EMBEDDED PC SPECIAL BONUS SECTION: EMBEDDED PC

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#71 JUNE 1996

DATA ACQUISITION

Analog Conversion Techniques Using Zilog's Embedded Controllers

Battery Load/Charger Analyzer

Dealing with EMI

EPC: Driving Multiaxis Stepper Motors

0 74470 75349 0 06

$3.95 U.S.
$4.95 Canada

TASK MANAGER

Good Data, Bad Results

All right. Show of hands. How many of you are guilty of collecting all manner of trivial information just in case it might come in handy, only to never get around to using it? Take a look at that bookmark list in your Web browser. How many of those sites have you revisited in the past month?

What about that bookshelf or file cabinet behind you? How many of those materials have you used lately, and how many will you likely never use again?

In this information age, we collect data on everything. In most cases, however, gathering the raw data is the easy part. What you do with it is the issue. And it's not an easy issue to resolve. Too much information can be worse than not enough when the meaningful parts are lost in the noise. Improper interpretation of the data-maliciously or not-has to be the worst.

Society today is too quick to accept conclusions made by "researchers" at face value without taking the time to explore the underlying tenets. A closer look may reveal that the data does not support the results.

So when you're busy acquiring data, no matter what the application, take care what you ultimately do with it.

THIS MONTH

We kick off this years Data Acquisition issue by revisiting an old favorite—the Zilog Z8—and looking at some techniques for doing A/D conversions with the latest in this expanding family.

Next, we tap one of 1995's Design Contest winners for all the details on his battery load/charger analyzer and get a lesson in battery chemistry in the process.

Moving into the realm of RF, we next explore what can be done to check for and reduce EMI in your latest design before getting to that very expensive testing lab.

In our final feature, we look at the newest in serial EEPROMs to see how they try to be everything to everybody.

In this month's Embedded PC, we start with a look at a Windows-like embeddable operating system: WinLight. Next, we cover a new pair of chips that make embedding an Intel '386EX processor much easier. In PC/104 Quarter, we check out the use of load cells in PC/104 setups. And last, we're shown how easy it is to use today's off-the-shelf components to piece together a stepper-motor controller.

In our columns, Ed continues with his look at 60x66 performance issues, Jeff falls back on the KISS principle to create a simple printer switch, and Tom overviews the latest in Dallas Semiconductor's Touch Memory line of products.

editor@circellar.com

CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

MANAGING EDITOR
Janice Marinelli

TECHNICAL EDITOR
Elizabeth Laurencot

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Fred Eady

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK-, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank.

Direct subscription orders and subscription related questions to 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 photography by Barbara Swenson

PRINTED IN THE UNITED STATES

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (860) 871-1988; 2400/9600 bps Courier HST, (860) 871-0549. World Wide Web: http://www.circellar.com/

EMBEDDED PC

See Pages 63–95 for our Special Bonus Section

INSIDE ISSUE 71

READER I/O

IF PROGRAMMERS ONLY...

I couldn't agree with something more than I do "The Old Curmudgeon" (*INK* 68). Do you remember when the first '486-class machines were introduced-how they showed that they truly performed at the benchmarks of the Cray 1?

It's a sad reflection on the programmers and software developers of today. Hard to believe it takes a Cray 1 to write the letters and do the simple spreadsheets I did on an 8086. Too bad some software house doesn't take its industry to task.

Imagine the kind of performance we'd see if the "lean-mean" coding philosophy existed today. We'd have real-time multitasking. I'd be able to do three or four things at once. My computer would still be useful for more than a fishing weight after three years.

I enjoy your comments. I've been a subscriber to ***INK*** since you left ***BYTE.*** Keep up the good work.

Kim R. Rogers
krogers@telestar.com

WITHOUT DOS, I'M DEAD

Steve's "The Old Curmudgeon" (*INK 68)* sure made me smile.

I'm still afraid of Windows. It's a terrible housekeeper and does many things beyond my control. Whenever I install software, it's there to stay. Uninstaller or Remove_It are of little help when the software overwrites some DLLs with another version.

So far, the only solution that works is a batch file to `xcopy c:\windows d:\win /s/eandthenbackagain` after I've done the latest demo of something I don't like.

Just imagine-without DOS, I wouldn't be able to do that.

Dusan Benko
New York, NY

PROGRAMMING THE IDEAL FRENCH FRY

I always look forward to ***Circuit Cellar INK.*** As an engineer and an HCS II home-automation fan, the "They Still Flip Hamburgers, Don't They" ***(INK 64)*** tagline piqued my interest.

My company, Tridelta Industries, designs and manufactures embedded-controller modules for commercial cooking equipment. When you visited the Restaurant Equipment Manufacturer's convention, I think you missed some of the controls you hoped to find. They are hiding out inside cooking appliances.

In some of our controls, simple low-cost temperature and time controls use analog circuitry. However, on the mid- to top-of-the-line products, we produce sophisticated computer-based controls.

These controls include 6–8-character alphanumeric displays, membrane keypads, thermistor sensing, programmability, remote communications, and so on. But, the heart of the product is the combination of cooking technology with electronics in a pretty hostile environment.

We need to study and implement the optimum "cooking curve" to make ideal french fries, nuggets, hash browns, and chicken patties (3 oz. to 4 lbs. of rock-solid frozen mass in a deep-fat fryer). And, safety systems must be inbred to the control and product FMEA to deal with 350" hot fat, boiling water spatter, fire flash points, and constantly changing kitchen help.

I'm preaching to the converted here about reasons to apply computer-control technology. But, I think your readers might be surprised at the level of electronics at work in restaurants-especially in quick-service stores.

Bob Weinberg
bobw@tridelta.com

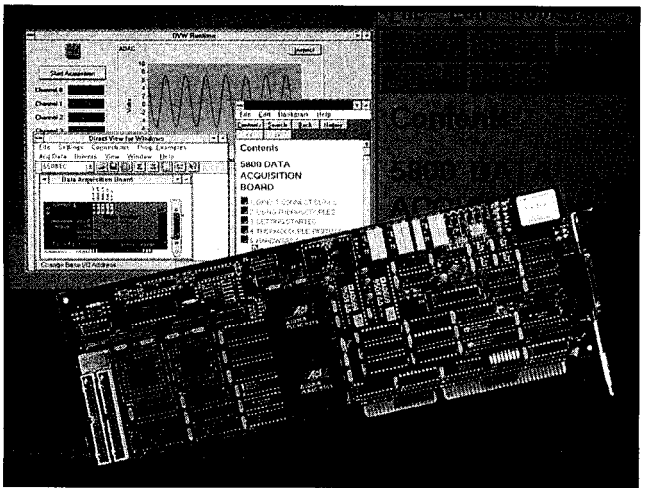
Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial offices at (860) 875-2199.
Fax: All faxes may be sent to (860) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300–14.4k bps, 8N1).
Internet: Letters to the editor may be sent to **editor@circellar.** corn. Send new subscription orders, renewals, and address changes to **subscribe@circellar.com**. Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article. For more information, send E-mail to **info@circellar.com**.
WWW: Point your browser to **http://www.circellar.com/**.
FTP: Files are available at **ftp://ftp.circellar.com/pub/circellar/**.

NEW PRODUCT NEWS

Edited by Harv Weiner



HIGH-SPEED DATA ACQUISITION BOARD

ADAC's 5803HR is a complete hardware and software product optimized for Windows. The board incorporates 16-bit analog inputs and outputs and includes Windows-specific hardware and software features.

The 5803HR includes a 1024-word FIFO, a hardware necessity for data acquisition under Windows. While the PC's ISR executes during an application, the 5803HR temporarily stores data in the onboard FIFO, so there's time for the DMA controller and hardware device registers on the board to be reprogrammed. Without the FIFO, data sampled during reprogramming would be lost.

Windows' handling of interrupts further complicates an application's call for different gains (or input ranges) selected on a per-channel basis. The 5803HR's onboard channel-gain list (stored in RAM) has all gain switches occur in the background. Any other method of providing a channel-gain list requires software intervention, severely limiting the acquisition rate.

