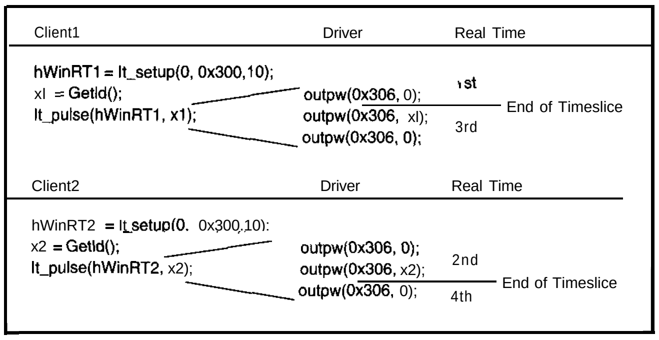
Park St., Vernon, CT 06066 (860) 875-2751

• www.circuitcellar.com

events from the timer board's FIFO on my 100-MHz Pentium.

It takes an average of 7.4 ms under Windows 95 and 16.2 ms under Windows NT 4.0. Thus, if the mean rate of events is 100 per second (8,640,000 per day) and the thread is sleeping for 1 s between event retrievals, then this thread uses less than 1.6% of the total time availableon thesystem.

These times can be improved by modifying the source code for 1 t_empty(). All these timing numberswould be proportionately faster on faster processors.



```
Client1                                   Driver                 Real Time
hWinRT1 = lt_setup(0, 0x300,10);                                  ıst
xl  = GetId();                            outpw(0x306, 0);        ---------- End of Timeslice
lt_pulse(hWinRT1, x1);                    outpw(0x306,  xl);      3rd
                                          outpw(0x306, 0);

Client2                                   Driver                 Real Time
hWinRT2  = lt_setup(0.  0x300,.10):
x2  = GetId();                            outpw(0x306, 0);        2nd
lt_pulse(hWinRT2, x2);                    outpw(0x306, x2);       ---------- End of Timeslice
                                          outpw(0x306, 0);        4th
```

Figure ***3—This*** figure shows how multiprocessing operating ***systems*** can cause ***software*** race conditions. From ***left*** to right, the three columns show the sequence of software timing calls from the point of view of two concurrently running clients, the *ou* ***tpw*** calls from the software libraries, and the difficulties that result in real time.

MULTIPROCESSING OSs

The software timing applications outlined here are designed to create the least possible interference with the transactions they're trying to time. To do this, I've adopted a slightly quick-and-dirty method to write pulses to the board.

1t_pulse(hWinRT, x) calls out pw() three times in rapid succession to write 0, x, and 0. This setup provides the software equivalent of race conditions if one client's timeslice ends and another's starts during the sequence of calls to outpw().

Figure 3 outlines the critical situation. Suppose that clients 1 and 2 are both active and that they've acquired different identification numbers [e.g., xl and x2).

Also suppose the timeslices dictated by the OS contrive to interrupt client 1 in 1 t_ pu **1 se** () before xl is written to the output port and then interrupts client 2 and reinstates client 1 after x2 is written. The driver handles this fine, but the sequence of writes to the output port is 0, 0, x2, xl, 0, 0.

If x1 sets any bits that were already set by x2, then the outcome is incorrect. Because the input lines on the timer board are triggered by positive edges rather than by levels, the event signaled by xl will be registered with the wrong mask.

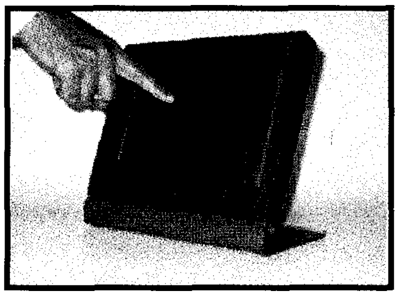
The bad luckoccurs, albeitwith verylowprobability, becausedata written to theoutput port is latched by the output drivers. Software creates pulses through three consecutive writes to the oufput port.

The buffer circuit in Figure 4 can be put between the output and input connectors of the timer board to convert a single write to the output port into pulses on the active lines.

This circuit takes advantage of a brief data available pulse (DAV) placed on one pin of the output connector each time anything is written to the output port. The data-ready pulse is delivered even if writing to the output port doesn't change the voltage on any of the output lines.

Writing x2 and xl (or even xl and xl) sequentiallycausespulsestoappearon the

Touch The Future



LCD Touch Monitors
LCD Touch Screens
VGA LCD Displays
LCD Controllers
*ISA, **PC104,** Analog, Video*

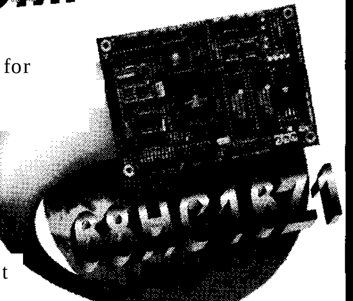
EARTH Computer Technologies   Lowest Prices on Earth!

1110 Calle Cordillera - SanClemente - CA - 92673
Ph: 714-361-2333 - Fax: 714-361-2121
Email: oemsales@flat-panel.com
FREE CATALOG available at http://www.flat-panel.com

#122

$249.95 QTY. 1

SINGLE BOARD COMPUTER



This board comes ready for demanding applications, with 16 bit DSP capability, Analog & Digital I/O, and **68HC** 11 Compatibility. Backdoor Debug and up to one Megabyte of battery backed RAM creates a robust development environment.

* 16 Programmable I/O lines
* 8 High Drive Digital Outputs
* 8 Channels of Fast 10 bit A/D
* Optional 8 Channel, 8 bit D/A
* Up to 2 RS232/485 Serial Ports
* Backlit capable LCD Interface
* Optional 16 Key, Keypad & Interface
* 16 bit Timer/Counters with PWM & Clock/Calendar
* 64K RAM & EEPROM included plus Flash Capability

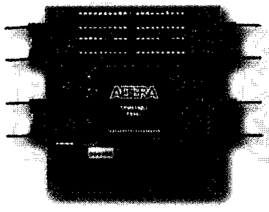
1985 - 1997 OVER 12 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc.
618-529-4525 Fax **457-0110** BBS **529-5708**
11 EMAC WAY, CARBONDALE, IL 62901
WORLD WIDE WEB: **http://www.emacinc.com**

#123

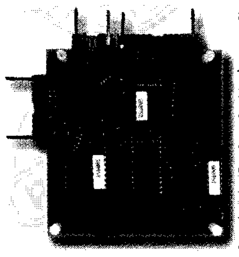
The PC/I 04 Motion Control Experts

Need motion control within your PC/104 application? Overwhelmed by the number of products & vendors out there? Looking for a motion control specialist instead of just another PC1104 vendor with one motion controller?



Model 5912
Encoder Interface

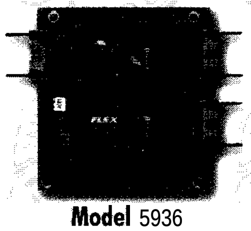
Look no further. With more than a decade of experience supporting the motion control needs of OEM customers, Tech 80's family of DC1104 modules can meet the encoder interfacing and servo & stepper control demands of your embedded application.



Model 5928
Servo Controller

All Tech 80 products feature extensive C-compatible software libraries, example code and tools to provide you with a flexible development environment.

AND if your needs extend beyond the PC/104 realm, Tech 80 has the industry's most extensive line of board-level motion control products for IP, PC, STD and VME-based systems.



Model 5936
Stepper Controller

For more information on our PC/104 motion control products or to speak with an sales engineer regarding your current project, please contact us at 800/545-2980 or visit us at:

www.tech80.com/cc/pc104.html

The People in Control of Motion™

Minneapolis, Minnesota **USA**
800/545-2980 • 612/542-9545 • 612/542-9785 (fax)
www.tech80.com • info@tech80.com

#124

listing 2-This complete dynamic link library **(DLL)** assigns a unique identification number between Ox **10** and **OxFFFO to** each software client that **calls** `lt_GetId()`.

```
// LTDLL.H header file
#define EXPORT _declspec (dllexport)
extern EXPORT unsigned short CALLBACK lt_GetId (VOID);
extern EXPORT void CALLBACK lt_IniId (VOID);

// LTDLL.C
#include <windows.h>
#include <string.h>
#include "ltdll.h"
#pragma data_seg ("shared")
unsigned short lt_nindx = 0x10;       // next index to assign
unsigned short lt_ison = 0:           // is the DLL activated?
#pragma data_seg ()

// Generic DLL main routine-does nothing
int WINAPI DllMain(HINSTANCE hInstance, DWORD fdwReason,
    PVOID pvReserved){
  switch (fdwReason){
    case DLL_PROCESS_ATTACH:
    case DLL-THREAD-ATTACH :
    case DLL-THREAD-DETACH :
    case DLL-PROCESS-DETACH :
      break ;}
  return TRUE ;}

// Assign a unique ID number
EXPORT unsigned short CALLBACK lt_GetId 0 {
  unsigned short rvalue;
  if (!lt_ison)
    return(0);
  rvalue = lt_nindx;
  lt_nindx += 0x10;
  if (lt_nindx >= 0xfff0)
    lt_nindx = 0x010:
  return(rvalue); }

// Reset ID numbers
EXPORT void CALLBACK lt_IniId(){
  lt_ison = 1;}
```

correct output lines for both writes. The correct input lines are pulsed, and the correct masks are recorded by the timer board.

If the buffer circuit is used, there's no need to create pulses with three calls to out pw () A single call suffices, the software race condition never occurs, and (as a pleasant side effect) the amount of time taken by the sleeker `lt_pulse()` is reduced to 22 µs under Windows 95 and to 76 µs under NT.

DRIVER IMPLEMENTATION

I implemented the Windows NT and 95 drivers using the WinRT toolkit from BlueWater Systems. It's extremely easy to use and the exact same API is presented for both operating systems.

listing **3a—Each** client program obtains a unique ID and issues start and end pulses that are the **ORed** Dwi **0x4** or 0x8. b-The server turns on **the** ID issuer and generates start and end pulses for each transaction on lines Ox I and 0x2.

```
a) client_mainO {
     HANDLE hwinrt = setup(0,0x300,10);
     unsigned short id = lt_GetIdO:
     lt_pulse(hwinrt, id | 0x4);
     // client code goes here
     lt_pulse(hwinrt, id |0x8);}

b) server_mainO {
     lt_IniId();
     HANDLE hwinrt = setup(0,0x300,10);
     lt_pulse(hwinrt,0x1);
     // server code goes here
     lt_pulse(hwinrt,0x2); }
```

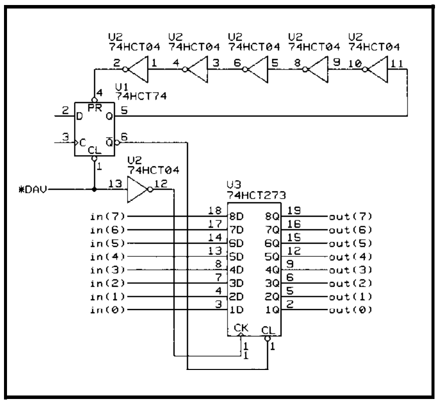


Figure *4—This* buffer circuit can be inserted between *the* output and input to ***the*** *timer* board to convert *its* latched high or low outputs *into pulses on each line that is* high. Only bits ***0–7*** are shown.

Without knowing much about either Windows OS, **I was able to** port my DOS library to a workable NT and 95 system in about ***10*** hours. The library developed on Windows 95 worked the first time I tried it in NT. The toolkit is relatively inexpensive, and if you want to expand on the capabilities of my API, you can do so easily.

The WinRT toolkit isn't intended for high rates of data acquisition because its generic driver interprets commands from the application program. Thus, the relatively long times needed to retrieve 100 events from the timer board can be attributed to the driver I used rather than to any slowness caused by the timer hardware.

It's possible to improve substantially on the time needed to retrieve 100 events by making a specialized driver using a DDK.

PROJECT WRAPUP

The timer board provides microsecond precision in timing the intervals between external events. But, with minor software and hardware modifications, it also times software transactions.

My methods are an excellent compromise between the ideal (no Heisenberg) and the real, asynchronous world of modern computing. Near perfection can be attained through feasible improvements I mentioned in the text. EPC

Special thanks to Dave Hermeyer for assistance in designing and manufacturing the timing board, Paul Lever of Blue Water Systems for hints on timing software transactions, and Adam Bernstein and Paul Lever for their helpful comments.

Steve Lisberger is a neuroscientist at the University of California, San Francisco. He also owns lisberger Technologies, a small business dedicated to innovative* solutions *for the problems of real-time data *acquisition* **and analysis. You may reach Steve at** *sgl@listech.com.*

SOFTWARE RELEASE
Completeschematicsforthetiming board are available **in** 1 isberger **.pdf andmaybedownloadedfromthe Circuit Cellar Web site.**

REFERENCES
C. Petzold, **Programming Windows 95, Microsoft Press, Redmond, WA, 1996.**

SOURCES
Timer-board hardware and libraries
Lisberger Technologies
848 Clayton St.
San Francisco, CA 941 17
(415) 661.2097
www.listech.com

WinRT toolkit
BlueWater **Systems, Inc.**
P.O. Box 776
Edmonds, WA 98020
(425) 771.3601
Fax: (425) 771-2742
www.bluewatersystems.com

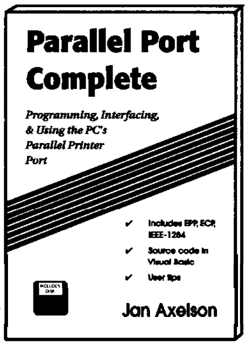
I R S
416 **Very Useful**
417 Moderately Useful
418 Not Useful

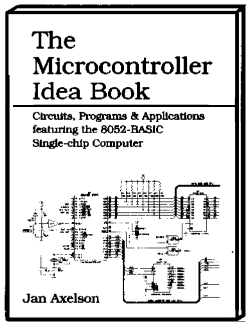
JUMP START YOUR DESIGNS with these practical guides

NEW! "A focused book that delivers what it promises: detailed technical information on the parallel port." - *Windows Developer's Journal*

"It's been a while since I've seen a book as practical as this one." *-Nuts & Volts*

ISBN 0-9650819-l-5
$39.95
343 pages
Includes disk


Parallel Port Complete
Programming, Interfacing, & Using the PC's Parallel Printer Port
✔ Includes EPP, ECP, IEEE-1284
✔ Source code in Visual Basic
✔ User tips
Jan Axelson


The Microcontroller Idea Book
Circuits, Programs & Applications featuring the 8052-BASIC Single-chip Computer
Jan Axelson

MICROCONTROLLERS!
"An ideal introduction to low-end embedded design." - *EDN*
"The writing is a model of clarity and conciseness."- *PC Magazine*
ISBN O-96.50819-0-7
273 pages. $31.95

Order li ne: l-800-247-6553

For more information or international orders:
Lakeview Research Phone 608-241-5824
2209 Winnebago St. Fax 608-241-5848
Madison, WI 53704 Email jaxelson@lvr.com

Shipping: $5.00 (U.S. & Canada)

Read sample chapters on the web:
http://www.lvr.com

#125

...erything you'd expect ... at an unexpected ...
...hand held PC you can afford to include in your ... Our PC-Lite series of hand held, MS-DOS ... featuring four affordable un...
...available w... memory
Microsoft Corporation
*MSDOS not included
MEMORY CARD
...omputers for OEM applications

#126

Dennis Liles
& Rick Lehrbaum

PC/104 Steps old of the Box

There are times when *the* feenie-weenie *PC/1* 04 *form* **factor just doesn't cut if. Why? Because if fakes** 7-2 years **for** *the* **latest desktop** *CPUs to* **shrink enough** *to fit,* **and sometimes you need more high-end functions on a board.** *Enter* EBX.

With all the hype over Pentium Pro, PowerPC, Java, and the Internet, an article on a new single-board computer form-factor standard might seem mundane. After all, only a few hundred engineers **design** SBCs. But, if you're one of the few who specify or design embedded systems, read on.

If **your** specs call for a high-performance, highly functional embedded computer that fits within a physicallychallenged environment, the new EBX standard may be what you need.

ANOTHER STANDARD?

No doubt you're wondering, "Don't we have enough standards?" After all, there's STD, Multibus, VME bus, G64, passive-backplane PC, PC/ 104, PC/ 104-*Plus,* CompactPCI, and probably a few we've forgotten to list.

But remember, embedded systems come in all shapes and sizes. Back when all embedded computers were based on proprietary designs using a microprocessor or single-chip microcontroller, you simply designed your electronics to fit the requirement.

But as 100+ MHz CPUs, DRAM over 64 MB, high-resolution graphics, TCP/IP networking, and highcapacity data storage have become the norm, today's designers are using off-the-shelf embedded computer modules wherever possible. This way, they can put more attention on software, packaging, and specialty interfaces.

But when it comes to off-the-shelf SBCs, one size can't possibly fit all applications. That's why there are so many form factors.

And, it isn't just size. There's also the physical arrangement of boards to think about. Early off-the-shelf embedded computer solutions were based on a variety **of** backplane buses (e.g., STD, Multibus, VME, and G64). Even the PC/AT architecture was adapted to a backplane bus (i.e., the passive-backplane PC).

These multiple-board solutions require a "passive" backplane with several boards plugged in perpendicularly, supported by a

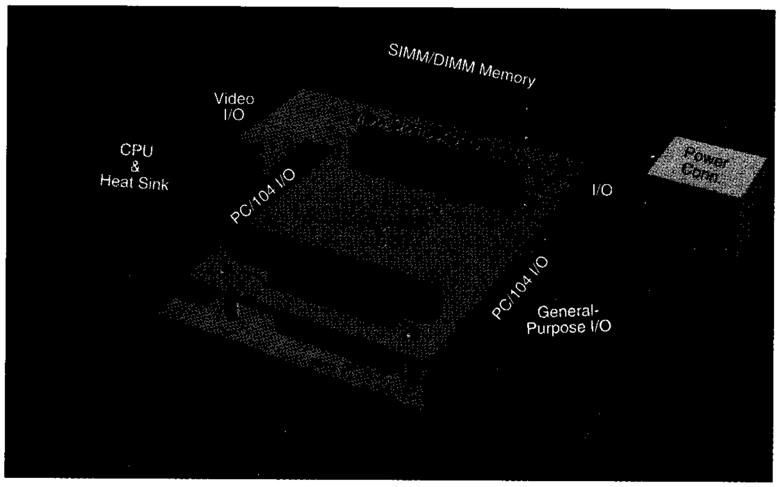


Figure ***1—Look*** *familiar? The new EBX standard evolved from* ***the*** *venerable* ***Ampro*** *Little Board form factor.*

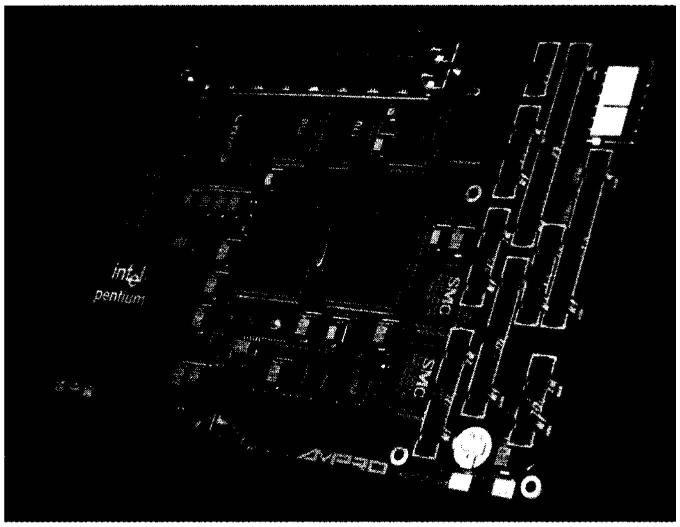


Photo 1—***The Ampro Little Board/P5i*** takes advantage of the EBX tall CPU option to offer an Intel Pentium processor along with lots of ***onboard I/O*** in the ***EBX-compliant SBC.***

metal card cage to hold it all in place. The result is a rather large and complex multiple-board configuration, which often cannot fit within the space-constrained environments of embedded applications.

That's not to say that backplane buses aren't great for lots of nondesktop applications, especially when system specs call for half a dozen or more easily swappable function modules (e.g., rack-mounted industrial-control computers and telecommunications switching systems).

But, backplane configurations usually can't fit the kind of applications where embedded microcontrollers were traditionally used. They just don't fit the size and other mechanical constraints of applications like portable test equipment, medical diagnostic instruments, operator interface displays, mobile control and monitoring systems, and many other embedded control applications that come to mind.

What about PC/104 and PC/I 04-*Plus*? After all, these popular embedded-PC building-block standards were designed to fill the need for a modular off-the-shelf approach to deeply embedded system design, without the mechanical limitationsof backplanes and card cages.

Sure, PC/I 04 modules help you avoid reinventing the wheel for embedded PC systems. But, even PC/ 104 isn't a perfect fit for every application.

Why? For one thing, when the latest desktop CPUs are first introduced, they don't come in small enough packages to reliably fit on PC/I 04 CPU modules. It usually takes l-2 years for them to shrink enough to fit the compact PC/I 04 form factor.

Another problem is that the compact PC/104 form factor naturally limits how many high-end PC functions you can fit on a single module along with all the required I/O connectors. These are the reasons why Ampro, the inventor of PC/I 04, continues to support its little Board SBC form factor.

Ampro's Little Board has been around for over 14 years. Originally patterned after the footprint of the 5.25" disk drive, this form factor has been used in thousands of applications and continues to be quite popular for new designs.

Over the years, this form factor has been used by dozens of embedded computer suppliers and has hosted lots of different CPUs, including the 280, 64180, 68k, 80186, '286, '386SX,'386DX,'486SLC, '486DX, Pentium, and now Motorola's PowerPC-which brings us to EBX.

MOT'S MOTIVATION

Motorola has long advocated open architectures and industry standards. They were one of the originators of the VME-bus specification and continue to support new and emerging standards (e.g., CompactPCI).

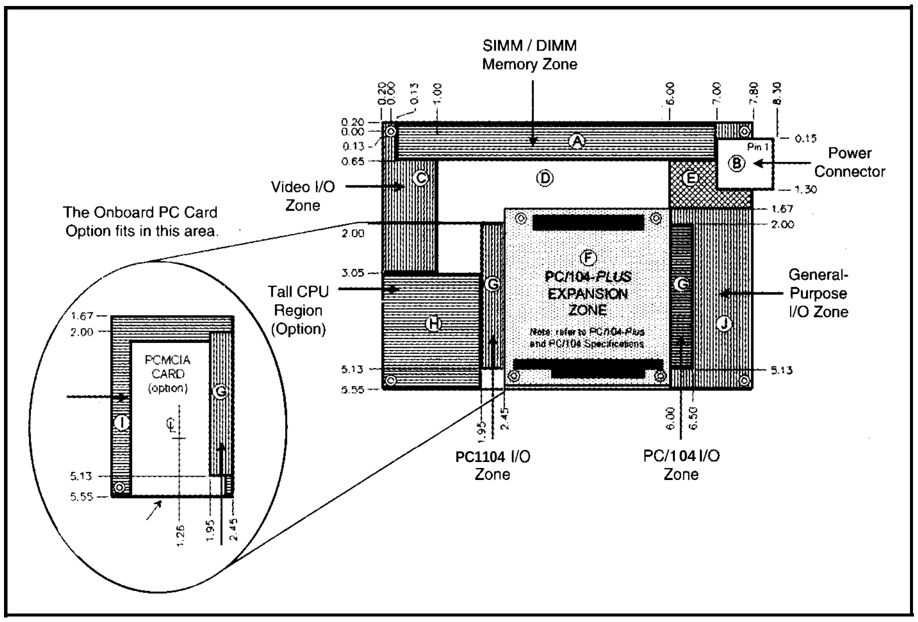
In August 1996, Motorola started work on a new MBX family of SBCs based on the Motorola MPC82 1- and MPC860-series PowerPC microprocessors, which are highly integrated devices with an on-chip integer and communications processor.

A key design goal of MBX was to be physically smaller than conventional PC motherboards, yet offer higher price/performance and functionality. It had to be small enough for deeply embedded applications but large enough to contain all the embeddedcomputerfunctionalityrequired for a specific task or application.

Motorola was already manufacturing desktop PC/AT-style motherboards based on the PowerPC processor in the ATX desktop motherboard form factor. But embedded applications need something smaller and more rugged.

It quickly became apparent that no industry standard met Motorola's needs. The closest match was the form factor implemented on Ampro's Little Board and by several other SBC companies.

At about the same time Motorola's form-factor search was taking place, Ampro was getting ready to announce a new high-



end SBC-the Pentium-based Little Board/P5*i* with PCI-enhanced PC/I 04. Unbeknownst to Motorola, the Little Board/P5*i* was about to add PCI to PC/I 04 and, hence, to the Little Board form factor. PCI on the Little Board form factor made it even more interesting to Motorola, since PCI was to be a key feature of the MBX product family.

The companies agreed to convert the Ampro Little Board form factor into an open, multivendor, and architecturally neutral standard-the EBX (embedded board, expandable) form-factor specification depicted in Figure 1.

THE END PRODUCT

The EBX form factor is small enough for deeply embedded applications, yet large enough to offer the functions of a full embedded computer system-CPU, memory, mass storage interfaces, display controller, network interface, serial/parallel ports, and other system functions.

EBX isarchitecturallyneutral, so itdoesn't specify that the architecture must be PC compatible—or even that the CPU be 'x86 compatible. As a result, the first two EBX-compliant products, the Pentium-based Little Board/P5*i* from Ampro (Photo 1) and Motorola's PowerPC-based MBX SBC (Photo 2), are different architecturally.

EBX SPEC

As you see in Figure 2, the key features of EBX are its:

- rectangular (5.75" x 8.00") form factor
- PC/I 04-*Plus* expansion stack location
- eight mounting-hole pattern
- seven-pin power connector
- recommended areas for I/O connections, DRAM, CPU, and optional PC Card slots

The EBX spec provides two drawings that define regions for various types of I/O, SIMM or DIMM memory, and one of two basic configurations (i.e., the tall-CPU and PC Card-slot options).

The tall-CPU option is intended for large-package CPUs (e.g., Intel's Pentium), which require heat-sink assemblies. The PC Card-slot option specifies a recommended location for onboard PC Card slots (see Figure 2), assuming a smaller, lower profile CPU can fit in the PC/I 04-*Plus* expansion location beneath any plugged-in modules.

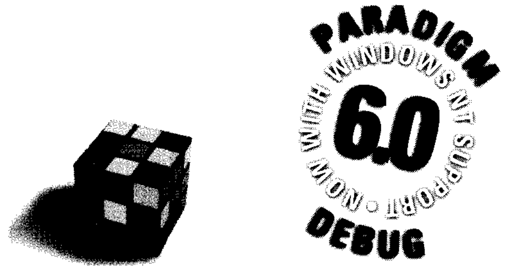
The PC/104-*Plus* expansion stack is one of the most important aspects of EBX. PC/I 04-*Plus* basically implements the same electrical bus as desktop ISA and PCI but in the form of rugged and reliable pin-and-socket connectors instead of the desktop motherboard's edge-card connectors. This setup results in a modular, compact, and highly rugged assembly.

With approximately 40 in? of real estate, EBX-compliant SBCs have enough space to implement a high level of performance and functionality on a single card. Still, it fits within tight space budgets.

EBX's modular expandability lets you easily adapt an EBX SBC to match the needs of your project. Further flexibility comes from the onboard PCMCIA sockets (or a plug-in PC/I O&based PCMCIA interface with the tall-CPU option), which offer access to PC Cards from the mobile and hand-held computing markets.

USING EBX

The dimensions of EBX (5.75" x 8.00") are about the same as those of many VGA-resolution LCD panels. EBX SBCs are there-



Par•a•digm: your source for the most, high-powered, comprehensive set of time-saving software and hardware development tools for embedded application development.

1: Paradigm LOCATE the most popular tool for creating embedded C/C++ applications with Borland and Microsoft compilers; 2: Paradigm DEBUG the only x86 debugger with C++, RTOS, scripting language, and full in-circuit emulator support; 3: Paradigm SUPPORT the best technical support in the industry supplied to our customers for free.

Developing real-time embedded applications doesn't have to be time consuming or difficult-you just need to have the right tools. Paradigm alone has the high performance development tools you need to streamline the embedded system software development process so your Intel and AMD x86 applications are ready in record time. Paradigm's complete suite of tools work with industry standard C/C++ compilers from Borland and Microsoft, as well as hardware development tools from Applied Microsystems, Beacon Development Tools and other popular in-circuit emulator vendors.

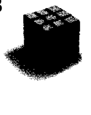
Call us at 800-537-5043 today and let us take care of all your development tool needs. so you can keep your focus where you need it-on your application.

Paradigm

Paradigm Systems
3301 country Club Road
Suite 2214
Endwell, NY 13760

1-800-537-5043

Phone **607-748-5966**
Fax **607-748-5968**
info@devtools.com
http://www.devtools.com



#128

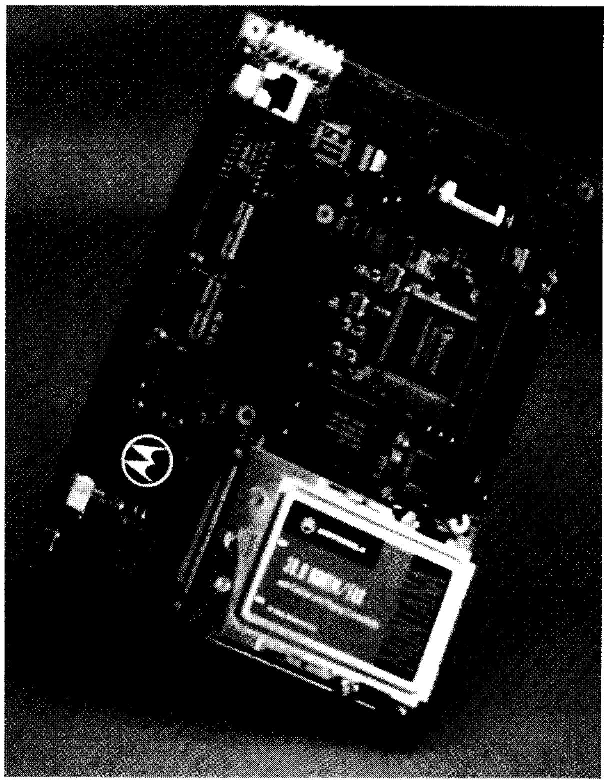


Photo 2—Motorola's MBX860, based on the PowerPC 800 series microprocessors, is the first EBX-compliant SBC to implement the EBX onboard PC Card-slot option.

fore well-suited as the brains behind flat-panel operator interfaces. With its modular PC/l 04-*Plus* expansion, EBX makes a great basis for a panel-PC SBC.

A common question is, "On average, how many PC/l 04 modules are plugged into the EBX SBCs in typical embedded applications?" The answer: one.

Due to the uniqueness of most embedded systems, such applications require at least some custom electronics. Often, system designers create a custom application board that contains all required functions and interfaces not provided directly on the EBX SBC.

This board plugs into the header connectors of the PC/l 04-*Plus* bus, resulting in a two-board sandwich consisting of the EBX SBC and the application board. Depending on bandwidth and other requirements, the application board may interface to the EBX SBC using signals of the ISA bus only, the PCI bus only, or both.