The board includes ADAC's Direct View for Windows (DVW) board-tutorial and data-acquisition software. This interactive program guides the user through board setup, wiring connections, and board operation. After setup, DVW performs high-speed data collection, display, and streaming to disk without any programming. Its interactive help includes the complete 5803HR manual.

The 5803HR features gains of 1, 2, 4, and 8 and sells for $995. The 5804HR features gains of 1, 10, 100, and 500 and sells for $1095. Optional thermocouple panels provide 16 thermocouple inputs, and a low-cost, multiplexing panel expands the number of ADC inputs to 64.

ADAC
70 Tower Off ice Park • Woburn, MA 01801
(617) 935-3200 • Fax: (617) 938-6553
http://www.adac.com/ **#500**

SERIAL DATA ACQUISITION

The SoftwareWedge from TAL Technologies lets users direct serial data from any instrument or device into any Windows, DOS, NT, or Win95 application. The Windows version, WinWedge 32, features 32-bit processing power for directing data into other 32-bit applications. Data from process-control and lab instruments, gauges, sensors, PLCs, and analyzers can be input in real time into packages like Excel, Lotus, Access, and Quattro, as well as statistical and industrial software.

WinWedge 32 provides faster data acquisition and instrument control with native 32-bit processing power with Windows 95 and NT applications. Its features include support for up to 99 serial ports at the same time, baud rates up to 56 kbps, and the ability to have multiple applications open and receiving data simultaneously from several sources. It offers fully preemptive multitasking for faster data throughout.

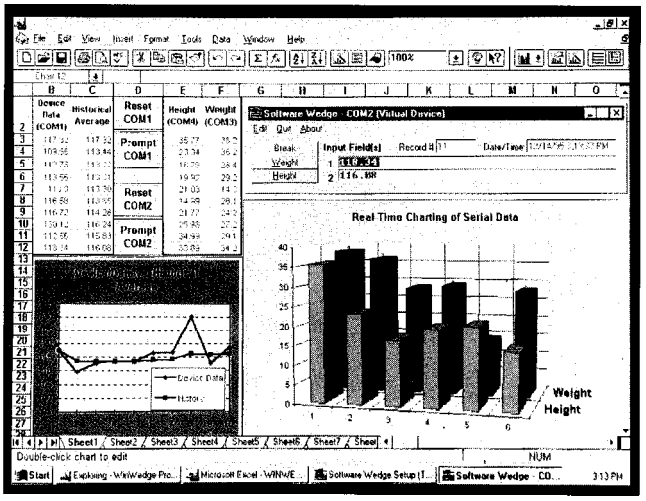
WinWedge 32 is useful for real-time analysis, charting, and graphing of serial data. It can run in the background, inputting data into one or several applications while other tasks are performed. Different instruments can send data simultaneously to different applications or to several fields within one application.

WinWedge 32 is extremely versatile and easy to use. It can be set up in minutes without any programming. Menu structure is highly intuitive, documentation is straightforward and easy to use, and technical support is unlimited.

WinWedge 32 sells for $495.

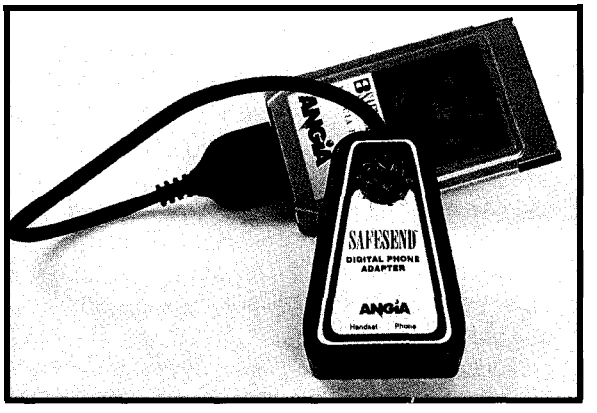
TAL Technologies, Inc.
2027 Wallace St.
Philadelphia, PA 19130
(215) 763-7900
Fax: (215) 763-9711
http://www.taltech.com/

#501



NEW PRODUCT NEWS

TELEPHONE-LINE ADAPTER



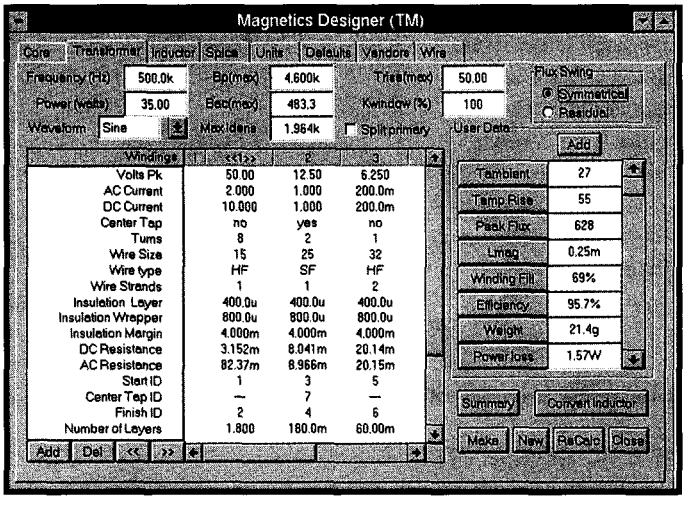
Angia Communications announces SafeSend-the first dedicated telephone line adapter to allow PC Card fax/modems to communicate over digital PBX telephones. Digital telephone lines, used by most large companies, government entities, and major hotels, deliver a higher current than standard analog lines and can damage the sensitive circuitry in most PC cards.

SafeSend requires no additional power source-such as batteries or transformers-to adapt the voltage for transmission over digital lines. A manual switch on the adapter enables the user to select the correct impedance match necessary for PBX telephone systems like Rolm, AT&T, NEC, Northern Telecom, and others.

SafeSend sells for $129 and is fully guaranteed under a lifetime warranty. Users receive lifetime, toll-free technical support and 24-hour BBS.

Angia Communications
441 **East Bay Blvd. • Provo, UT 84606**
(801) **371-0488 • Fax: (801) 373-9847**
http://www.angia.com/ **#502**

TRANSFORMER AND INDUCTOR DESIGN SOFTWARE



Magnetics Designer is a Windows-compatible package that facilitates the design of transformers and inductors. Design applications include high-frequency switching regulator transformers and output chokes for off-line, full-wave, and forward converters; 60-Hz single-phase line transformers; AC inductors; planar magnetics; and 400-Hz aircraft transformers. Virtually any single-phase, layer-wound inductor or transformer from 10 Hz to over 1 MHz can be synthesized.

Magnetics Designer produces a complete transformer or inductor design based on electrical specifications. An database with thousands of cores and a wide variety of materials is included. Custom core and material information-using a supplied Excel spreadsheet template-can be added. The program predicts magnetizing and leakage inductance, interwinding capacitance, peak flux density, DC winding resistance, high-frequency AC resistance, copper loss (both AC and DC), core loss, weight, temperature rise, layer fill, and window fill percentage.

The program produces a Berkeley SPICE model of the transformer or inductor that is customizable with different levels of complexity. Effects may include linear or saturable core, AC (frequency dependent) and DC resistance, core loss, leakage inductance, or interwinding capacitance. The SPICE modeling feature supports design investigation through simulation while waiting for the component to be built.

A specialized winding sheet describes how the magnetic device can be constructed. The winding sheet contains pertinent information about the materials and test specifications for your transformer or inductor. The package even lets you E-mail the winding sheet directly to the manufacturer.

A detailed applications manual with design equations and algorithms, core materials and geometries, and detailed examples is included. Virtually all of the program's documentation is online.

Magnetics Designer is priced at $1000 until June 28, 1996. After that date, the price will be $1500.

Intusoft
222 W. Sixth St., Ste. 1070 • San Pedro, CA 90731 • (310) 833-0710 • Fax: (310) 833-9658 • http://www.intusoft.com/ **#503**

NEW PRODUCT NEWS

DIGITAL I/O BOARD

Industrial Computer Source has announced the DIO48(S)I series of plug-in boards that provides users with 48 individual, optically isolated digital I/O lines. The series is particularly useful in factory automation, energy management, security systems, process monitoring, and other applications where high common-mode external voltages exist.

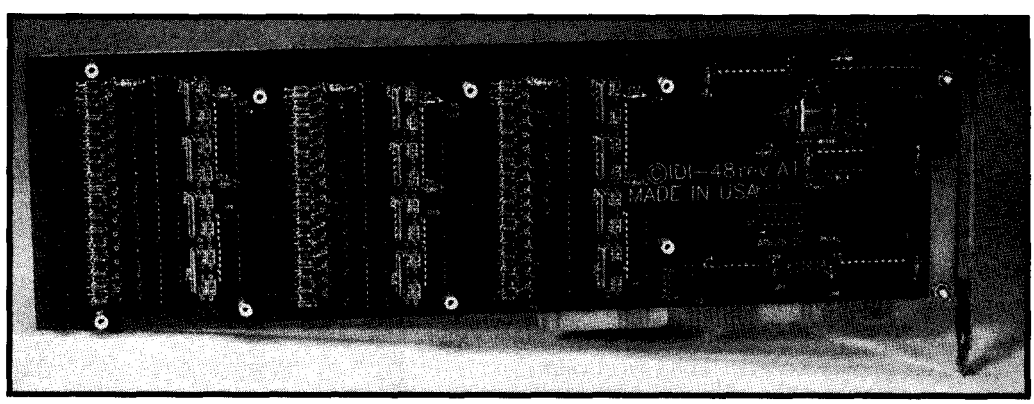
The boards provide important safety features for 48 parallel, differential input voltages up to 60 VDC, including protection circuitry to safeguard against accidentally reversing the polarity of the input connections. A choice of an enclosed or nonenclosed external termination panel is available for convenient field-wiring connections.

An onboard shield prevents the user from coming into contact with high input voltages. Isolation of 500 Vrms is provided between channels and between each channel and the host PC to protect users from accidentally contacting high voltages. Also, for added safety, alternate wires in the ribbon cables are not connected at the board. This feature provides higher bit-to-bit isolation than is available with other ribbon cables.

Four models compose the DIO48(S)I series. Boards are available for 28- or 60-V operation and may be specified with (48SI models) or without (481 models) change-of-state interrupt capability to automatically interrupt the host PC in real time per user-configured set-up parameters. DIO48SI models also provide an onboard microcontroller which allows a variety of operational modes, including a periodic scan of all active input channels to verify proper operation on a timed basis.

All DIO48(S)I series boards include set-up programs and a utility driver for Windows applications support. Prices range **$350–$485**.

Industrial Computer Source
9950 Barnes Canyon Rd. • San Diego, CA 92121
(619) 677-0877 • Fax: (619) 677-0615
http://www.industry.net/indcompsrc/ **#504**



PBASIC COMPILER

microEngineering Labs is now shipping its new PBASIC Compiler for the PIC 16C*xx* series of microcontrollers. The PBASIC Compiler takes programs written for the BASIC Stamp and converts them into PIC-compatible hex or binary files. These files can be programmed directly into a PIC microcontroller, eliminating the need for a BASIC Stamp module.

The easy-to-use BASIC language makes PIC programming available to everyone with its English-like instruction set. The benefits of the PBASIC Compiler include faster program execution than BASIC interpreters, the potential for longer programs, and substantial cost savings over a BASIC Stamp.

Along with the hex or binary file output, the PBASIC Compiler generates an intermediate assembler file which may be edited and reassembled to allow additional operations or access to other PIC registers.

While primarily intended for use with the electrically erasable PIC16C84, other PICs with more memory or larger pin counts may be substituted. The use of PICs other than the 16C84 allows for lower cost, longer programs, and access to more I/O or RAM through the use of additional assembler programming.

The PBASIC Compiler accepts the BASIC Stamp I instruction set and works with most PIC programmers. It is available for a special introductory price of $99.95. Low-cost PIC programmers are also available.

microEngineering Labs
P.O. Box 7532
Colorado Springs, CO 80933
(719) 520-5323
Fax: (719) 520-I 867

#505

NEW PRODUCT NEWS

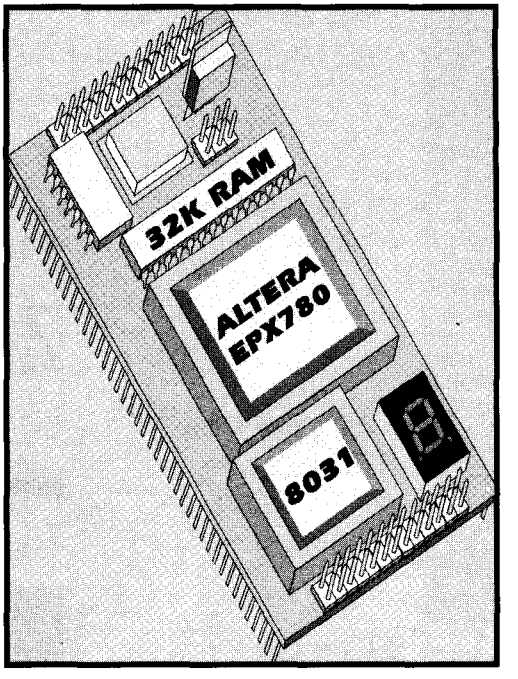
PROGRAMMABLE LOGIC MICROCONTROLLER

Engineers exploring the possibilities of combining programmable logic with a microcontroller should try the epX31 from XESS Corp. The epX3 1 combines the RAM-based epX780 CPLD with an 803 1 microcontroller and 32 KB of RAM.

During initialization, the epX780 downloads the 8031 data and programs through the PC printer port into the RAM. The 3500 gates in the epX780 are then reconfigured to support the operations of the 803 1. Customized address decoders, timers, interrupts, coprocessors, and run-time bus monitors are just a few of the possibilities. Multiple epX31 boards can be cascaded to build larger multiprocessor systems.

The epX31 kit requires a '386 DOS PC with at least 2 MB of RAM, 10 MB of disk, and a color VGA display.

The $399 Professional Edition epX3 1 kit includes a single epX3 1 board, downloading cable, programming software, run-time modules, application examples, and ***FPGA Workout-a*** textbook which shows how to design digital logic using CPLDs. Personal and student editions are also available for $249 and $165, respectively.



XESS Corp.
2608 Sweetgum Dr.
Apex, NC 27502
(919) 549-9377 • Fax: (919) 387-1302

#506

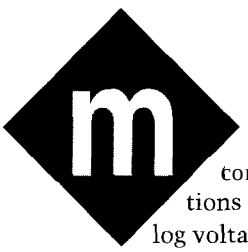
FEATURES

FEATURE ARTICLE

Don Newquist

A/D Conversion with Zilog's Z8 Microcontroller

Often applications using A/D conversion compromise accuracy, speed, or cost. Don offers four ADC configurations. He shows you how to use specific techniques to your advantage. His goal: little compromise!



Many embedded controller applications require an analog voltage to be captured. Depending on the application, a separate ADC chip may be necessary because of speed and resolution requirements. However, many designs don't need fast conversion speeds, and eight bits of resolution is adequate.

For instance, a digital thermostat samples the temperature periodically and turns the heater or air conditioner off or on when the temperature hits a trip point. Here, the speed at which it measures the voltage across a thermistor is not critical since temperature changes rather slowly. In this case, conversion times on the order of milliseconds are acceptable.

Capturing fast-changing signals, like audio, requires a much faster conversion rate. If the highest audio frequency coming into the ADC is 4 kHz, the sample rate should be at least twice this [i.e., 8 kHz). Because of the limited processing time in between samples (in this case, **125** µs), the ADC must do a conversion quickly, so the micro has time to process the data before the next sample.