As for size, the application board can be PC/l 04 or EBX form factor, but it really doesn't need to conform to any particular form-factor standard. Generally, the application board takes on whatever shape works best for the application.

Functions included on typical application boards include:

- specialized analog and digital I/O
- sensor or actuator interfaces
- interfaces to application-specific buses (e.g., CAN, Profibus, etc.)
- power supplies or power-conditioning circuits
- filtering, signal conditioning, or connector adaptation for the EBX SBC's onboard I/O
- custom keypad, display, or touchpad interfaces

Another point to consider is that the EBX standard includes an option for the PC/ 104-*Plus* bus to be built with stackthrough connectors, as on a normal (self-stacking) PC/l 04-*Plus* form-factor module. This op tion lets you plug an EBX SBC onto an application baseboard, instead oftheother way around.

By the way, it's interesting to note that with stackthrough bus connectors, an EBX SBC is a lot like a giant PC/104-*Plus* module! You can, for example, unplug a stackthrough-bus EBX SBC and plug in a PC/104 form-factor SBC in its place. In other words, either a stackthrough-bus EBX SBC or a PC/l 04 form-factor SBC can plug into the same PC/l 04-*Plus* socket.

EBX FUTURE

The EBX specification has been structured in a manner that gives board designers maximum flexibility and creativity and yet stays within the confines of a standard board specification. With backing from Amproand Motorola, the specification will remain architecturally neutral, open, and royalty free.

It provides an alternative to backplane-based solutions for high-performance, high-integration embedded computer applications and can thereby save you from having to create your own proprietary solution.

Already, hundreds of off-the-shelf PC/l 04 form-factor expansion modules can be used with EBX SBCs. And of course, you can expect dozens of EBX SBC suppliers to offer a broad range of CPUs and onboard functions.

Several chassis and enclosure makers are introducing EBX-compatible packaging products to simplify the design of EBX-based systems. Best of all, the EBX standard is open tocontinued technology advancements, since it is processor and payload independent.

One last comment: EBX isn't the first time Intel and Motorola processors have shared a common form-factor standard for non-desktop applications. Another important standard-VME bus-turned into a decades-long $1 **B+** industry. So, keep an eye on the emerging EBX market! PCQ.EPC

***Special* thanks to Daniel lehrbaum for creating the *3D color* rendering of the EBX form factor (Figure *1*).**

Dennis *Liles* is a seniorproductmarketer for Motorola Computer Group. His primary responsibilities are managing *the MC68000*-based VME *module and embedded motherboard product lines. You may reach him at* *dennis_liles@phx.mcd.mot.com*.

***Rick lehrbaum* *cofounded Ampro* *Computers where he served as VP of engineering from 1983 to 1991. Now, in addition to his duties as* VP of strategic development, Rick chairs the PC/ *104* Consortium. He may be reached at *rlehrbaum@ampro.com*.**

SOURCES

Little **Board/P5i**, EBX **spec**
Ampro Computers, Inc.
4757 Hellyer Ave.
Son Jose, CA 95 138
(408) 360.0200
Fax: (408) 360.0220
www.ampro.com

MBX SBC, EBX **spec**
Motorola Computer Group
2900 S. Diablo Way
Tempe, AZ 85282
(512) 434.1526
www.mot.com/computer

IRS

4 19 **Very** Useful
420 Moderately Useful
42 1 **Not** Useful

Fred Eady

Managfng a NASA Plant-Growth Chamber

Part 2: Coordinating Sensors and Analyzing Data

Fred offer a *traditional data-collection technique? No way. Instead, he's taking the hardware he selected last month, starting at the sensor end, tracing the datastream to the SBC301 flash file,* *and serving it to a* **Web browser.**

Last time, I described the hardware we'll use to collect our plant-growth chamber (PGC) data (INK 86). So, right now, you're expecting another run-of-themill data-collection article like the hundreds you've read in other magazines.

Not from me, buddy! I'm going to start at the sensor, follow the serial datastream to the SBC 301 flash file, and then "serve" it out to a waiting Web-browser screen.

Yep, you read it right: "serve" it out. You know what that means, so let's light the fuses on those solid boosters and put our PGC in orbit.

"PIC"ING UP THE DATA

If you recall, the PIC14000 was specified as the sensor interface because its characteristics are well-suited for the PGC data-gathering application. The physical interfaces are shown in Figure 1.

Under the covers, the PIC will fire off A/D conversions that import data from the humidity, temperature, and carbon-dioxide sensor array. Once the sensor data is realized, the PIC is responsible for converting the floating-point data into a numeric format that can be stored in the embedded platform's flash. A simple serial connection is the conduit to transfer the converted sensor data from the PIC14000's internals to the SBC 301's flash disk.

I won't go into the inner workings of the PIC 14000. That's an article in itself. Matter of fact, I've already covered it ("Take Your PIC-A took at the PIC 16Cxx Family," *INK* 65). Meanwhile, Figure 2 represents the programmatical flow taking place within the PIC's firmware.

MEASURING PGC TEMPERATURE

The temperature is taken by measuring the voltage across a voltage-divider network and calculating the resistance of the platinum RTD. To reduce RTD self-heating and improve measurement accuracy, the current through the RTD should be in the vicinity of 0.5 mA.

Ideally, at 0°C, our voltage divider should have a total value of 10 kΩ. Under ideal conditions, this should produce a voltage of 0.50 V across the RTD to ground.

Basic Ohm's law says that if the RTD changes resistance, which it will as its surrounding temperature fluctuates, the to tal voltage divider resistance will change and affect the current flowing through the resistors and RTD.

Since the PIC is not really measuring resistance, I have to derive the RTD resistance using a voltage measured across a known resistance in the divider. Once the voltage across the known resistor is taken, I can then calculate the current flowing through the voltage divider as a whole.

At that point, I have a known current value that enables me to calculate the new total resistance value of the voltage divider. Since two of the resistances are constants, it's then a simple matter of first-grade arithmetic to resolve the RTD resistance.

The RTD resistance is plugged into the resistance-to-temperature function and a resultant temperature is derived:

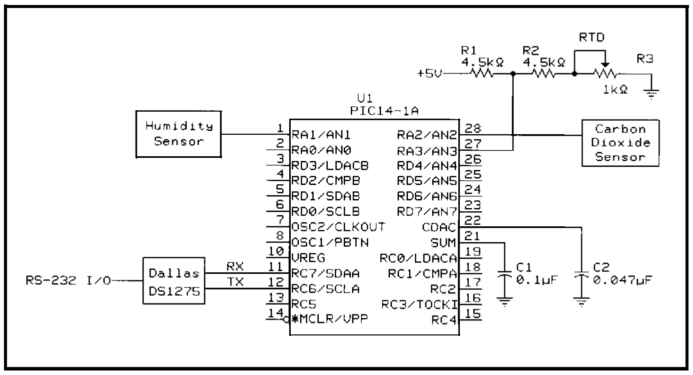


Figure 1—Ideally, the ***voltage*** *divider allows exactly 0.5* ***mA to*** *flow* ***at*** *0°C. The* ***DS1275*** ***is an power-stingy*** *1-pin RS-232 interface IC.*

$$\text{True RH} = \frac{\text{Sensor RH}}{1.0546 - 0.00216 \times T}$$

for T in degrees Celsius, or

$$\text{True RH} = \frac{\text{Sensor RH}}{1.093 - 0.0012 \times T}$$

for T in degrees Fahrenheit.

Another plus to measuring the higher voltage point is that I don't have to preload the A/D input of the PIC14000. If I measured across the RTD itself, the input voltage would be too close to the minimum voltage that the PIC A/D input circuitry needs to take a meaningful reading.

What I mean by preload is to program the PIC input to add about 0.45 V of offset to the input voltage. Thus instead of measuring 0.5 V, the input actually sees 0.95 V. Of course, the offset is compensated for in the PIC measurement software.

MEASURING PGC HUMIDITY

Humidity is a bit simpler to measure than temperature. The work has already been done.

The HyCal humidity sensor is equipped with integrated signal conditioning. I simply attach the output of the humidity sensor to another of the PIC's A/D input pins and read the sensor's output voltage.

My selected humidity sensor is capable of supplying a voltage that varies from 0.8 V at 0% RH to 3.9 V at 100% RH. The particular sensor used in this application provides 0.807 V at 0% RH and 3.015 V at 75.3% RH. Assuming a linear output, I easily convert the humidity sensor's output voltage to a relative-humidity measurement.

All is not done when the voltage is measured, though. Remember that relative humidity is temperature dependent. So, I have to add a little math to the mix.

I always know the temperature because I'm tracking it. So, using the formulas above along with the mechanical humidity and temperature measurements, I can derive the true relative humidity.

PGC CO_2 LEVELS

Using the UIP program that complements the Telaire 200 1 V, I arbitrarily set the maximum detectable carbon-dioxide level to 2000 ppm. Measuring the CO_2 levels is akin to measuring relative humidity. Again, most of the work is already done for me.

The Telaire can output a 0-1 0-V analog voltage that's proportional to the minimum and maximum ppm value set with the UIP program. Just as with the humidity sensor, the output is linear and the carbon dioxide levels are interpreted according to sensor output voltage.

DEPARTING SENSOR SPACE

So far, I've defined the sensor-scanning process. The PIC 14000 will use its talents to assimilate raw data and convert it to numerical format. Using a software-based serial interface, the data can be transmitted to the SBC 301 for storage and later processing.

DATA-COLLECTION SECTOR

The Tempustech SBC 301 is ideal **for** this application. With only 2 MB of flash onboard, we have to be skimpy with the code.

DOS would be overkill here, so let's embed our own kernel using Phar tap's TNT Embedded ToolSuite. The advantages to this are many.

First of all, the ETS is designed especially for embedded-system use. The ETS Kernel alone provides a simple operating system for running this PGC application.

Another plus is that the ETS product uses industry-standard compilation and debug tools. The program that will effect the PGC sensor scan and storage routines is composed of standard Microsoft C code.

All the set-up and initialization routines are part of the standard ETS Kernel, so I don't have to code that stuff. The most important reason to go this way is that the ETS Kernel is relatively small, weighing in at around 25 KB.

The ToolSuite connects a host PC to the embedded target via parallel **or** serial interfaces. Once connected, the software is developed on the host and transferred to the target embedded platform for execution.

You can clearly see the advantages. Again, I could write a whole article just on the ToolSuite, but I did that, too ("Embedded PC Development," *INK 77*). So, let's move on and describe what the code will do.

Basically, the SBC 301 need only poll the PIC 14000 at a predetermined interval and retrieve the numerical data into its solid-state disk. Software to effect the solid-state disk is included with the SBC 301 embedded PC. It's easy to implement and has a variety of configuration possibilities.

Since the PGC configuration won't be transmitting the data to any point in real time, I chose to configure the solid-state disk as drive C. This setup enables the SBC 301 to start the ETS Kernel in processor stand-alone mode.

When the collected data needs processing, a real-live drive D will be added that contains its own ETS-based program to process and serve the collected data.

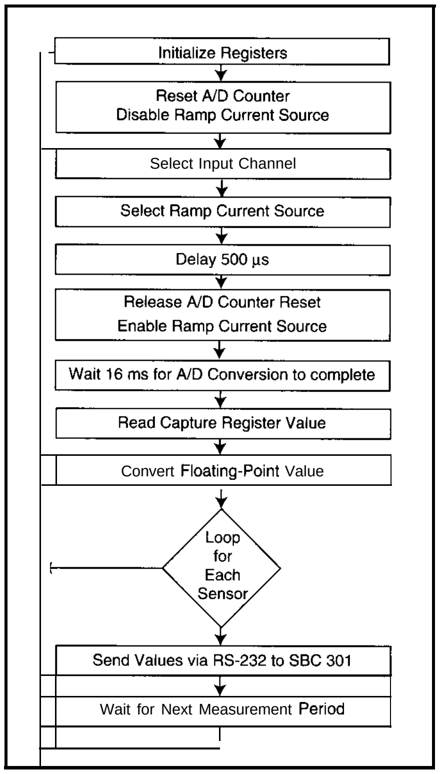

Initialize Registers
Reset A/D Counter
Disable Ramp Current Source
Select Input Channel
Select Ramp Current Source
Delay 500 µs
Release A/D Counter Reset
Enable Ramp Current Source
Wait 16 ms for A/D Conversion to complete
Read Capture Register Value
Convert Floating-Point Value
Loop for Each Sensor
Send Values via RS-232 to SBC 301
Wait for Next Measurement Period

Figure 2—Here's a ***PIC's*** *eye view of the firmware running in the* ***PIC*** *14000.*

80251 Embedded **Controllers**

MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-251
$199.00 (100 Quantity)

Midwest Micro-Tek is proud to offer its newest line of controllers based on the 8031/51/251 architecture. The 8031 comes in at a surprisingly low cost of $89.00 (100 quantity). Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro-tek.com
Guaranteed to be in your Future

#129

AMX The Real-Time Multitasking Kernel

680x0, 683xx	PowerPC™ family
80386 protected mode	i960® family
80x86/88 real mode	R3000, LR33xxx

NEW: • PowerPC support • *KwikLook* AMX-aware debug tool

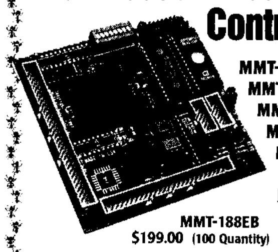
- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- Comprehensive documentation
- No royalties, source code included

For a sample of KwikLook and description of AMX,
Phone: ***(604)*** 734.2796
Fax: (604) 734-8114
E-mail: ***amxsales@kadak.com***
Web: ***http://www.kadak.com***

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

#130

Low Cost Embedded **Controllers**



MMT-188EB
MMT-Z180
MMT-HC11
MMT-196
MMT-31
MMT-251
MMT-PIC44
& OTHERS

MMT-188EB
$199.00 (100 Quantity)

If you're interested in getting the most out of your project, put the most into it. Call or Fax us for complete data sheets and CPU options. Custom work welcome.

MIDWEST MICRO-TEK
2308 East Sixth Street
Brookings, SD 57006
Phone 605.697.8521
Fax 605.692.5112
www.midwestmicro.tek.com
Guaranteed to be in your Future

#131

Listing 1—While most** of the work is done by the* `StartWebServer` *function,* `InitializeCriticalSection` ***acts as a function-entry traffic cop.

```
int main(void)
{
  BOOL fDone;               // flags program ready to terminate
  char ch;                  // character typed by user
  InitializeCriticalSection(&csHtml);

// Start Web server
  if (!StartWebServer(0)){
    printf("Can't initialize HTTPSERV.LIB\n");
    goto ERR: }

// Loop reading and processing keystrokes until user types "q"
// Since Z-byte keycodes are not currently used, they are discarded
  printf("*** Waiting for HTTP requests, type 'q' to exit\n");
  for (fDone = FALSE: !fDone; ){
    while (!_kbhit())
      // yield rest of time slice
      Sleep(10);
      ch = _getch0;
      if (ch == 0 || ch == (char) 0xE0){
        _getch();           // discard Z-byte keycode
        continue: }
      switch (ch){
        case 'q':
        case 'Q':
          printf("Q -- quitting program\n");
          fDone = TRUE:
          break;
        default:
          if (isprint(ch))
            printf("Unprocessed keystroke: %c\n",(int)ch);
          else
            printf("Unprocessed keystroke: %02Xh\n",
          (int)ch);
        break: }}

// User typed "q". Tell Web server to terminate.
  StopWebServer();
  DeleteCriticalSection(&csHtml);
  return 0;
ERR:
  return 1;
}
```

WEB TRANSPORT READY

You didn't really think I'd use all that computing power within the SBC **301** just to do simple async communications and data logging, did you?

When I spoke of "serving," I wasn't talking Wimbledon or the U.S. Open. I was talking Internet and Intranet.