Since most designs are cost-sensitive-especially in consumer electronics-there may not be the luxury of adding relatively expensive ADC chips

	Type	Pins	ROM Bytes	RAM Bytes	I/O	Speed MHz	Package
Z86C03	Masked	18	512	60	14	8	PDIP, SOIC
Z86E03	OTP	18	512	60	14	8	PDIP, SOIC
Z86C04	Masked	18	1K	124	14	8	PDIP, SOIC
Z86E04	OTP	18	1K	124	14	8	PDIP, SOIC
Z86C06	Masked	18	1K	124	14	12	PDIP, SOIC
Z86E06	OTP	18	1K	124	14	12	PDIP, SOIC
Z86C08	Masked	18	2K	124	14	12	PDIP, SOIC
Z86E08	OTP	18	2K	124	14	12	PDIP, SOIC
Z86C31	Masked	28	2K	124	24	8	PCIP, Chip, Carrier
Z86E31	OTP	28	2K	124	24	8	PDIP
Z86C30	Masked	28	4K	236	24	12	PDIP
Z86E30	OTP	28	4K	236	24	12	PDIP
Z86C40	Masked	40/44	4K	236	32	12	PDIP, PLCC, QFP
Z86E40	OTP	40/44	4K	236	32	12	PCIP, PLCC, QFP

Table 1—*In the Zilog CCP* microcontroller line, the masked ROM versions are designated by the *letter "C" (Z86C03, for example),* while *"E"* stands for OTP versions.

to the design. Design engineers must look for a more integrated solution. Zilog happens to have the solution.

INTRODUCING THE ZILOG Z8

In 1991, Zilog introduced their Consumer Controller Processor (CCP) line of microcontrollers. These devices range in package sizes from 18 to 40 pins. ROM sizes vary from 5 12 bytes to 4 KB and are available in masked and OTP configurations. All use the Z8 architecture, which can be clocked up to 12 MHz using an RC, LC, ceramic resonator, or crystal oscillator.

Two power-down modes are available: Halt and Stop. Halt freezes code execution, but leaves the timers enabled. The processor exits Halt via an external or internal interrupt.

Stop completely shuts down the chip, including the oscillator. To come out of Stop mode, you apply a positive- or negative-going edge to one of the external digital inputs or via the internal watchdog timer. This application is set up via the Stop Mode Recovery (SMR) register.

Table 1 lists the entire Zilog CCP family. Figure 1 offers a block diagram of typical Z8 CCP architecture.

The onboard dual analog comparators, along with one counter/timer, implement the ADC routines. The analog comparators are muxed with the digital inputs on port pins P31, P32, and P33, as you see in Figure 2.

The analog comparators are selected via the P3M register. The comparators share a common reference pin, P33. The input range of the comparators is 0-4 V. The input offset voltage is typically 10 mV with V_{CC} at 5 V.

The output of the comparators are examined by a TM instruction (Test under Mask) on port P3. The outputs also generate an interrupt, based on the falling or rising edge of the comparator output.

These outputs can also connect to the P34 and P37 output pins under software control. The PCON register in the extended register file controls this connection (these are not available on C04/E04 and C08/E08).

The comparators are enabled during Halt mode, but are disabled in Stop mode.

Five ADC configurations will be presented:

- a single-slope method using a PWM-generated ramp
- binary-ramp counter ADC
- successive-approximation ADC
- RC ramp-generated ADC
- voltage-frequency converter

Figure 3 shows a demo board containing the necessary circuitry for all five configurations. Jumpers on the board configure the hardware for each technique.

The software routines are designed around the Z86C04/E04, but can be adapted for the entire CCP line. The Zilog CCP emulator tested the routines. These routines are generic in nature, so should be usable with any CCP micro.

PWM-GENERATED RAMP ADC

This design uses a Pulse-Width Modulated (PWM) generated ramp and a timer to implement a single-slope ADC. Here, the duty cycle is steadily

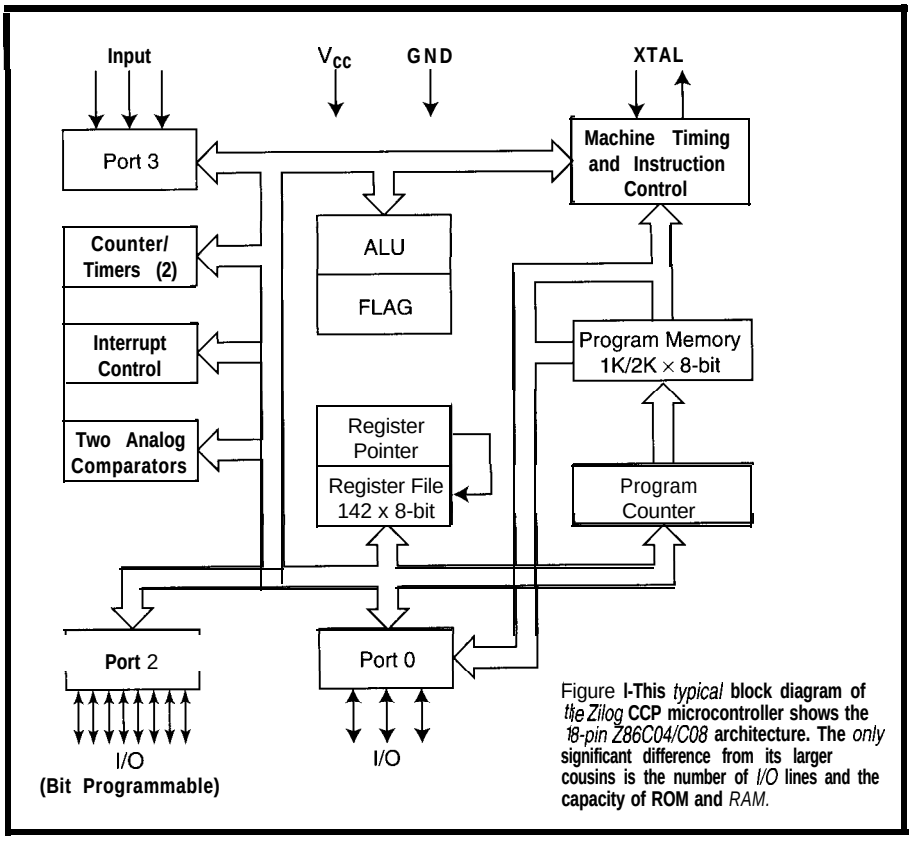


Figure 1-This *typical* block diagram of *the Zilog* CCP microcontroller shows the *18-pin Z86C04/C08* architecture. The *only* significant difference from its larger cousins is the number of *I/O* lines and the capacity of ROM and *RAM.*

increased by incrementing the timer count at the end of each sample period.

The PWM output (POO) is fed to an RC integrator, which produces a linear ramp at the V_{ref} input of the comparators at P33. The analog voltages to be measured are sensed at P3 1 and P32. The voltage range at these inputs is O-4 V. Listing 1 shows the single-slope software routine.

Each comparator has its own interrupt level. When the positive-going ramp exceeds the input voltage, the corresponding interrupt loads the contents of the timer, which is scaled to the measured analog input.

Generate the ramp by incrementing Timer l's value after each sample period. Incrementing the count from 1 (01H) to 200 (C8H) resets the ramp.

The value loaded into the register called Delay determines the sample period. The conversion speed is determined by the speed of the system clock, the analog-input voltage range, sample frequency, and the resolution of the timer.

In this example, the crystal frequency is 8 MHz, the input voltage range is O-4 V, the sampling frequency is about 4800 kHz, and the resolution is 8 bits. The maximum conversion time is then 40 ms.

If you know that the input voltage range is 2-4 V, then you can restrict the ramp voltage to this range, which yields a faster conversion time. Also, if 8 bits of resolution is overkill, adding two counts to the timer at the end of each sample period results in 7 bits of resolution. Adding four counts yields 6 bits.

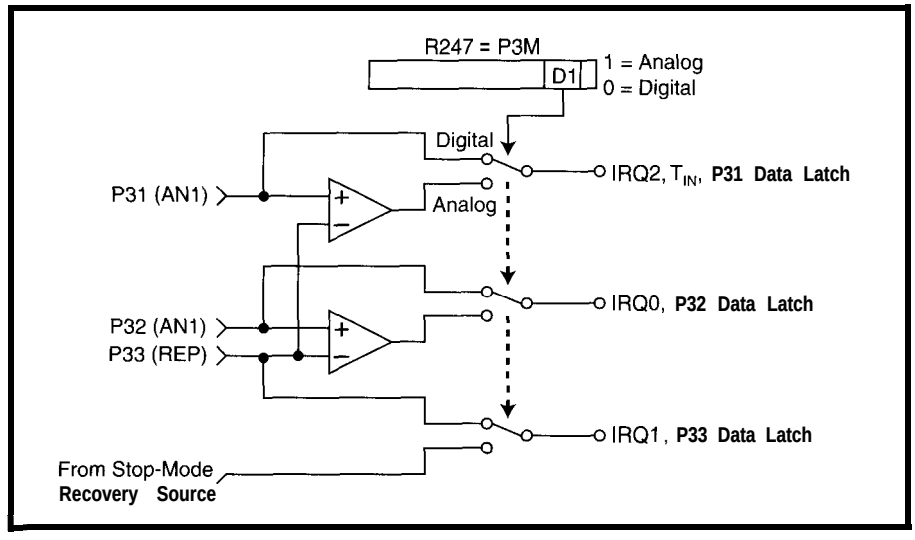


Figure 2—*The* **schematic of the infernal comparator logic shows** *that the* **analog inputs are** *muxed with the* **digital input and are** *selected by setting a bit in the P3M* **register.** *Note that the output of the* **comparators are** *also* **tied** *to the internal interrupt logic.*

BINARY-RAMP COUNTER ADC

The binary-ramp counter uses a DAC in its feedback loop to be compared with the analog input voltage. You implement the DAC by using one of the 8-bit output ports (P2) of the Z8 to drive an R2R resistor network.

A binary-weighted ladder could have been used, but the resistor values tend to get quite large. Individual resistor tolerances and temperature coefficients make it less desirable.

Although the R2R ladder uses twice as many resistors to implement than the binary-weighted ladder, it requires only two resistor values to implement instead of eight for an 8-bit DAC. Of course, if fewer than 8 bits are needed, you'll want to consider the binary-weighted resistor DAC.

Board space may become an issue for the R2R approach when using individual resistors, but these are available in a SIP package. The output of the resistor ladder is fed into the V_{ref} input of the comparators (P33). The analog voltages to be measured are connected to P3 1 and P32, the noninverting inputs of the comparators.

The software for this routine is given in Listing 2, which corresponds with Figure 3. What's nice about this routine is that it can be run totally in the background as long as the conversion time is noncritical.

By using P2 as a counter, a positive-going ramp is generated at P33. (Actually, a negative-going ramp could be

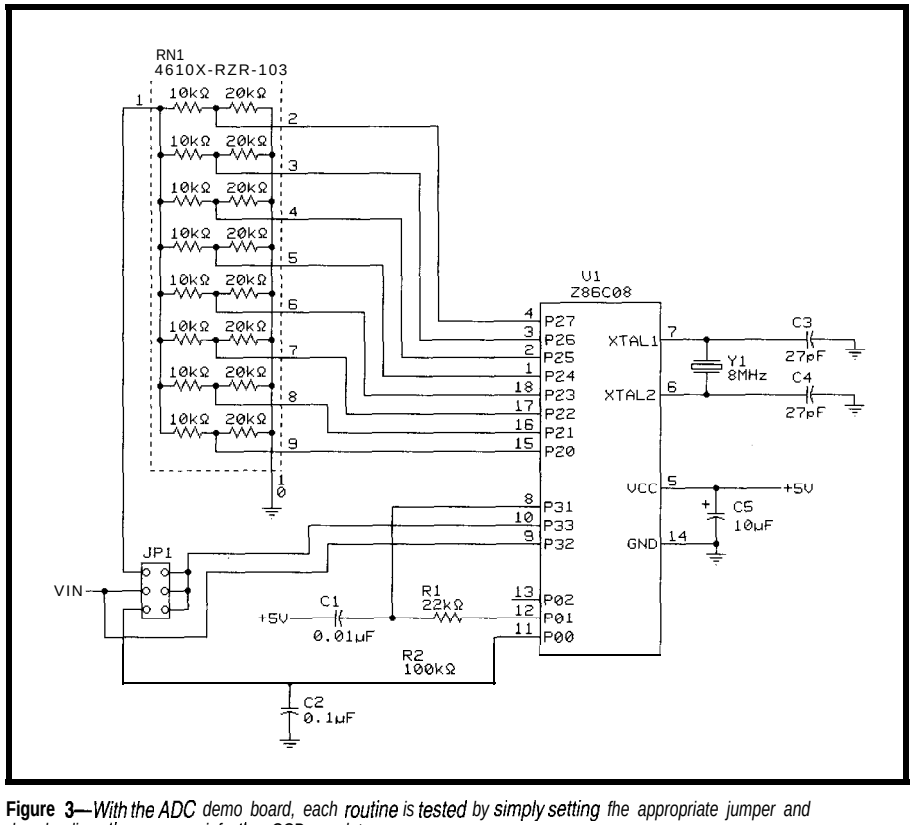


Figure 3—*With the ADC demo board, each routine is tested by simply setting fhe appropriate jumper and downloading the program info the CCP emulator.*

Listing 1—*The PWM-generated ramp ADC uses two timers and counters and an RC integrator to find the unknown input voltage. When the ramp voltage exceeds V_{IN}, an interrupt is generated. The value in timer T1 is the digital representation of this voltage.*

```
delay-hi    .equ  r4
delay-lo    .equ  r5
delay       equ   rr4
count       .equ  r6
irq_0       .equ  1

            .org 0

            .word compptrip
            .word 0
            .word 0
            .word 0
            .word timer-0
            .word timer-1

            .org  Och

            di                  ; disable interrupts
            srp   #0            ; set reg pointer 0-15h
            ld    sp1,#80h      ; set stack pointer at top of stack
            ld    p01m,#4       ; internal stack
            ld    p3m,#3        ; active pull-ups on P2, comp on
            ld    p2m,#0ffh     ; inputs on P2
            ld    pre0,#04h     ; divide by one, one-shot
            clr   to            ; t0 sets the sample period
            ld    pre1,#06h     ; divide by one, one-shot mode
            ld    count,#4      ; start counter with minimum count
            ld    t1,count      ; load counter
            ld    p0,#04h       ; take port pin PO2 high to start
            ld    imr,#31h      ; load interrupt mask register
            clr   irq           ; clear interrupt request reg
            clr   ipr           ; clear interrupt priority reg
            ld    tmr,#0fh      ; load and enable both counters
wait-here:  ei                  ; enable interrupts
            jr    wait-here     ; wait for interrupts


; Timer 0 interrupt routine. Here, the count is incremented every
; 1000 us until a maximum count of 240 is reached. The count is
; then reset, and the integrating capacitor discharges. Port pin
; P00 is then taken high, and the timers are reloaded and enabled.

timer_0:    inc   count         ; start with min count
            cp    count,#240    ; max count?
            jr    ult,continue  ; if not maximum, reload timer
            ld    count,#4        load count with initial value
delay_loop: decw  delay           let cap discharge
            jr    nz,delay_loop ; wait awhile
continue:   ld    t1,count        load timer with count
            or    p0,#04h         take PO2 high
            or    tmr,#0fh        load and enable T1
            iret                  return from interrupt


; Timer 1 interrupt routine. Port Pin P00 is toggled low
; This sets the duty cycle for the PWM.

timer-1:    xor   p0,#04h       ; take PO2 low
            iret                ; return from interrupt


; Comparator interrupt routine. When the ramp voltage at P33
; exceeds the input voltage, the program vectors here. The value
; in register called count is proportional to the input voltage.

comp_trip:  jr    comp_trip     ; stop here

            .end
```

Listing 2—*The binary-ramp counter ADC* **uses** *an R2R ladder to produce the reference voltage. The* **output** *port P2 is* incremented, which provides the positive **ramp voltage. When** *this voltage* **exceeds** V_{IN}, **an** *interrupt is generated.* **P2 contains** *the digital representation of the input voltage.*

```
result    .equ   r4
irq0      .equ   Olh

          .org   Oh
          .word  comp_trip
          .word  0
          .word  0
          .word  0
          .word  0
          .word  0

          .org   Och

init:     di               ; disable interrupts
          ld     spl,#80h  ; set stack at top of reg file
          ld     p01m,#4   ; set for internal stack
          ld     p3m,#3    ; active on P2, turn on comp
          ld     p2m,#00   ; P2 all outputs
          srp    #0        ; set pointer to bottom of reg file
          clr    p2        ; start out at zero
          clr    ipr       ; clear priority reg
          clr    irq       ; clear interrupt pending register
          ld     imr,#irq0 ; enable IRQ0 vector
loop:     ei               ; enable interrupts
          inc    p2        ; increment P2 count
          jr     loop      ; loop around and wait for interrupt
comp_trip:ld     result,P2 ; get P2 count
          clr    irq       ; clear IRQ pending reg
          clr    P2        ; reset P2 count
          iret             ; return from interrupt
          .end
```

generated by first loading P2 with FFH and decremented. However, the DAC logic would then be inverted.)