Just so happens that the Phar Lap TNT package I have is the latest and greatest they've got. It includes services that enable the implementation of a full-blown Web server-embedded, of course.

The idea is to collect the data from the PIC and place it in nonvolatile solid-state disk. Once the PGC package is retrieved, add a D drive that contains the embedded Web server and present the collected data in any format desired via any standard GUI-based Web browser.

All of a sudden, I don't have to write any host data-display code and I can make the data display really fancy with minimal effort. I like that. Here's how the Phar Lap embedded MicroWeb Server works.

The Phar Lap ETS MicroWeb Server is a collection of libraries and plug-insdesigned specifically for embedded systems that are linked with a user-written application to create an embedded Web server running under the control of the Realtime ETS Kernel.

The MicroWeb Servercan generate HTML pages on-the-fly in standard C code. Presenting the data is no problem. It's just HTML!

Using server-side-generated HTML, the designer can embed any information collected from the PGC into static HTML pages. These on-the-fly pages can then be served via normal communications channels to any Web browser for viewing.

listing **2—Notice** the "needs slashes" entries. This makes sure that the correct syntax *is* entered regardless of the user's little fingers.

```
PAGE_ENT TopPageTable[] = {
  "", (void *)top_page, NULL,
  "hello.htm", (void *)hello,      "Hel1 o World",
  "hello2.htm", (void *)hello2,    "Hel lo World with embedded HTML"  ,
  "di rl/", (void *)DirlTable,     "Next level of HTML examples",
  "dirl", (void *)hpg_DirNeedsSlash, NULL,
  "debug/", (void *)onld_page_table, "On-line debugger",
  "debug", (void *)hpg_DirNeedsSlash, NULL,
  NULL };
```

Depending on the environment, the pages can be served via Ethernet LAN, asynchronous point-to-point protocol, and PPP or SLIP protocols. The desktop interface is designed with any of the standard commercial HTML editors. If you've ever put together an HTML Web page, you can design the data-display interface.

Four libraries make up the MicroWeb Server. All the real work is done within the HTTP Server Library.

This library implements an embedded HTTP/l .O micro server compliant with the RFC 1945 standard. User-written server programs call functions within this library to effect the actual network communications that make the server go.

The user program must begin the serving process by calling the StartWebServer function, which creates synchronization objects, initializes the WinSock interface, and starts a thread that sleeps while waiting to service incoming HTTP requests.

The mere mention of threadsand programmatic sleep implies that the ETS Realtime Edition supports real-time programming. In fact, it includes software mechanisms to effect a deterministic scheduler and manage preemptive multitasking of threads.

By using real-time-programming methodology, the MicroWeb Server checks in as a lightweight embedded application with heavyweightskills. Toefficientlyservice multiple HTTP requests concurrently, the server must be able to start a new thread for each HTTP request.

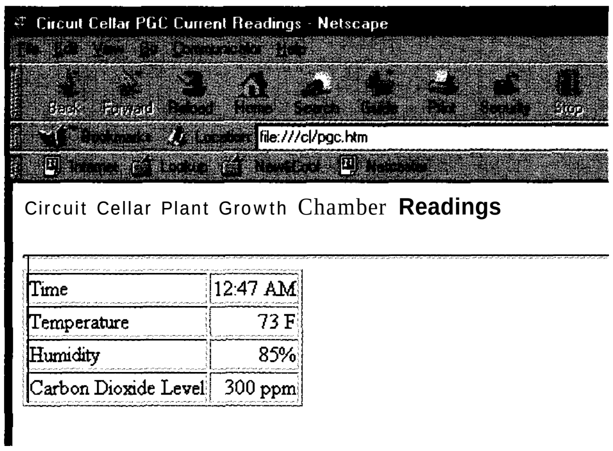
By starting a new thread for each client request, the chances of overrunning the socket connection queue are pretty much eliminated. As a result, HTTP request response times are reduced.

Just like the ETS Kernel process, the MicroWeb Server contains code ensuring that all necessary server start-up processes actually come up and run. We don't live in a perfect world, and neither does the data.

SendError is called to inform the user's Web application of any problems that may occur during server startup. The programmer may embed any information deemed **necessary** within the SendError code to help the user identify the problem.

The HTTPSERV. LIB library is the truth and the lightwhen itcomes to creating HTTP server threads. HTTP server threads process client requests and serve the proper HTML data and/or error pages.

Once the start-up process is completed without error, the designer's main program can simply loop waiting for userdefined keystrokes. These keystrokes are decoded to perform user-initiated tasks until a request



Circuit Cellar Plant Growth Chamber **Readings**

Time	12:47 AM
Temperature	73 F
Humidity	85%
Carbon Dioxide Level	300 ppm

Photo 1—It's not real fancy, ***but*** you ***get the*** **idea.** The ***HTML*** sky ***is the*** **limit.**

IF YOU DO

FUNCTIONAL TESTS

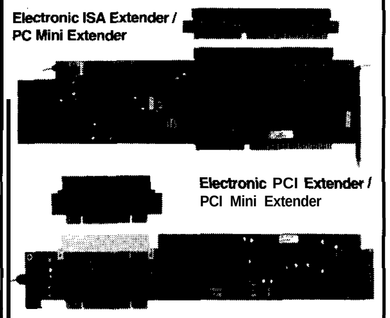
YOU NEED

HOT SWAPPING

ELECTRONIC

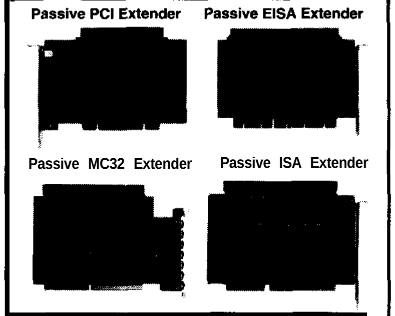
EXTENDERS

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 Cards**
- **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 Features**
- **Breadboard Area For Custom Circuitry**
- **And More...**

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



AZ-COM., INC.

3343 Vincent Rd., #D
Pleasant Hill, CA 94523 U.S.A.
Tel: 510-947-1000
Fax: 510-947-1900

24-Hour Fax on Demand:
510-947-1000 Ext.7
***EMail*: sales@az-com.com**
VISIT OUR HOME PAGE AT:
http://az-com.com/

#132

to serve data or terminate the Web server is keyed in.

In the MicroWeb Server example code included with the Tool-Suite, a q received from the keyboard calls the StopWebServer function, and you know what happens then. Listing 1 outlines the Web-server start-up task.

The second library is concerned with HTML pages. The functions included in the HTMLPAGE. LIB are primarily used to build HTML pages in real memory. You can also redirect these in-memory pages to a file.

Calls to HTMLPAGE create empty HTML pages, build error pages, and process in-memory GIF and HTML files. One useful feature contained within the HTMLPAGE library is involved in locating and processing URIs contained within page tables.

A URI (Uniform Resource Identifier) is a product of a URL. For the MicroWeb Server, page tables are directories of the preprogrammed or predefined HTML pages that are available to the server. When a request is presented in URL form, functions within HTTPSERV. LIB are invoked to translate the requested URL into a URI.

The URI is simply a DOS-compatible file path pointing to a particular file or directory. Each page-table entry consists of an HTML file name, a pointer to the function that produces the HTML page, and a text string used as a hyperlink to the HTML page.

For example, if our target HTML file is represented by the hyperlink "Hello World", our function to create the "Hello World" HTML page is hel1 o(), and the HTML page to be created is hel 1 o.htm, a simple click on the hyperlink begins the page-table locate process.

The page-table entries are then scanned and parsed. Once a URI match is found, the function associated with the selected URI kicks off creating the HTML page that ultimately winds up on the browser screen. Listing 2 shows how a typical page-table entry is formatted.

This is a very powerful concept. Custom HTML pages can be stored and dynamically assembled with real-time data on-the-fly.

The remaining two libraries are known as "on the fly" libraries. Function calls are made to HTML.LIB and HTMLFORM.LIB during the page-creation process. These libraries contain all the functionality necessary to assemble each individual part of the HTML page to be served.

The results of the on-the-fly page-creation process are no different than if you were to hand code a page. Listing 3 is pretty simplistic, but it's a good picture of how a typical on-the-fly HTML page is generated.

MISSION REPORT

Where are we? We've started the PIC14000, and the SBC 301 is polling the PIC for data. All that's left is to process the collected data and send it to a browser. Since there's an Ethernet card on the ground-support PGC configuration, I'll use it as the data-transport medium.

Most of the time, space projects depend on funding. Funding depends on good salesmanship. And, good salesmanship relies on good presentation skills.

Photo 1 is the pretty picture containing the PGC data. I used the page-table method to get the Web page to this point.

SPLASHDOWN

Yeah, I know. They land 'em these days. But, my point is that the Internet and Intranets are becoming the way the world communicates. The Web browser has become the defacto user interface standard in the Web world, and I expect it will become the preferred way to view data outside of that world, too.

Do you know ***anyone*** who hasn't used a Web browser? **Do you** know anyone who ***can't*** use a Web browser? I didn't think so.

With the assistance of Phar Lap's MicroWeb Server and some really tricky PGC sensor components, we have implemented an embedded data-collection platform that can serve its database to anyone with a Web browser anywhere in the world. APC.EPC

Fred Eady ***has over*** 20 ***years'.*** experience as a systems engineer. He has worked with computers and communication systems ***large and small, simple*** and complex. His forte is embedded-systems design *and communications. Fred may* be reached at *fred@edtp.com.*

Listing 3—Looks familiar, huh?

```
BOOL _cdecl  hello(REQ_INPUTS *pInp, REQ_OUTPUTS *pOutp){
  /* Open empty HTML document first */
  if (!hpg_CreatePage(pOutp))
    return FALSE:

  /* Build HTML document */
  html_brtext(HTML_TITLE, "Hello world HTML demo page");
  html_text("\n");
  html_endtag(HTML_HEAD "\n");
  html_tag(HTML_BODY "\n");
  html_text("Hello world!!!!\n");

  /* Return completed HTML document */
  return  hpg_DonePage(pOutp);}
```

SOURCES

PIC 14000
Microchip Technology, Inc
Chandler, AZ 852246199
(602) 786.7200
www.microchip.com

MicroWeb Server, TNT **ToolSuite**
Phar Lap Software
60 Aberdeen Ave.
Cambridge, MA 02138
(617) 661-1510
Fax: (617) 876-2972
BBS: (617) 661.1009
Telnet: bbs.pharlap.com
www.smallest.pharlap.com

CardTrick, ROM-DOS
Datalight
1881059thAve. NE
Arlington, WA 98223
(360) 435-8086
Fax: (360) 435-0253
www.datalight.com

Survivor II Sensor
HyCal Engineering
9650 Telstar Ave.
El Monte, CA 9173 I-3093
(8 18) 444-4000
Fax: (8 18) 444-I 3 14
www.hycalnet.com

Ventostat **2001V**
KELE & Assoc.
2975 Brother Blvd.
Bartlett, TN 38 133
(901) 382-4300
Fax: (901) 372-253 1
www.kele.com

VMAX SBC 30 1
Tempustech
295 Airport Rd.
Naples, FL 34104
(941) 643-2424
Fax: (941) 643.4981
www.tempustech.com

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

DEPARTMENTS

66 MicroSeries

74 From the Bench

78 Silicon Update

Ingo Cyliax

MC68030 Workstation

The Boot PROM Monitor & Device Drivers

Part 2 of 3

Stranded by Motorola's decision to discontinue an affordable MC68000 eval board, Ingo built a 68k platform to teach programming in assembly. After covering the hardware last month, he looks at the boot-PROM monitor and how to build device drivers.

Last month, I described the hardware architecture for a 68030-based workstation used in our computer-architecture lab at Indiana University. In this installment, I want to discuss the onboard monitor software on this machine as well as some issues that arise when writing device drivers for I/O devices that are onboard or attached to this system's ISA bus.

But, let's start with what the monitor does. 1'11 cover how it initializes all the I/O devices and the processor to a consistent, usable state.

Once this is done, I'll show you how the monitor implements a simple command-line-based user interface. This enables the student to download and boot code from the network; boot from a floppy or IDE drive; view and change memory, I/O, and processor registers; and debug programs.

In our typical lab configuration, these machines have a VGA card and monitor, floppy and IDE disks, an AT keyboard, two RS-232 serial ports, one parallel port, and an Ethernet card. The monitor supports all these devices.

MONITOR

The monitor resides in the system's boot PROM and is activated at reset. It was written entirely from scratch—

mostly in C, with some of the start-up and exception processing routines in 68k assembly language.

The complete monitor may seem fairly Spartan since it doesn't implement fancy command-line editing, history functions, or even a programming/scripting language. However, if you consider that it fits in a single 32-Kb EPROM, you'll realize it gets a fair amount of bang for the buck.

Let's look at the monitor's operation and features in more detail. At reset, the 68030 expects to load the system stack pointer at location 00000000 (hex) and the initial PC at location 00000004. Besides these vectors, many exception vectors reside below 00000400.

The reset vector causes the CPU to start executing the cold-reset start-up code. The first thing the start-up code does is to disable interrupts. While the interrupts are already off after a real reset, this task enables any code to use the reset vector (which is in a known location) to cause software to give up control and reset the system.

Once the interrupts are off, the monitor checks whether it's already running from DRAM by checking the actual running address in the PC against the desired address the monitor was linked for. If the addresses don't match, the monitor copies the desired address to DRAM (at physical address 80000000 [hex]) and resumes operation there.

Listing l-i *de_cmd() tries to read or write a block of data from the IDE hard disk by first seeking the location and then reading or writing the disk data through the data port.*

```
ide_cmd(un,cmd,hd,cy,se,buf,len)
int un,cmd,hd,cy,se;
short *buf;
int len; {
  int i,s,st;               /* busy? */
  while(((st=inp(IDE(7)) & (SBSY|SRDY))) != SRDY);
    outp(IDE(2),len/512); /* sec  count  */
    outp(IDE(3),se);        /* sector */
    outp(IDE(4),cy&0xff); /* cylinder low byte */
    outp(IDE(5),cy>>8);     /* cylinder high byte */
    outp(IDE(6),0xa0 | (un  <<4) |
          (hd & 0xf) );     /* secsize,unit,head */
    /* wait for drive to become ready */
    while(!(inp(IDE(7)) & SRDY));
      outp(IDE(7),cmd);     /* command */
      /* do a write */
      if(cmd == 0x30){      /* wait for drive to accept data */
        while(!(inp(IDE(7)) & SDRQ));
         len = len / 2;
        i = 0;
        while(i < len){
          outpw(IDE(0),*buf++);
          i++; }
        i = i*2;            /* wait for completion of write here? *
        if((st = inp(IDE(7))) & SBSY){
          while((st = inp(IDE(7))) & SBSY);}
        if(st & 0x01){      /* an error occured */
          printf("ide_cmd: Error status %x error %x\n",
                   st,inp(IDE(1)));
          return(-1); } }
    else{
      if((st = inp(IDE(7))) & SBSY){
        while((st = inp(IDE(7))) & SBSY);}
      if(st & 0x01){        /* an error occured */
        printf("ide_cmd:Error status %x error %x\n",st,inp(IDE(1)));
        return(-1); }
      i = st:
      if(len){              /* is drive ready to send data? */
        while(!(inp(IDE(7)) & SDRQ));
          len = len / 2;
          i = 0;
          while(i < len){
            *buf++ = inpw(IDE(0));
            i++; }
          i  =  i*2;}}
  return(i); }
```

The code also changes the base of the exception vector table to start in DRAM. There are several reasons for this.

Since the boot PROM is an 8-bit device, execution speed is enhanced if it's copied and run from 32-bit-wide DRAM. Also, DRAM makes it easy to alter the exception-vector table and patch the monitor code to add support for new devices and fix bugs if needed.