To begin, P2 is cleared. Since P2 can be treated as a general-purpose register, it can be incremented using the instruction INC P2. For an 8-bit DAC, this requires 256 iterations. This loop can be done inside the main loop, until one of the comparators trips, generating an interrupt.

The value of P2 can be read inside the Interrupt Service Routine (ISR), which represents the analog voltage. The conversion time here depends on how tight the main loop is. If the main loop consists of just a few instructions, then the conversion time is within 1 ms and has a crystal frequency of 12 MHz.

If your design is I/O intense and needs the S-bit port for other functions, you can add an external hardware counter like the 74HC4040. The 74HC4040 is a 12-bit binary counter which helps to implement up to a 12-bit DAC using the appropriate R2R ladder. It needs only two I/O lines from the microprocessor to control the counter: Clock and Reset.

In this design, the counter feeds the R2R ladder, whose output ties to the V_{ref} input of the comparators. The clock is provided by P36, which can either be the internal system clock output divided by two or automatically toggled when one of the timers reaches terminal count (this feature is not available with C04/E04 and C08/E08).

This function uses the Timer-out mode and is completely under software control. The reset line to the counter is provided by P35.

SUCCESSIVE-APPROXIMATION ADC

For applications requiring faster conversion times, you want to consider the successive-approximation method. Like the binary-ramp counter ADC, it also uses a DAC in its feedback loop.

Unlike the binary counter, however, it does a binary search on the input voltage. This search is achieved

by first setting the most-significant bit of the DAC and testing the comparator output. If the comparator output is zero, then the DAC output for this bit is set to zero. If the comparator output is one, then the DAC output for this bit is set to one.

The bits from the output port (in this case P2) move in descending order, performing the same test. This procedure continues until all the bits have been tested, at which point the conversion is complete. This technique uses the same R2R resistor network connected to P2, whose output becomes the comparator's reference voltage.

The analog voltage to be measured connects to either P31 or P32, the noninverting inputs of the comparators. The software for this routine is shown in Listing 3.

To start the conversion, the most-significant bit of P2 is set, resulting in a voltage of half of V_{CC} at the V_{ref} input of the comparator. If V_{CC} is 5 V, the result is 2.5 V. The noninverting comparator input is then tested to see whether it went low. If it didn't, the analog voltage must be between 2.5 and 5.0 V.

Port pin P26 is then set, and the input port is tested again. If low, then this bit is reset and P25 is set. The process continues until all bits of P2 have been tested. The resultant value at P2 is the digital representation of the analog input.

With a crystal frequency of 12 MHz, the conversion time is approximately 70 µs. Even more resolution is available from 10- and 12-bit R2R networks. Of course, more port pins are then needed.

If a track-and-hold function is desired, analog switch CD4016 can be added to the front end. In between samples, the switch is closed, charging a small capacitor. At the end of each sample period-as determined by timer Tl-the switch is opened.

Any error induced by trying to convert a fast-changing analog input is eliminated. The value of C is chosen so that the voltage across it tracks the input voltage, does not bleed off during the conversion, and follows the input voltage in between samples. For a sampling rate of 8 kHz, the input frequency must be less than or equal to 4 kHz to prevent distortion due to aliasing.

Listing *3-The successive-approximation ADC uses an R2R ladder connected to an output port. The software performs a binary search on the input voltage. At end of the conversion, P2 contains the digital representation of the input voltage.*

```
Test_p32  .equ  r4
ring      .equ  r5
results   .equ  r6

          .org Oh
          .word 0
          .word 0
          .word 0
          .word 0
          .word 0
          .word 0

          .org Och

init:     di                        disable interrupts
          ld    spl,#80h            set stack at top of reg file
          ld    p01m,#4             set for internal stack
          ld    p3m,#3         ; active on P2. turn on comp
          ld    p2m,#00             P2 all outputs
          srp   #0             ; set pointer to bottom of reg file
          ld    test_p31,#01h; test comparator with working reg
          clr   p2                  start out at zero
          clr   ring                reset ring register
          scf                       set carry flag
next-bit: rrc   ring                rotate a one through ring reg
          jr    c,EOC               end of conversion if a carry
          or    p2,ring        ; set P2 bit
          nop                       let it settle
          tm    p3,test_p32    ; test P32 input
          jr    nz,next_bit    ; if not low, set next bit
          xor   p2,ring             reset bit if it went low
          jr    next-bit            continue
EOC:      ld    results,p2     ; gets results
          jr    EOC                 loop here when done
          .end
```

W-GENERATED RAMP ADC

For higher resolution or when only one comparator is needed, consider using this routine. This design takes advantage of the Z8's Timer-in mode, which automatically gates off the timer on a falling edge at P3 1.

Bear in mind that a multiplexer internal to the chip allows either digital or analog inputs at P3 1. The output of the mux is then connected to the internal interrupt logic. In this case, the falling edge is at the output of the comparator, just before the interrupt logic.

To take advantage of this feature, the measured analog voltage is connected to the inverting comparator input (P33), and the RC junction is connected to the noninverting input (P31). Port pin PO1 is taken high to discharge the cap.

Listing 4—*The X-generated ramp ADC uses a negative-going ramp for the reference voltage. The timer contains the digital value of the input voftage when the ramp goes below the input voltage. The timer is automatically gated off, and and an interrupt is generated.*

```
result    .equ   r4
delay     .equ   r5

irq2      .equ   4

          .org   0            ;interrupt vector table
```

(continued)

Listing 4-continued

```
            .word   0           ;interrupt 0 vector address
            .word   0           ;interrupt 1 vector address
            .word   0           ;interrupt 2 vector address
            .word   0           ;interrupt 3 vector address
            .word   0           ;interrupt 4 vector address
            .word   0           ;interrupt 5 vector address

            .org    0ch         ;start code execution here

            di                  ;disable interrupts
            srp     #0          ;set reg pointer 0-15h
            ld      spl,#80h    ;set stack pointer at top of stack
            ld      p01m,#4     ;internal stack
            ld      p3m,#3      ;pull-ups on P2, comparators on
            ld      p2m,#00h    ;outputs on P20-P27
            ld      pre1,#06h   ;divide by one, one-shot mode
            clr     t1          ;clear timer
            clr     tmr         ;Timer 1 off
            clr     imr         ;clear interrupt mask reg
            clr     irq         ;clear interrupt request reg
            clr     ipr         ;clear interrupt priority reg
start:      ld      p0,#02h     ;discharge cap
            clr     delay       ;load delay with discharge delay
delay_loop: djnz    delay,delay_loop  ;wait until cap discharged
            clr     P0          ;take PO1 low (start ramp)
            or      tmr,#0ch    ;load and start timer
poll_irq2: ei                   ;enable interrupts
            tm      irq,#irq2   ;test for comparator trip (IRQ2)
            jr      z,poll_irq2;if not set, continue polling
            ld      result,t1   ;get count from T1
conversion_done:jr conversion_done   ;loop here when done
            .end
```

The voltage at P31 is at +5 V before conversion, so the output of the comparator is positive. To start conversion, port pin PO1 is taken low, which starts to charge the cap. At the same time, the timer is enabled. When the voltage at P31 is less than P33, the comparator trips and gates off the timer.