After the monitor and exception table is relocated, the assembly-language start-up code is done and the C code is called. If it's the first time into the monitor code after a real hardware reset, the monitor initializes all the needed I/O devices and prepares itself to boot from the IDE drive unless the user enters a character to interrupt the auto-boot sequence.

If the auto-boot sequence is interrupted in time, the monitor enters a command-line-based interface so the user can execute monitor routines to boot from an alternate device or debug a program. If the monitor is entered from either a software-initiated reset or an exception that isn't trapped by another program, the auto-boot sequence is bypassed and the command loop entered directly.

The code also prints out the current state of the PC and status register and what exception caused control to be transferred to the monitor. This scenario occurs when a user program encounters an exception (e.g., an address/bus error or divide-by-zero) or when an unexpected interrupt occurs.

Let's look at what it takes to boot the system from one of the boot devices. The monitor is supported for booting from an AT-compatible IDE drive and floppy drive on the ISA bus. It also knows how to boot from Ethernet using one of two possible 8-bit Ethernet card types (also on the ISA bus).

Booting from a disk device is relatively straightforward. The monitor reads the first sector from the floppy or IDE drive to location 80008000 (hex) and extracts the number of blocks and location on the media to start reading the boot image.

The format of the boot sector looks like Table 1. After loading the boot sector, the monitor continues to read

1F0	Data Register
1F1	Error Register
1F2	Sector Count Register
1F3	Sector Number Register
1F4	Cylinder Register Low
1F5	Cylinder Register High
1F6	Drive/Head Register
1F7	Status/Command Register

Figure 1 —*To perform a disk I/O operation, the sector location address must be programmed before a command can be given. The Data register is 16 bits, whereas the rest of the registers can be accessed via 16- or 8-bit I/O instructions.*

the boot image until all specified blocks are read.

Once the boot image is in memory, the monitor does the equivalent of a subroutine call starting at address 80008000. There, a branch instruction should cause execution to continue somewhere in the boot image, depending on the application.

The Ethernet boot code is similar. It uses standard Internet protocols to determine the boot image's location and filename on the network and uses another standard protocol to download the image to location 80008000.

Once loaded, it executes a subroutine call to the start of the boot image. This way, the same boot image can be loaded from floppy/IDE and Ethernet without changes to the image. In Part 3, I'll discuss the protocols used in loading the boot image over the Ethernet.

Once control is given to the boot image (usually some kind of operating system), you only get back to the monitor if an unexpected exception occurs that the OS can't handle or if the OS gives control back to the monitor.

When students are learning to write code for this system, we need to load code images and allow debugging from the monitor code. The student normally assembles and compiles code on a workstation and manages to get the code image into system memory for debugging. (More on how the program image gets there next time.)

The monitor's debugging facilities are Spartan so the student is exposed to a different environment than they're used to. Since the lab's purpose is to teach computer-science students computer architecture, we want them to see low-level machine constructs. We hope they learn how abstract programming constructs in C/C++ programs are implemented.

They'd also never experience the joy of doing a manual stack trace to figure out where their program bombed. Remember, this lab is as close as most computer-science students get to programming real hardware these days.

After they load a code segment into memory, they can use the monitor's `go` command to start executing at a specified address. They may also set a breakpoint anywhere in their code.

When the processor encounters a breakpoint, which is implemented with a software interrupt (`trap`) instruction, an exception control jumps back to the monitor where the context of the running code is saved. The user can then examine and modify registers as well as memory and continue executing their code. The user can also trace their program's execution, for which the 68030 has a special hardware facility.

Another useful feature of the monitor PROM is the ability to save more than context. This feature enables the student to save the context of an OS, if active, and the context of the program being debugged. There are two default contexts for this purpose and a shortcut command that restores the OS context and continues execution.

DEVICE DRIVERS

Of course, the monitor also has to implement device drivers for several devices that are present. I already discussed the boot sequence, so let's look at some of the low-level routines used to implement the device driver for the boot devices.

The IDE controller contained on the disk drive uses a parallel port on the IDE interface card to connect to the ISA bus. Last month, I talked about how the ISA-bus interface is implemented.

To understand how the software works, you only need to know that the ISA bus is divided into four different memory areas depending on what type of ISA-bus bus cycle is required. These are 8- and 16-bit memory space access and 8- and 16-bit I/O access.

The IDE interface uses 8-bit I/O accesses to communicate with its registers. The register map and I/O locations for the first IDE drive can be seen in Figure 1. The data is transferred through a single 16-bit I/O register using programmed I/O.

Listing 1 starts the sequence of steps necessary for reading from and writing to the IDE disk. The routine `ide_cmd` sets up the IDE controller with the desired disk address by programming the appropriate register.

When reading, initiate the command and poll the status register until the disk drive is idle. Then, check the status register for any errors. If none are found, read the data from the data register.

Address	Description
000 (hex)	Branch to start
004 (hex)	Starting block number of boot image
008 (hex)	Number of blocks in boot image
00c–1ff	Not used by monitor

Table 1 --The *boot sector contains information for the* boot loader about *where to find the boot image and how long it is.*

For writing, the data should be transferred through the data register before issuing the `write`. The data is simply read or written to the data register until the required amount is transferred.

The floppy driver is a bit more complicated since some timing issues are involved. The floppy driver is another reason why we want to execute from 32-bit memory and why the monitor was moved to DRAM.

It would have been possible to write a floppy driver that ran from 8-bit memory. But, it would have required extensive assembly-language programming to make it work fast enough during the data-transfer phase.

The data-transfer phase involves the most critical timing. There's only a single register to buffer the data to and from the floppy disk.

3F2	Operations Control Register
3F4	Master Status Register
3F5	Data Register
3F7	Data-Rate Register

Figure 2—*The* Operations register *resets the controller and* spindle-motor *control, while the* Data-Rate register programs *the* data rate *and density desired. Most of the control is accomplished by sending commands to and reading status packets from the Data register.*

If we're too slow in reading the data, the data coming from the floppy disk overruns the data register, causing an error. If we're too slow in writing the data to the floppy, the data register is empty when data needs to be serialized to the floppy disk, resulting in a data underrun. In a real PC architecture, the DMA offloads the CPU from these timing requirements.

Motor control brings up another complication on the floppy controller. The floppy drive has a motor to drive the spindle. However, if it runs all the time, the media can become worn out.

So, the motor needs to be started before any floppy operation can proceed. It also needs to be stopped sometime after the last floppy operation is done.

Other than that, the floppy works mostly like the IDE drive. The driver calculates the sector number, cylinder, and head number from the block number requested, sets these parameters into the appropriate register, and requests a seek operation.

Listing 2-f *7oppy_read() seeks the specified location on the disk and then transfers the data from the floppy data port to memory. Since the 68030 motherboard doesn't implement DMA, programmed I/O is required to transfer the data.*

```
floppy_read(drv,block,buf,len)
int drv;
unsigned block;
register unsigned char *buf;
int len; {
  int sec;
  int trk;
  int hd;
  register int n;
  unsigned char resp[7];
  unsigned short x;
  register unsigned char i;
  floppy_seek(drv,block);
  trk = block / 18;
  sec = (block % 9);
  hd = (block / 9) % 2:
  len = (len > ((9-sec)*512))?(9-sec)*512:len;
  n = len;
  fdc_out(0x66);   /* read command MFM,SK */
  fdc_out(drv|hd<<2); /* drive & head select */
  fdc_out(trk);    /* cylinder */
  fdc_out(hd);     /* head address */
  fdc_out(sec+1); /* sector address */
  fdc_out(2);      /* sector size 512 */
  fdc_out(9);      /* end of track */
  fdc_out(0x1b);   /* gap */
  fdc_out(0xff);   /* data length, only if sector size == 0 */
  while(n){        /* anything happening? */
    while(!((i = inp(FDC(4))) & 0x80)); /* still executing */
      if((i & 0xe0) == 0xe0){
        *buf = inp(FDC(5));
        buf++;
        n--; }     /* oops, not executing, must be response data */
      else if((i & 0xe0) == 0xc0)
         break; }
    for(x=0;x<7;x++){
      while((inp(FDC(4)) & 0xe0) != 0xc0);
        fdc_in(&resp[x]); }
    if((resp[0] & 0xc0) || n){
      if(!n && (resp[0] & 0xf8) == 0x40 &&
        (resp[1] == 0x10|| resp[1] == 0x80) && resp[2] == 0x00)
        got0 out;
      puthex2(i);
      putchar(' ');
      for(x=0;x<7;x++)
        puthex2(resp[x]);
      putchar(' ');
      puthex4(n);
      putchar('\n');
    return(-1); }
  out:
    return(len   n); }
```

Tight Budget '51 Emulation

8051 Family Emulator is truly Low Cost!

The DrylCE Plus is a modular emulator designed to get maximum flexibility and functionality for your hard earned dollar.

805family processor pods that are low in price. Features include: Execute to breakpoint, Line-by-Line Assembler, Disassembler, SFR access, Fill, Set and Dump Internal or External RAM and Code, Dump Registers, and more. The DrylCE Plus base unit is priced at a meager $299, and most pods run only an additional $149. Pods are available to support the 8031/2, 8751/2, 80C154, 80C451, 80C535, 80C537, 80C550, 80C552/62, 80C652, 80C85 1, 80C320 and more. Interface through your serial port and a comm program. Call for a brochure or use INTERNET. We're at **info@hte.com** or **www.hte.com.**

Tighter budget? How about *$149* emulation?

Our $149 DrylCE model is what you're looking for. Not an evaluation board - much more powerful. Same features as the DrylCEPlus, but limited to just the 803 1/32 processor.

You can afford it!

So, if you're still doing the UV Waltz (Burn-2-3, Erase-2-3). or debugging through the limited window ROM emulators give, ***call us now*** for relief! Our customers say our products are still the best Performance/Price emulators available!

Look into our Single Board Computer solutions, too!

HTE

HiTech Equipment Corp.
9672 Via Excelencia
San Diego, CA 92126
[Fax: (619) 530-1458]

Since 1983

(619) 566-l 892

VISA MasterCard

Internet e-mail: info@hte.com
World Wide Web: **www.hte.com**

#136

Once completed, the data can be read or written to the data register. But, you need to make sure the floppy is ready by read-testing the data-request status bit. Listing 2 shows the code for reading one sector from the floppy, while Figure 2 shows the I/O register layout.

ETHERNET CONTROLLERS

Now that you know how to control the floppy and IDE drive, what about an Ethernet device? Ethernet controllers are a bit more complex even at the hardware level. Let's examine the code that sends and receives a packet on one of the controllers.

Ethernet runs at 10 Mbps, which turns out to be a little better than 1 MBps. Since many Ethernet packets are not for this node, it's wasteful to get the processor to read every packet. It would keep an 8-bit ISA-bus system 100% utilized whenever there's traffic.

To reduce the I/O requirements, Ethernet cards employ some kind of buffer to store at least one received packet. If this packet isn't destined for this station, the write point to the packet memory is simply reset.

Once a packet is received, the Ethernet card signals the processor with an interrupt request or a bit in the status register that it's done. Listing 3 shows the code for receiving a Ethernet packet from an Ethernet card.

Sending packets is easy. The CPU copies the packet to be transmitted to the card's transmit buffer and tells it to start. If the Ethernet card transmits the buffer, it simply indicates that it's done and interrupts.

Transmitting the packet can take a long time. The card has to find an idle period on the Ethernet and then try to transmit. If another station tries to transmit, a collision occurs and both cards retry after a randomized timeout.

These actions are transparent to the software, but you need to be aware of them, since it may take a while before the transmit request completes. Check out the transmit routine in Listing 4.

KEYBOARD

In Part 1, I talked about the physical interface of the keyboard, which involved how the data and clock interface to the USART in the 68901 MFP chip.

Listing 3-we8003 read() pulls a received Ethernet frame from the Ethernet card's internal memory. Since frames are stored in a circular queue on the card, pull the oldest frame and advance the tail pointer for another.

```
we8003read(ifp,buf,count,timer)
struct ifconfig *ifp;
unsigned char *buf;
int count:
int *timer; {
  int len;
  unsigned long x;
  struct we8003 *we = (struct we8003 *)ifp->ifc_addr;
  unsigned char *mem;
  int head,tail,i,stat,resid;
  #define ltail   (*(unsigned char *)(ifp->ifc_addr2 + 0x5ff))
  len = 0:
  x = 1000* *timer;
  mem = (unsigned char *)ifp->ifc_addr2;
  while(x && !len){
    while(!((stat = we->we_p0_isr) & 0x05) && x)
      x--;
    if(!x) break:
      if(stat & 0x04){
        i = we->we_nic_reg[13];
        i = we->we_nic_reg[14];
        i  = we->we_nic_reg[15];}
  tail = we->we_p0_bnry;       /* first packet */
  we->we_cmd = 0x42:           /* ps=1, running */
  head = we->we_p1_curr;       /* next free */
  we->we_cmd = 0x02:           /* ps=0, running */
  /* printf("isr %x hd %x tl %x lt %x\n",stat,head,tail,ltail); */
  if(ltail)
    tail = ltail;
  while(tail != head){
    i = tail<<8;
    len = mem[i+3]<<8;
    len |= mem[i+2];
    tail = mem[i+1];
    /*printf("  stat %x nxt %x len %x\n", mem[i],mem[i+1],len); */
    len -= 4;                  /* lop off CRC */
    len = len > count ? count : len;
    /* need to check for wrap--ped buffers */
    resid = WE_MEMSIZ-i-4;
    if(resid < len){
      bcopy(&mem[i+4],buf,resid);
      bcopy(&mem[0x600],&buf[resid],len-resid); }
    else
      bcopy(&mem[i+4],buf,len); }
    ltail = tail:
    tail--:
    if(tail < 6)
      tail = 0xlf;
    we->we_p0_bnry = tail:
    we->we_p0_isr = 0xff; }
  return(len); }
```

Listing 4—Compared to reading Ethernet frames, sending them is straightforward. Just copy it to the Ethernet card's internal memory, and tell it to go. Since the monitor uses polling, wait here to make sure if was sent OK.

```
we8003write(ifp,buf,count)
struct ifconfig *ifp;
unsigned char *buf;
int count; {
  struct we8003 *we = (struct we8003 *)ifp->ifc_addr;
  unsigned char *mem = (unsigned char *)ifp->ifc_addr2;
  count = count > 60 ? count : 60;
  bcopy(buf,mem,count);
  we->we_p0_tbcr0 = count & 0xff;
  we->we_p0_tbcr1 = count >> 8;
  we->we_cmd = 0x06;       /* transmit, running */
  while(!(we->we_p0_isr & 0x0a));
    we->we_p0_isr = 0x0a;
  return(count); }
```

On the software side, the interface looks like a standard USART interface.

When a byte arrives from the keyboard, the receive-buffer fill bit gets set in the status register and an interrupt is generated when enabled. Reading the received byte from the data register resets the receive-buffer full condition and readies the USART for another byte.

It seems simple, but the received byte is only the scan code from the keyboard or, worse, one byte of a multiple-byte message. What the monitor really needs is an ASCII character that responds to the keycap legend on the keyboard.

The keyboard driver achieves this by looking up the scan code-one for each key-in a character table. The driver also tracks the state of the keyboard shift and control keys, which modify the ASCII code. Listing 5 explains all.