The software reads the contents of the timer at its leisure. Listing 4 offers this routine. The timer resolution is 1 us. The timer count is directly proportional to the input voltage. Since the maximum comparator input voltage is 4 V, the maximum timer count is about COH.

VOLTAGE-TO-FREQUENCY CONVERTER

If you're xpanding on the binary-ramp counter ADC, you can easily implement a voltage-to-frequency converter. This converter uses P2 for the DAC output, feeding a R2R ladder. The output of the ladder then feeds the V_{ref} input of the comparators.

Inside the main loop of the program, the value of P2 is incremented.

When the voltage at the V_{ref} input exceeds that of V_{IN}, the comparator trips and an interrupt is generated.

Inside the interrupt service routine, simply load the value of P2 into one of the counter/timers. The counter/timer must be set up for Timer-out mode, in which case the output port pin P36 toggles automatically when the timer hits terminal count.

When run in modulo-N mode, the timer continues to count down and reload itself with the initial count when it reaches zero. This procedure produces a square wave at P36 whose frequency is inversely proportional to the input voltage.

To produce a frequency that is directly proportional to the input voltage, take the one's complement of P2 before loading the timer. This routine is completely interrupt driven with minimal software overhead. The output frequency of the VCO can be scaled by using the timer prescaler-a six-bit down counter that precedes the timer.

The output frequency can be calculated by the following formula:

$$F = \frac{\frac{Fxtal}{16}}{prescalerx \quad timer}$$

Assuming a 12-MHz crystal, the frequency can range from kilohertz to megahertz. The software listing for this routine is shown in Listing 5.

Notably, all CCP Z8s work with this routine, except C04/E04 and C08/E08.

ONE FOR THE ROAD

Just in case the previous examples don't whet your ADC appetite, Zilog has announced a new Z8 with an S-channel, 8-bit ADC. Dubbed the Z86C83/C84, it comes in a 28-pin SOIC package with 4 KB of ROM, 237 bytes of RAM, and up to 21 I/O lines.

The onboard ADC is a half-flash 8-bit ±½ LSB with an 8-channel mux. The conversion time depends on the clock frequency, but with a 16-MHz crystal, it is specified at 8 µs maximum. Any unused analog inputs are usable as standard digital I/O.

The ADC uses a unique programmable offset control of the resistor ladder that compresses the converter's dynamic range for a maximum effective 9-bit resolution. The block dia-

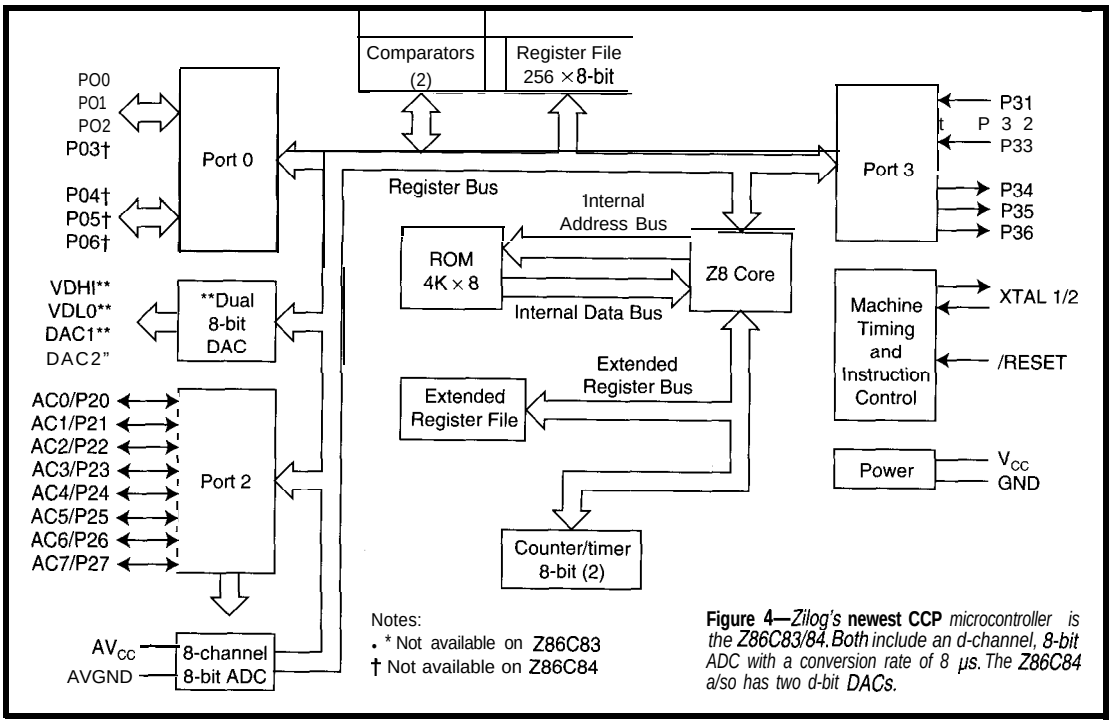


Figure 4—*Zilog's* **newest CCP** *microcontroller is the Z86C83/84. Both include an d-channel, 8-bit ADC with a conversion rate of 8 µs. The Z86C84 a/so has two d-bit DACs.*

Listing 5—*The voltage-frequency converter uses an R2R /adder to find the input voltage using the binary-ramp technique, then loads this value into a timer which automatically toggles a port pin when it times out.*

```
irq0    .equ 01h

        .org 0h
        .word comp_trip
        .word 0
        .word 0
        .word 0
        .word 0
        .word 0

        .org 0ch

init    di              ;disable interrupts
        ld   spl,#80h   ;set stack at top of reg file
        ld   p01m,#4    ;set for internal stack
        ld   p3m,#3     ;active on p2, turn on comp
        ld   p2m,#00    ;p2 all outputs
        srp  #0         ;set pointer to bottom of reg file
        clr  p2         ;start out at zero
        clr  ipr        ;clear priority reg
        clr  irq        ;clear interrupt pending register
        ld   pre1,#07h  ;/1, modulo N mode
        clr  t1         ;timer initialized to max count
        ld   tmr,#8ch   ;load and enable T1, T-Out mode
        ld   imr,#irq0  ;enable irq0 vector
        ei              ;enable interrupts
        inc  p2         ;increment P2 count
        jr   loop       ;loop around, wait for interrupt
```

(continued)

Listing 5-continued

```
comp_trip: ld    T1,P2     ;load P2 count into timer
           clr   irq       ;clear irq pending reg
           clr   P2        ;reset P2 count
           iret            ;return from interrupt
           .end
```

gram for the Z86C83/C84 is shown in Figure 4.

COST REDUCTION WITHOUT SACRIFICE

You can see that applications requiring an ADC can be achieved without compromising accuracy, speed, or system cost. It is up to the design engineer to experiment with the routines for optimum performance. These routines are designed to be functions called from the main program.

It's possible to modify the program to be completely interrupt-driven. This change allows the microprocessor to handle other tasks in between the interrupts. By adding a software UART routine to the above programs, you can implement a truly low-cost, PC-compatible data acquisition system. ❑

Don Newquist received his BSET from California Polytechnic State University in 1983. His background includes digital design, programming, and product development. He has worked as a field applications engineer for Zilog for the last 5 years. You may reach Don at dnew@zilog.com or at 104143.234@compuserve.com.

REFERENCES

Ciarcia, S., "Analog Interfacing In The Real World," ***Ciarcia's Circuit Cellar, Volume IV, 1984.***

Distaso, L.A., "Analog to Digital Conversion Techniques With COPS Family Microcontrollers," National Semiconductor Corp., ***National Semiconductor Microcontroller Databook, 1987.***

Zilog, ***Zilog Discrete Z8 Microcontroller Product Specification Databook,*** DC 8318-02, Q3, 1994.