Listing 5—The keyboard routine pulls bytes from the USART in the MC68901 (MFP). **These bytes are fhe scan codes** from the keyboard and need to be translated info ASCII codes using a look-up fable (i.e., co de tab [ ]). The receive routine a/so tracks the state of the shift/capslock and control keys.

```
kbd_getc() {
  unsigned char c;
  while(!(c=kbd_stat())) ;
    return(c); }
kbd_stat() {
  unsigned char *port = KBD:
  unsigned char x,ox;
  extern struct codetab codetab[];
  static int brk=0;
  if(!(port[0x15] & 0x80))
    return(0);
  x = port[0x17];
  if(x == SC-BREAK) {
    brk = 1:
    return(O); }
  if(x > 0x7f){
    brk = 0;
    return(0); }
  if(brk){
    brk = 0;
    ox = codetab[x].code;
    switch(ox){
      case CC_LSHFT
      case CC_RSHFT: shft = 0: break;
      case CC_CTRL: ctrl = 0; break:
      default: break; }
    return(0); }
  ox = codetab[x].code;
  switch(ox){
    case CC_LSHFT:
    case CC_RSHFT:   shft = 1; break;
    case CC_CTRL:    ctrl = 1; break;
    case CC-CAPS:    caps = caps?0:1; break;
    default: if(ctrl)
                return(codetab[x].cchar);
             if(shft|| caps)
                return(codetab[x].schar);
             return(codetab[x].uchar); }
  return(O); }
```

VGA DRIVER

Of course, I saved the hardest until last-the VGA driver. It's hard because each VGA chip and card has a different low-level configuration register set.

In the PC world, this situation is handled by calling the appropriate initialization and mode change routines from the BIOS on the card itself. The code is usually proprietary, and I'd need an Intel instruction emulator for my system to execute the routines.

I ended up writing a driver for only a limited number of possible VGA chipsets, including the Paradise and Western Digital 8-bit VGA chipsets for which I already had extensive datasheets. Other chips/cards may also work if they behave like a Western Digital chip.

The monitor only needs a character-based interface, so I initialize the VGA chip to its CGA mode, which emulates the 6845-based CGA card. The monitor then has a character-based frame buffer with 8 KB of video RAM. To speed up scrolling, the monitor's character-output routine uses the extra display memory to page the display.

UNTIL NEXT MONTH...

I hope I've given you an idea of how involved even a simple debugging monitor can become-without even trying.

Next month, I'll discuss the software-development environment the students and I use to write software. I'll also tell you about some of the things we've accomplished with this system. ❑

Ingo ***Cyliax is a research engineer in the Analog VLSI and Robotics Lab and teaches hardware design in the computer-science department at*** *Indiana* ***University. He also does software and hardware development with*** *Derivation* ***Systems, a San Diego-based formal-synthesis company. You may reach*** *Ingo* ***at*** *cyliax@EZComm.com.*

REFERENCES

Motorola, ***Programmer's Reference*** *Manual,* Motorola Literature Distribution, Phoenix AZ, 1992.

Western Digital, 1992 ***Devices,*** *Sys*-***tern Logic, Imaging, Storage,*** Western Digital Corp., Irvine CA, 1992.

SOFTWARE

Complete source code for the monitor can be found at <ftp.cs.indiana.edu/pub/goo/mc68030>.

SOURCE

MC68xxx
Motorola
MCU Information Line
P.O. Box ***13026***
Austin, TX 78711-3026
(512) 328-2268
Fax: (512) 891-4465
freeware.aus.sps.mot.com

I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

FROM THE BENCH

Jeff Bachiochi

Prototypes of the Rich and Famous-Not!

Once, the undersides of proto-type boards were a spaghetti-factory wonder. Now, with the advent of mail-order PCBs, engineers can unwire their prototypes. Heck, they can just E-mail them in via the Internet.

Where would I be without my local Radio Shack-or for that matter, the mail-order Digi-Key, Jameco, and Mousers who can get me those low-quantity parts fast?

We'd all be in deep trouble without access to these stores. Taking our pencil-sketch designs into 3D reality is what gets an engineer's blood pumping.

The engineer is like the painter, and the PCB is his canvas. Both begin with blank panels and end with a work of art.

Wiring up a prototype without a PCB is a laborious task. Although tools abound to make the task a wee bit easier, nothing lends itself more to circuit legitimacy than a real PCB.

Unfortunately, PCBs seem to be like some of those specialized electronic parts that can only be obtained in high quantities or in high-priced evaluation units by big-name companies. But, what mail-order has done for the availability of electronic parts it has now done for prototype PCBs.

MAIL-ORDER BRIDE

Not looking for a mate, you say, but you could use a good prototype PCB if it wasn't too expensive? Well, listen up.

You no longer need to order a hundred pieces. Now, there are many shops around the country who specialize in making a few quick, inexpensive, prototype PCBs of your design.

Around the country is not as far away as you might think. Everything can be handled over the phone-be it sending your photoplot files directly to the service's BBS or indirectly to its Internet site. In just days, the PCBs are delivered directly to your door via mail or special carrier [see Photo 1).

Let's look at the process in a little more detail, and then maybe you'll feel comfortable enough to explore it on your own.

SHOP SHOPPING

Don't expect to find a PCB shop next to the l-hour Moto-Foto at the

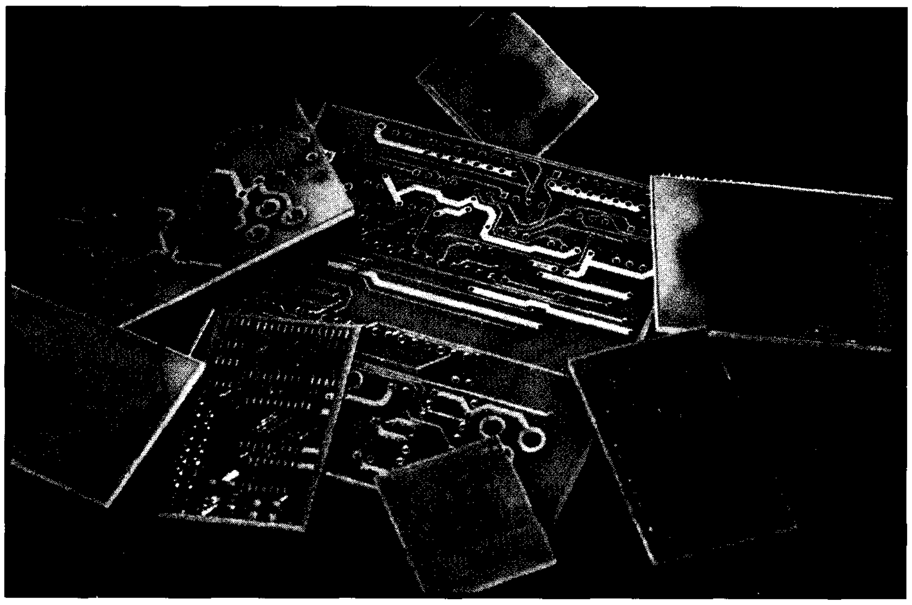


Photo 1 -Prototype *PCBs of various sizes don't have to cost an arm and a leg. Depending on the size and quantity, many boards are under $100.*

nearest mall. I get direct mailings, card decks, and magazines advertising fast-turn proto-board houses every month.

The magazine you've got in your hands right now has a few choice fabricators advertising in the back pages. You'll find information easy to gather, as many have Web sites promoting their services.

Getting quotes isn't really necessary because the costs are spelled out for you. Use the given formula for estimating your own costs.

You'll find cost to be based on two factors-a flat fee plus a cost per square inch. The flat fee pays for the labor involved in preparing your zipped files for processing. The cost per square inch pays for the materials and labor to fabricate your PCB.

Usually, there's a minimum square-inch buy, but unless you're getting very small boards, that's generally not a problem. For instance, a couple of 5" x 7" boards will set you back less than $100.

That 70 in.2 of PCB can be divided up into 10 pieces of a 2" x 3" board. Remember, the total area is made up of not only usable board space but also the waste area around the PCB that will be whacked off.

So, what do you get for your Ben Franklin?

Although not every PCB house plays by the same rules (visit their Web site or BBS, or call for the real scoop), you will basically get a double-sided PCB sheared (not routed) to roughly your external dimensions (four straight cuts).

The PCB will have plated through holes drilled to a maximum density of about 24 holes per square inch. You may be limited to particular drill sizes or a maximum number of drill changes, so read carefully. Exceeding the rules may cost you additional dollars.

Minimum technology is generally 8-mil traces and 8-mil spaces. That's 0.008" minimum trace width with a minimum 0.008" clearance around everything.

Although soldermasks and silkscreens aren't included in this price, they can be obtained for an extra fee. I don't get either of these on my prototypes.

Photo 2—*The* 1210 *and* 0603 *sizes practical/y disappear on the face* of *Abe Lincoln. Tiny park are* ***almost*** *impossible to p/ace by hand.*

PLAYING BY THE RULES

One of the first things you'll find is that all this works only if you play by the rules and give these vendors photoplot files in a format they can work with. The rules are spelled out rather completely-often with examples. However, they can be a bit heady if you don't know the lingo (e.g., RS-274-D format).

Essentially, all files must contain only ASCII information. The files you need to include to the vendor for this basic service are the photoplot files of the top and bottom layers, the aperture list for those plot files, the NC drill file, and the tools list for the NC drill file.

The aperture list indicates which size and shape drawing tool to use for each of the photoplot steps to create your top and bottom layers. The NC tool list tells which drill sizes should be used at each of the NC drill-file coordinate locations.

In addition to these five files, a README.TXT file is often included. This file is generally an order form which spells out exactly what you're asking for, which pieces you're providing, and how you'll pay for it.

Believe it or not, this can sometimes be the most important file of all. If problems arise, it might be your only bargaining tool. Zip these files together, and you're ready to cast your design in FR4.

WHAT, NO LAYOUT TOOLS?

You've always wanted to get a prototype PCB but couldn't afford the cost of the layout software? Well, there seems to be a new wave of inexpensive CAD tools out now.

Almost all the mail-order parts suppliers offer inexpensive software packages. While you're visiting Web sites, be sure to check around the site for free public-domain software.

Yup, you heard right. Free. Don't expect to get all the latest super tools, but you should be able to create excellent PCB designs and output the necessary files for the fab house.

COMPONENTS AND TECHNOLOGY

Many think surface-mount components come in a single size-small. But, there's small, and then there's tiny. The size you choose may have more to do with the technology PCB you're shooting for than the absolute minimum real estate a component takes up.

Take a look at the SMT resistors in Photo 2. The sizes range from 25 12 down to 0405. These numbers tell their own story. They are the length and width of the component in millimeters. One can hardly pick up the smallest with tweezers.

As parts become smaller, pick and place equipment needs to be more sophisticated. So, some vendors won't be able to work with some parts.

Warning #1: If you plan to use a specific assembly house sometime in the future, make sure they can handle the parts you've chosen in your design.

The least expensive PCBs are single sided. These boards have traces on one side only and any through holes have no plating.

Warning #2: Provide sufficient pad sizes. Inexperienced designers of single-sided boards tend to forget that without enough copper pad, stresses will fatigue the bond between the copper and the fiberglass.

Since the through holes are not plated, all the mechanical stress of a component's lead is placed on the adhesive strength of the solder pad. A normal-size pad is too flimsy and doesn't provide adequate mechanical strength for those heavy components.

Small SMT circuits can be designed on single-sided PCBs. The larger SMT parts are valuable here since their pads are far enough apart to allow traces to route underneath them, avoiding the need for jumpers (i.e., those nasty extra components necessary to get a signal across a busy maze of traces).

The most common PCBs today are double sided. This technique requires through holes to be plated with copper, making an electrical connection from the trace on one side to the trace on the opposite side.

This task is performed before any copper has been removed on the outer surfaces. The solid copper surfaces give a path for current necessary in electroplating the holes.

Plated holes also give through-hole components a good mechanical as well as electrical bond to the PCB. Capillary action enables solder to wick up through the holes, making a fillet of solder around the component lead on top as well as on the bottom of the PCB.

While double-sided PCBs are more expensive, the layout can be crammed together a lot tighter, in effect giving you a smaller PCB. In the case of SMT parts, the double-sided board lets parts be mounted on both top and bottom.

But, that doesn't mean twice the parts in the same space. After all, some real estate must be set aside for component interconnections.

Although a double-sided PCB might be considered multilayer, usually multilayer refers to more than two layers. A designer might need to use more than two layers for a couple of reasons.

First and most obvious, additional layers mean additional real estate for routing connections. Although the inner layers cannot hold components, this setup does not free the layer to 100% routing.

Vias or feed throughs must be used to connect the components on the outer layers to the interconnecting traces on the inner layers, and these take up valuable routing space. So, as the number of layers increases, the actual routing space per layer goes down. Eventually, adding more layers won't increase the available routing area at all!

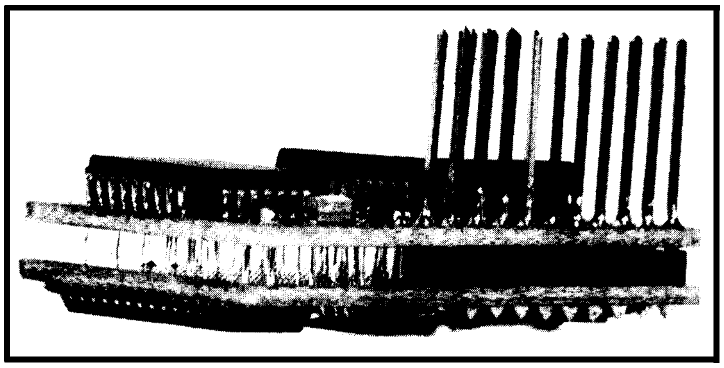


Photo 3—*Simulating* a *four-layer board can save a lot of money. You can expect to pay 10–15 times as much for a* **real** *four-layer prototype.*

The second reason to go multilayer is to add ground and power planes, The addition of layers exclusively used for power and ground can reduce the need for as many decoupling capacitors, which reduces EM1 and provides the lowest resistance paths for power and ground connections to the circuitry.

By removing the power and ground connections from the outer layers, many designs are simplified to the point that the remaining connections can be completely routed on the outer layers.

MULTILAYER t PROTOTYPE ❑ $$$

Although double-sided prototype PCBs have never been cheaper, multilayer PCBs are still prohibitively expensive. You can expect to pay 10–15 times as much for a four-layer PCB. Many of the smaller fab houses don't even offer this service.

So, how can you have your cake and eat it, too? Let's start with a look at a simple four-layer board and how it might be constructed.

The layers from top to bottom consist of an outside trace layer, inside power plane, inside ground plane, and outside trace layer. The design could be through hole or SMT.

Through-hole technology entails component leads piercing through each layer. If the component does not connect to power or ground, there must be an annular ring (of no copper) around the component's lead on that particular layer, insulating it from the plane.

SMT technology eliminates the need for component holes. SMT designs only require a via (i.e., an unsupported hole] when the component's lead must connect to a component or plane on a different layer.

These vias need not pierce all layers. This setup can lead to some interesting drill files where each pair of layers may have a different drill pattern.

An edge view of the four-layer board would show alternating layers of copper and fiberglass. That's four layers of copper and three layers of fiberglass.

So, you'd need three layers of PCB material-either a double-sided PCB with two single-sided PCBs laminated to the top and bottom, or a bare (no copper) PCB with two double-sided PCBs laminated to the top and bottom. Although the second example is more common, you should include a fabrication drawing so the fab house doesn't have to guess on the makeup.

Why am I discussing multilayer PCBs when the inexpensive houses don't do more than double sided? If you take another look at the second example of the 4-layer construction, you might get what I'm hinting at.

If you take your four-layer design and prototype the top layer and power plane on a double-sided PCB and ground plane and bottom layer on a second double-sided PCB, you'll end up with an inexpensive four-layer PCB. The only drawbacks are that you must add a center layer of insulation between the two PCBs and make a multitude of interconnections.

As you can see in Photo 3, the black plastic carrier of a square-pin sip header spaces two boards about 0.1" apart. In cases where thickness is not a problem, the air space is plenty wide enough to ensure that the inner layers don't short to one another.

Since this design is totally SMT on both outer layers except for the header, all the vias need to be soldered using pieces of solid wire and clipped off flush with the outer layers. There are many materials (besides air) that you can use as to insulate between the inner surfaces (i.e., the power and ground planes).

You want to choose a material which doesn't melt easily but which punctures without much effort. A good source of this stuff is the material on which some bacon is packaged. It enables the boards to be sandwiched rather compactly. Once all the interconnects are soldered, the SMT parts can be added.

Although a printed copy of the photoplot files works well, I like to have at least two prototypes. One is for assembly, and the second is to see where the traces go once the first prototype is assembled and I need to refer to the inner layers or underneath a component.

BEER-BOTTLE BUDGET

In these times of bigger, faster, more complex computers, it's nice to know that smaller, fast, less complex designs have a chance of making it in our world.

In fact, if you look around at the consumer items being purchased today, you'll see technology creeping into every appliance. None are controlled by high-power PCs but much smaller micros.

Many times, micros are added to help products conserve energy or provide a level of safety. These designs start with people like you and me. Now, we have access to professional PCBs without the big bucks normally associated with the real thing.

I'd like to thank these companies for their service in giving the little guy a chance. Try 'em. You'll like 'em.

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@circuitcellar.com.

SOURCES

Mail-order prototype boards
AP Circuits
3112 40th Ave. NE
Calgary, AB
Canada T2E 5T8
(403) 250-3406
Fax: (403) 250-3465
staff@apcircuits.com
www.apcircuits.com

Capital Electra-Circuits, Inc.
7845-J Airpark Rd.
Gaithersburg, MD 20879
(301) 977-0303
Fax: (301) 990-6715
sales@capitalelectro.com
www.capitalelectro.com

EP Circuits
5468 Highroad Crescent
Chilliwack, BC
Canada V2R 3Y1
(604) 824-1238
Fax: (604) 858-7663
epproto@uniserve.com
www.uniserve.com/epcircuits

Imagepro Circuits, Inc.
4372 Dawson St.
Burnaby, BC
Canada V5C 4B6
(604) 294-332 1
Fax: (604) 294-3302
billchan@istar.ca

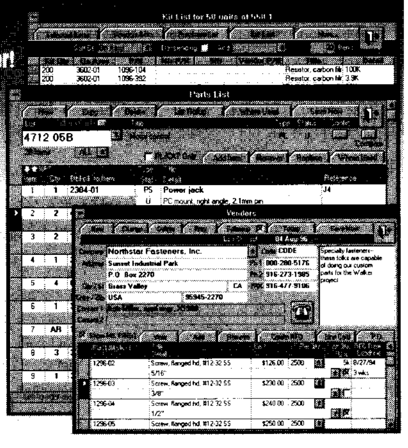
I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

Parts List Manager

For Engineers, Product Designers, Prototypers

Windows-based software helps you stay organized ...And makes the job easier!



Easily create and manage multi-assembly parts lists for products in development ...And after!

Keep track of:
- **Part Specs & Dwgs**
- **Suppliers & Mfrs**
- **Boms and Kit Lists**
- **Product Costs**
- **Engineering Stock**

Parts & Vendors™ Version 2.0

For independent and departmental projects. Use alone or in a workgroup

Version SE: $99 + s/h EXtended: $299 + s/h

Call 800-280-5176

Trilogy DESIGN
916-273-1985
fax 916-477-9106
P.O. Box 2270, Grass Valley, CA 95945
http://www.trilogydesign.com

Requires 486, 10 meg min. ram. Win 3.x or Win95

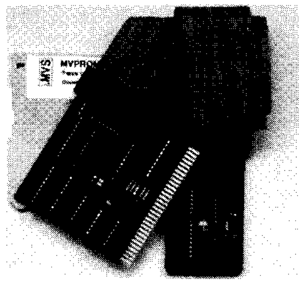
386 SBC $83

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
- IRQO-15 (8259 X2)
- 8237 DMA 8253 TMR
- BUILT-IN LED DISP.
- UP TO 8 MEG ROM
- CMOS NVRAM

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

$95 SINGLE PIECE PRICE

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



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

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

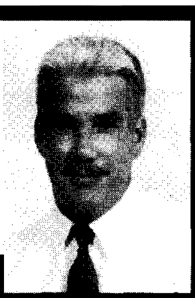
MVS
MVS BOX 850
MERRIMACK, NH
(508) 792 9507

#135

SILICON UPDATE

Tom **Cantrell**

A(r)Ray of Analog Hope



Although Tom trembles before the analog god, he regrets the losses that digital brings. Seeking analog fidelity and digital ease, he finds his Mecca in Motorola's programmable analog array.

I've often lamented the fact our creator failed to make the real-world TTL compatible. Oh well, nothing to test your faith like trying to get some balky op-amp lashup to deliver a little more signal and less noise.

When it comes to analog, I often feel like one of those meek citizens of the future held hostage by machines meant to serve but which turn on their users when no longer held in check by a dwindling priesthood of "old ones."

In fact, it seems much of the trend to replace analog with digital techniques, theoretical advantages of the latter aside, can be traced to the fact that few really understand (and fewer are being taught) the ever-more-forgotten art of analog design. Whether it's software modems or MPEG decoders, users would rather watch a variable and change a line of code than fiddle with a scope and fumble around with resistors and caps.

Is it the end of the road for op-amps? Will all signal processing devolve into software? Or, is there perhaps a middle ground that combines the natural fidelity of analog circuits with the easy design and debug of digital?

What might the latter look like? Check out the Motorola Programmable Analog Array to get a glimpse of the possible digilog future.

DIGITAL DÉJÀ VU

The MPAA020 Field Programmable Analog Array isn't the first chip to combine the digital PLD/FPGA concept with analog circuits. Credit for that notable first probably goes to the IMP EPAC 50E10 (see "EPAC Epoch," *INK 58)* though, much as the famous PAL was predated by ROMs and PLAs, earlier providers of simple metal maskable linear arrays could also claim credit.

However, the MPAA arguably goes further in mimicking its digital counterparts. Photo 1 gives you a dieshot view.

The chip comprises a 4 x 5 array of so-called CABs (Configurable Analog Blocks), which are conceptual counterparts of Xilinx's digital CLBs (Configurable Logic Blocks).

The similarity continues with provision on both chips for block interconnection using a combination of local global routing that accommodates the tradeoff between wiring flexibility, speed, and cost. On the MPAA, the vertical lines represent local wires that any adjacent block can connect to, while the horizontal "long lines" span the entire chip.

Most PLDs and FPGAs supplement their basic function blocks with robust, flexible I/O drivers. The MPAA follows suit with thirteen of them on the periphery of the chip.

So far, I think you can see the similarity between the MPAA and digital PLDs and FPGAs. Just replace the latter's digital logic and I/O blocks with analog variants, and you've got the gist of it. It's not surprising to find that the MPAA configuration scheme, relying on SRAM (6864 bits to be exact) to establish the cell function and interconnect, is quite similar as well.

TAKE A CAB

Question: If a digital PLD/FPGA relies on gates as the basis for block functionality, what should an analog version use?

The answer is, of course, that analog stalwart-the op-amp (see Figure 1). Search the dark and dusty recesses of your bookcase to remind yourself how the components in the input and feedback path determine the gain. (I found Howard M. Berlin's *Design Of Op-Amp Circuits.)*

By configuring and combining op-amps in different ways, just about every interesting analog circuit can be concocted, including filters (high, low, band pass, and notch), function generators (sine, triangle, square, ramp), peak detectors, rectifiers (half- and full-wave), VCOs, constant-current sources, and so on.

Conventional op-amp wisdom calls for resistors in the feedback path. Instead, the MPAA relies on switched capacitors (five per cell or 100 total per chip) whose static capacitance (255 levels) and switching are both controlled digitally. When configured and switched appropriately, these capacitors act as resistors and fulfill the role of tuning the op-amp.

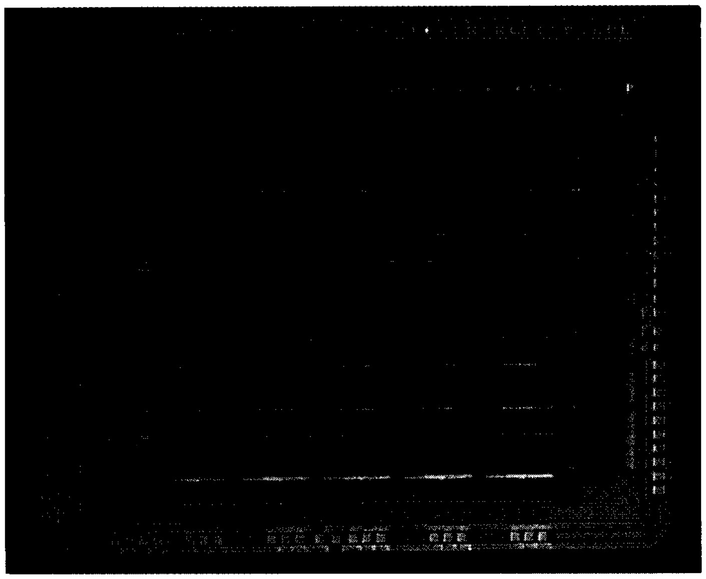


Photo 1 —*Much of the MPAA die area is consumed by wiring that programmably interconnects the 20 CABs, which are laid out in a pair of 2 x 5 strips.*

One interesting implementation challenge that was overcome (according to Motorola) is how to establish a particular capacitor's value in spite of the variability imposed by the programmable interconnect. As you'll see, the MPAA development tools take care of it automatically, so you don't have go through a tedious recalc every time the design changes.

Typical analog circuits call for more than just op-amps. For instance, any comparison or limiting functions need a reference against which to compare or limit.

The MPAA offers a number of capabilities in this regard, including ±2.5 and 0 V, as well as an 8-bit programmable reference. There's also an option to provide an external reference input. For utmost versatility, the references are made available (i.e., routable) to each of the 20 CABs.

It might seem odd, but it's a fact that all but the simplest analog circuits also call for that most digital of assets—a clock. After all, that's where the "switch" in switched-capacitor circuits comes from.

Further, clocking is critical for sampled-data circuits, particularly filters, in which clock rate is a fundamental factor of merit. The well-known Nyquist limit specifically states the sampling rate must be at least twice the highest frequency encountered in the sampled data to avoid misleading results due to aliasing.

However, it's often overlooked that the 2x factor is impractically optimistic as it needs a completely noise-free input. Add the real-world's complement of glitches, drift, offset, and quantization errors, and you'll find the sampling rate should actually be much higher.

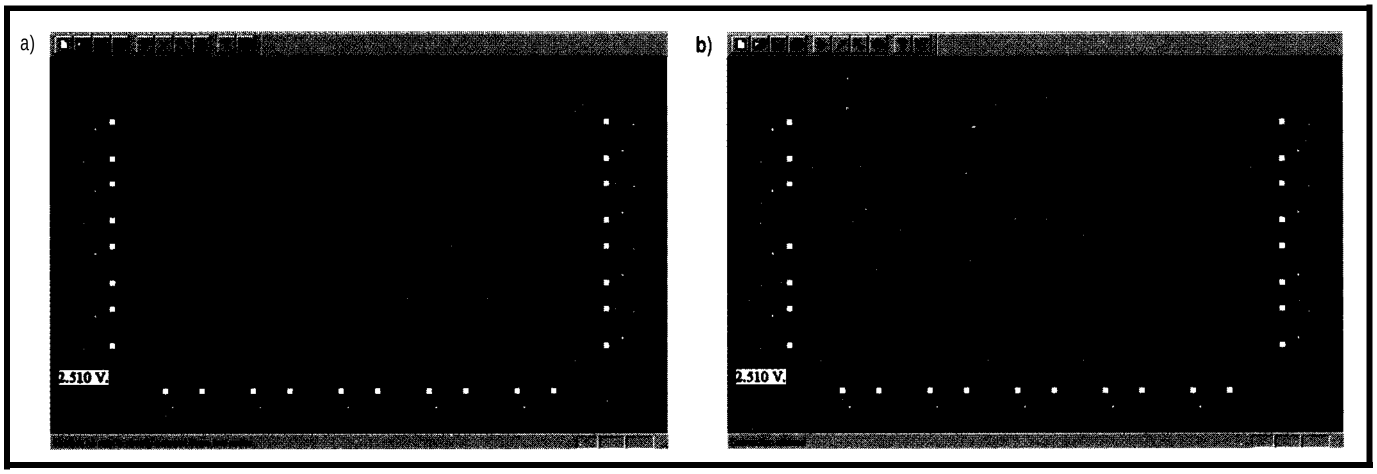
The MPAA generates an internal 1 -MHz clock and, as with reference voltages, also accommodates an external input. Once on-board, this fundamental timebase goes through four independent programmable 5-bit dividers, and the various clocks are then made available to the CABs.

PINS APLENTY

With conventional quad op-amps available in 14-pin packages and given that many of the connections are likely confined inside the MPAA, you might guess something along the lines of a 68- or 84-pin package would do the trick.

Thus, you-like I-might be a little surprised to see the chip end up with 160 pins. But, whittle them down a bit, and you'll find the chip isn't as imposing as the package implies.

First, there are a few dozen No Connects, perhaps reserved for future expanded parts, proprietary test access, or just because there was a 160-pin fire sale. Also, nobody with any analog experience will complain about plenty of provision (a dozen lines or so) for



Photos 2a & *b-Connecting wires is as simple as clicking the mouse. In this pair of photos, notice how the EasyAnalog software doesn't allow an invalid connection to be made.*

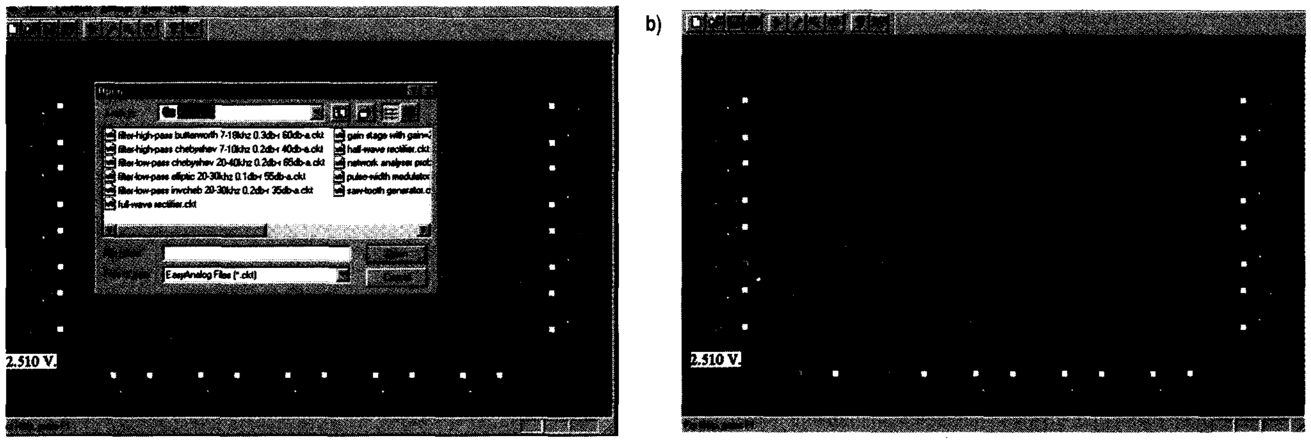

filter-high-pass butterworth 7-18khz 0.3db-r 60db-a.ckt
filter-high-pass chebyshev 7-10khz 0.2db-r 40db-a.ckt
filter-low-pass chebyshev 20-40khz 0.2db-r 65db-a.ckt
filter-low-pass eliptic 20-30khz 0.1db-r 55db-a.ckt
filter-low-pass invcheb 20-30khz 0.2db-r 35db-a.ckt
full-wave rectifier.ckt
gain stage with gain
half-wave rectifier.ckt
network analyser prob
pulse-width modulator
saw-tooth generator.c
EasyAnalog Files (*.ckt)
2.510 V.
b)
2.510 V.

Photo 3-*Beyond the one or two CAB macros, the* ***EasyAnalog software also*** *includes a library of even more complex cells (a), such as a* ***PWM*** *(b).*

the separate analog and digital power supplies.

Another major pin sink is the configuration logic. Remember, since the MPAA is SRAM based, it's brain dead after powerup until all the SRAM bits are set.

For the greatest flexibility, Motorola offers a variety of ways to get inside the MPAA and init the SRAM. The particular approach used is determined by strapping some mode pins.

As is usual for SRAM-based logic chips, a serial EPROM works well. The MPAA supports both those that require an external address as well as those with an internal-address generator.

However, Motorola goes even further by supporting generic byte-wide (EP)ROMs, though at the cost of 8 data and 18 address lines. Why the chip includes so many address lines isn't exactly clear from the datasheet. Perhaps another expansion plan down the road?

In any case, it's simply a matter of connecting up the address and data lines, along with provided control (e.g., *CE, ● RD, etc.) lines, and you're in business. Naturally, there's also a RESET line and the equivalent of a sleep input that forces the op-amp array into powerdown mode.

Ignoring all the baggage, you'll find the guts of the MPAA boils down to three lines for each of the previously mentioned 13 I/O cells and three lines for each of an extra 8 uncommitted op-amps. The latter are available for any application-specific use, but the external wiring and resistors and so forth are up to you.

The I/O cells themselves are simple enough in concept, consisting of an op-amp, programmable switch, and three lines. As you can see in Figure 2, between enabling and disabling the switch connecting terminals x and y and the op-amp, the I/O cells can be configured as either input or output with or without the op-amp buffering.

CONNECT THE DOTS

Replacing wires and discretes with bits is fine, but only with good tools. Only the most hard-core designer would relish hand assembling the 6 Kb, presuming they had the internal SRAM address map, which wasn't included in the documentation I perused.

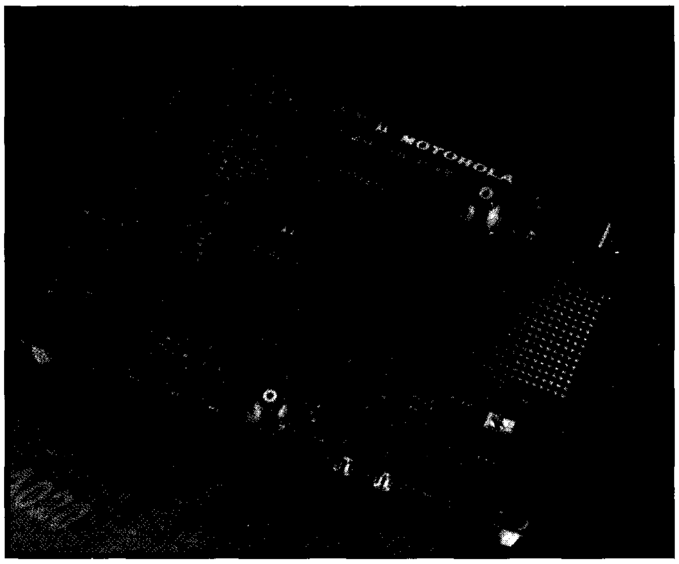


Photo 4-*The* ***MPAA3BRD*** *Evaluation Board offers the opportunity to* **gel** your *hands—and scope-on an MPAA.*

Fortunately, Motorola offers handy development software known as Easy-Analog. As the name implies, it goes a long way towards hiding the gory details. Best of all, the price is right ($0), and your own copy is only a download from the Web page away.

Thankfully, the software avoids the bloatware tendencies and runs handily on any '486, 4-MB RAM, VGA, serial-port box. I must sheepishly admit the fact that it requires Windows 95 did set me back. I'm hoping to ride my old '386 Windows 3.1 box all the way to Windows 98.

As you see from my photo, I have enough gray hair already. Too many more upgrade exercises, and I'll be hitting the Grecian Formula.

Fortunately, there's no shortage of bleeding-edge types here in Silicon Valley, including my old friend Phil. He's involved in some heavy-duty I-way projects (not sure exactly what, but mainly it seems to involve "reinstalling NT") and has plenty of the newest iron. Thanks to Phil for helping me check out the software (and Phil thanks Motorola for including an uninstall).

EasyAnalog starts with the basics-the ability to establish the basic parameters for CABs, I/O cells, clocks, and so on, and manually connect wires. Working at this level is still the province of analog gurus, though the software does help a bit.

For instance, as shown in Photos 2a and 2b, EasyAnalog won't let you make an invalid connection or, similarly, specify a cell characteristic that isn't possible. You can't produce an invalid configuration, though that doesn't mean your valid configuration will do what you want.

As far as I, an analog neophyte, am concerned, the real benefit of EasyAnalog is the design expertise embodied in the supplied macros (see Table 1). While they do consume one or two CABs each, they also compose a rather complete catalog of high-level functions. As Photo 3a shows, these macros can be further combined into cells (e.g., the PWM in Photo 3b) included with the software.

Once your design is entered, EasyAnalog saves the corresponding SRAM bit file in a variety of formats (i.e., Intel and Motorola hex, forward, and reversed). One of these should be decipherable by your EPROM programmer.

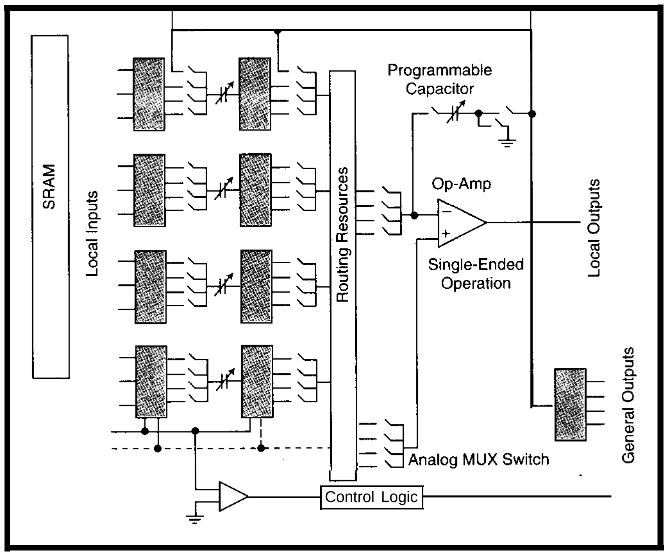


Figure 1--Each *CAB is centered around an op-amp with SRAM-defined characteristics and interconnects*

capacitors serve to tune the CAB's function (e.g., setting

SIM OR EM?

I was disappointed that EasyAnalog doesn't include a simulator. Instead, instant gratification calls for an evaluation board (see Photo 4) as well as the means and will to cough up $1500.

The board basically provides a versatile living space for the MPAA with plenty of jumpers, breadboard area, and niceties such as voltage regulator, external clock, and voltage reference—not to mention the switches and LEDs.

Along with the board comes a pod that sits between it and your PC serial port. In essence, the pod fulfills the role of an EPROM emulator, receiving the hex file of your design and impersonating the boot device the MPAA expects to find on RESET. Interestingly, the pod doesn't actually use a memory chip but instead coerces an 'HC11 MCU into impersonating one.

Motorola's idea is the easiest way to study the MPAA. Download a design (only takes a few seconds even with a lowly RS-232 port), hook up a function generator and scope, and have at it.

Pondering the situation a bit, I realized the digital and analog worlds are unequally served by the emulator and simulator concepts. For a digital PLD or FPGA, a simulated 5-V system is accurate even if the actual design ends up running at 4.8 V with a bit of noise thrown in for good measure. A 1 is still a 1, and likewise for 0.

By contrast, it's quite easy for an analog simulator to get out of sync without complete knowledge of a particular system's wiring, component tolerances, temperature, noise, and the like. In fact, Motorola points out you'll likely to have to go back and tweak various parameters (e.g., gains, offsets, references, etc.) once the actual board is designed and the MPAA can be exercised in place.

THE YOLK'S ON ME?

In the earlier article on the EPAC, I put forth the proposition that "though I might end up with egg on my face," the digilog revolution, with analog equivalents of PLDs and FPGA as the foot soldiers, was underway.

So far, it seems things are lagging a bit. Despite the IMP EPAC and now the Motorola MPAA (along with a lot of mixed-signal hype and hope), open today's electronic gadget and you're likely to find an analog section with a sea of op-amps and RCs little changed conceptually from 20 years ago.

To be frank, Motorola's $50 price tag for the MPAA doesn't exactly prod things along. Yes, the whole concept is neat, but a couple dozen op-amps and bag of RCs cost a lot less. Perhaps many are willing to pay 1.5–2× for the convenience, integration, and whizziness, but 5–10× is tough to swallow.

Nevertheless, I remain hopeful, mainly because I remember the similar "it costs more than 74xx" complaint levied against the suppliers of the early-and very pricey at the time—digital PLD and FPGA chips. Of course, the naysayers have long since gotten

Macro Functions	
Single-Cell	Two-Cell
Gain stages	Low Q Biquad Filter
(max=20; min=1/20)	(Hi or Lo Pass) – 2 cells
Sum & Diff Amps	Low Q Biquad Filter
(3 weight inputs)	(Notch, Bandpass) – 3 cells
Sample & Hold	High Q Biquad Filter
	(Hi or Lo Pass) – 2 cells
Track & Hold	High Q Biquad Filter
	(Notch, Bandpass) – 3 cells
Integrator	Maximum Corner Frequency = T/10
n-Path Integrator	Minimum Corner Frequency = T/100
Differentiator	Limiter
Half-wave Rectifer	Interpolator
Full-wave Rectifier	Schmitt Trigger
LPF Rectifier	Voltage-Controlled Oscillator
Cosine Filter	Sine-wave Oscillator
Decimator	Square-wave Oscillator
Bilinear Filter	Triangle-wave Oscillator

Table *1-Beyond simple place and route, much of the value of the EasyAnalog software is incorporated in the library of predefined macros.*

RS-485 NOW!

Standard RS-485 Network Software

Deploy a rugged, proven Cimetrics RS-485 solution in just days. Software, full documentation, development tools. All backed by outstanding service and support from Cimetrics.

Comprehensive Microcontroller Master & Slave Support

PC (DOS/Win/NT)	68HC05	PIC16C74
8051	68HC11	PIC17C42
8096	68HC16	H8/300H
80C196	68302	TMS370
80C186EB	68332	Z180
	68333	

- Process Control
- HVAC & Lighting Control
- Point-of-sale
- Transportation
- Fire & Security
- Telecommunication
- Medical Equipment
- Laboratory Data Acquisition

Cimetrics TECHNOLOGY

55 Temple Place • Boston, MA 02111-1300
Tel: 617-350-7550 • Fax: 617-350-7552
Website: www.cimetrics.com

#141

The Intelligence of Miniaturization

General Software is the pioneer of embedded systems. Since 1989 our software has been the core intelligence of miniaturization. From cellular phones to the space shuttle, General Software enables dreams. Our new Embedded BIOS" and Embedded DOS" systems will take your projects farther faster than ever before.

If it's time for a new design start, it's time for General Software.

GENERAL™ SOFTWARE

Call for information on Embedded Systems: 800-850-5755

We Start the World

Please visit our web site: www.gensw.com

#142

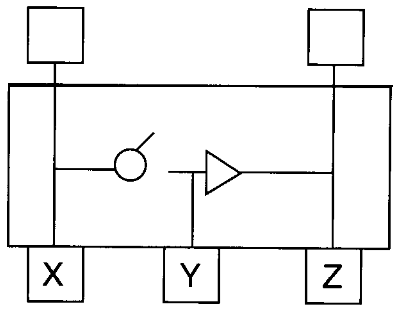


Figure 2—*The* 13 *I/O* cells comprise *three* pins, a *programmable switch, and yet another op-amp that allows buffering signals headed on- or off-chip.*

religion and, along the way, propelled outfits like Xilinx, Altera, Lattice, and their kin to great heights.

For now, just remember, "Don't put all your eggs in one basket because you don't know whether the chicken or egg is coming first and you shouldn't count your chickens before they hatch."

By the time you figure out what the heck that all means, I hope I can point to an accelerating digilog revolution. (Motorola's already talking about next-generation MPAAs.) Otherwise, all I can say is that I like 'em over easy. ❑

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. You may reach him by E-mail at tom.cantrell@circuitcellar.corn, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

REFERENCE

H.M. Berlin, *Design Of Op-Amp Circuits,* Howard W. Sams and Co., Indianapolis, IN, 1977.

SOURCE

MDEL612, EasyAnalog, MPAA020 FPAA
Motorola
2100 E. Elliott Rd.
Tempe, AZ 85284
(602) 413-4663
Fax: (602) 413-4034
www.mot-sps.com/fpaa

I R S

431 Very Useful
432 Moderately Useful
433 Not Useful

PRIORITY INTERRUPT

The People's Contest

I mentioned that we did a reader survey a couple months ago. It's something we should do more often, but given the hectic pace around here, tasks like these are as much a luxury as they are a necessity. Unlike an advertising focused survey-you know, the ones that ask whether you prefer Brand A or Brand B products-this was purely an editorial questionnaire. We asked things like what kind of processor you like and what technical subjects you'd like to see covered in *INK*. We had a 50% survey response from our fantastically loyal audience.

One of the more interesting conclusions from the survey is that, while you have particular processor preferences, you appreciate the diversity of selections available when designing embedded controls. Today's technology has provided us with a plethora of processor and cost options that must be duly considered (gone are the good old days when every embedded control problem could be solved with 64k on an 8-bit processor).

Part of the survey asked what processors you've used, are using, or expect to use. Not surprisingly, the big winner in past applications was the Z80, 8*x*51, and 68HC*xx*. Present applications seem to be evenly divided among 80*x*86, 68*xxx*, 8*x*51, and PICs. For the future, however, the two winners are 80x86 (including Pentiums) and PICs!

I admit right up front that I'm from the old school. I look at solving embedded control problems with a scalpel rather than a sledgehammer. Being a realist, however, I know that many applications are solved best with ColdFire and MMX technology. That practical realization is the rationale behind our *Embedded PC* section each month, and it was the impetus for our *Embedded PC* Design contest.

The wide disparity in processing power between PICs and Pentiums in future applications is not a simple issue of choice. It's an exercise in practical engineering. Not all problems are equal, and certainly no one processing solution is adequate. The *Embedded PC* Design Contest focused on demonstrating high-end problems and PC-based solutions. The response was significant, and you'll soon see a number of these entries as *INK* articles.

With that concluded, we determined that our next contest should test the prowess of designers at the decidedly less expensive end of the embedded solution spectrum. It only took a few E-mail messages back and forth for Microchip Technology to agree to sponsor the whole contest. Certainly we had evidence of PIC popularity among the readership, but it was only coincidental and decidedly after the fact that our survey results bear out the veracity in our choice of sponsors.

This month, Circuit *Cellar INK* announces Design98. Sponsored by Microchip Technology in association with Hewlett-Packard, Bell Electronics, and Pioneer Electronics, Design98 gives you an opportunity to show the rest of us what can be really done with PIC processors.

Winning is a function of finesse and not complexity. Projects can range from the very simple to the most intricate. The deadline is March 1, 1998, and we welcome entries that run the gamut. There are $23,000 in cash prizes alone. The contestant awarded Best Overall will receive $5000 and a HP546D Mixed Signal Oscilloscope. There are three design categories—PIC12C*xxx*, '16C*xxx*, and '17C*xxx*—with first ($3000) second ($2000) and third ($1000) prizes awarded for each category. Plus there are cash bonuses if you design in some specified components. Visit our Web site at www.circuitcellar.com for contest rules and an entry form.

As the local lottery ads say, you have to play to win. And fortunately, design contests make us all winners in the end. Our survey ratifies the popularity of the PIC, and Design98 provides incentive to document really neat applications. The real reward will be when we read about the winning projects in INK.

Steve

steve.ciarcia@circuitcellar.com