Zilog, ***Zilog Z8 Microcontrollers User's Manual,*** UM95Z8000103, ***1995.***

SOURCE

Z8 CCP, CCP emulator (Z86CCP001ZEM)
Zilog, Inc.
210 E. Hacienda Ave.
Campbell, CA 95008
(408) 370-8000
Fax: (408) 370-8056

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

David Gaddis

Battery Load/Charge Analyzer

Here's the circuitry and software of a universal, portable test instrument to evaluate battery charge/discharge. It transforms tedious time-based measurements into simple, automated operations.

The last time I designed a battery-powered product, I promised myself I'd design a stand-alone test instrument as soon as possible. I needed it to assist me in evaluating battery charge and discharge performance.

The 7th Annual Circuit Cellar Design Contest presented just the opportunity. Let me describe the instrument I designed.

DESIGN NEEDS

In the past, the method I most often used to collect battery-performance data required several VOMs connected for voltage and current measurements. Recording the readings at regular intervals and monitoring for any abnormal readings was time consuming.

I also used a PC with an analog-to-digital card. But, even though I could perform all the desired measurements, it was not universal or portable. Plus, I'd rather not monopolize my PC collecting data.

I imagined a test instrument that connected, looked, and operated like a standard digital VOM with a few more connections and functions. It would provide basic voltage and current measurements and would transform tedious time-based measurements into a simple, automatic operation.

I needed to measure milliamp and milliwatt hours, charge and discharge times, peak, average, minimum, and maximum voltages, current, and powers. Figure 1 catalogs these measurements and calculations.

I wanted this test instrument to stand alone, run repetitive tests, record necessary measurements, and connect between the battery, load, and charger. It needed to cycle the battery through charge and discharge cycles. It would work with any battery chemistry and terminate and alarm when the charge or discharge exceeded preset limits.

When required, this instrument would provide a PC interface for recording real-time data. It would be able to select the number of cells, specified mAh, and battery chemistry and to program cyclic tests.

I called the instrument a *Buttery Charge/Discharge Analyzer* (BCDA). Figure 2 lists its principal hardware design specifications.

BATTERIES

Batteries are devices that convert energy via chemical or physical reac-

Voltage
- Instantaneous
- Peak (spikes or transients at 1-kHz sample rate)
- Maximum at full charge or start of test
- Average over test duration
- Minimum at full discharge

Current (signed)
- Instantaneous
- Peak (spikes or transients at 1-kHz sample rate)
- Maximum
- Average over test duration
- Minimum

Power (signed)
- Instantaneous
- Maximum
- Average of test duration
- Minimum

Discharge and charge mAh (signed)
Discharge and charge mWh (signed)
Full-charge detection (i.e., maximum cell voltage)
Full-discharged detection (i.e., minimum cell voltage)
Real-time mAh and mWh capacity calculator
Battery efficiency for mAh and mWh charge and discharge
Time to fully charge battery
Time to fully discharge battery
Number of charge and discharge cycles

Figure 1—*The list of measurements and calculations merges the instantaneous and time-dependent values. All operations occur automatically after the start of a test run.*

Microcontroller:	MC68HC705C8
Crystal:	4 MHz
EPROM:	
	12-bit—serial
	500-Hz
	+5 to
Power Supply Input:	6.0-20 VDC @ 5 mA
	DC (both relays on)
PCB Dimensions:	5" × 3.2"
Pkg. Dimensions:	6.25" x 3.8" x 2.5"

gure 2-The principal hardware design specifications of the Battery Charge/Discharge Analyzer fit neatly in a small, portable unit.

tion into electric current. Figure 3 shows the breakdown of different battery classifications along the major lines of chemical versus physical energy batteries.

The chemical battery group is composed of three categories. *Primary batteries* are batteries whose energy exhausts when the active materials are consumed. *Secondary* or *storage batteries* use materials that are regenerated by charging. *Fuel cells* are batteries which receive active materials from external sources and then convert them into electrical energy.

Examples of primary batteries are carbon-zinc, alkaline, lithium, silver oxide, and mercury. The carbon-zinc, alkaline, and silver oxide have a nominal voltage of 1.5 V per cell. The lithium has a nominal voltage of about 3 V per cell and has by far the best capacity, discharge, shelf-life, and temperature characteristics.

The mercury has a nominal voltage of 1.4 V per cell. The carbon-zinc and alkaline are used most in medium-current applications. The silver oxide and mercury are used for applications where constant voltage at low currents over long periods is desired.

Examples of secondary or storage batteries are nickel-cadmium (NiCd), lead-acid, lithium, and even a new alkaline. The NiCd has a nominal voltage of 1.2 V Per cell while the lead-acid has a nominal voltage of 2.0 V per cell. Table 1 summarizes the voltage specifications for several types of batteries.

Battery capacity is the amount of electrical energy the battery delivers under specified discharge conditions. The capacity value, specified by "Ah" for ampere-hours and "mAh" for milliampere-hours, is the product of load current and time. The capacity value C changes depending on the discharge rate.

For example, a NiCd cell might be rated at 660 mAh at a C/5 rate, but only 600 mAh at 1-C rate. Most manufacturers of rechargeable batteries specify the minimum standard capacity based on a 5-h rate at a 0.2-C discharge current.

The minimum and maximum battery voltage, current, and power are very important during charge and discharge. Many products won't power up if battery voltage is too low. If it is too high (i.e., after a fast charge), the input filtering can be damaged.

High currents can indicate abnormalities, while low currents can mean that power hasn't been applied to all circuits. Knowing the power requirement at different inputs aids in designing the battery pack and charger [3].

Current and power discharge profiles reveal a lot about a product's operation. Knowing where it pulls the most current and at what voltage helps answer nasty run-time questions.

CIRCUIT DESCRIPTION

Figure 4 shows the schematic for the BCDA. As you can see, it is built around a Motorola MC68HC705C8 8-bit microcontroller with a 12-bit ADC to digitize battery voltage and load and charge current. Calibration coefficients and setups are stored in an EEPROM along with some temporary data.

The user controls the BCDA with four softkeys, feedback by key click and alarm annunciator, and a 1 -line x 16-character LCD. Control of the load and charger is provided by two normally open 5-A relays. A PC can be connected via a three-wire (TX, RX, GND) RS-232 port.

The charge and discharge current is sampled across R6, a 0.1-Q 5-W power resistor. The voltage developed across R6 is buffered by a differential amplifier, U7a. U7 is a quad, single-supply, rail-to-rail operational amplifier.

To keep the input voltages at the op-amp within the supply rails, the differential amplifier attenuates the input voltage by approximately 13. An amplifier built with U7c provides a gain of 5 1. Therefore, the DC and low-frequency AC gain from the voltage developed across R6 to the ADC is 3.9.

Chemical Batteries
- Primary Batteries
  - Carbon-Zinc Dry Cell
  - Mercuric Oxide Battery
  - Alkaline-Manganese
  - Silver Oxide Battery
  - Silver Chloride Battery
  - Zinc-Air Cell
  - Lithium Battery
    - Fluorocarbon-Lithium
    - Manganese Dioxide-Lithium
    - Copper Oxide-Lithium
- Secondary Batteries
  - Lead-Acid Battery
    - Vented Type
    - Sealed Type
  - Alkaline Batteries
    - Nickel-Iron (Edison)
    - Nickel-Cadmium Batteries
      - Jungner
      - Sintered
      - Sealed
    - Silver Oxide-Zinc
    - Silver Oxide-Cadmiun
  - Carbon Lithium
- Others-Fuel Cells

Physical Energy Batteries
- Solar Cell
- Nuclear Energy
- Thermal Battery

Figure 3—*Batteries* can be classified as chemical and physical [1]. Also, see Dave *Prutchi's* overview of *batteries* (INK 55). *The BCDA can perform tests on all types listed, and if is* flexible enough to adapt to *almost any* situation that requires combinations *of voltage, current, power, and energy measurements.*

	Voltage (VDC per cell)				
Chemistry	No Load	Loaded	Cut Off	Charged	Maximum
Silver Oxide	1.4	1.4	0.9	na	na
Alkaline	1.5	1.15	0.8-1.1	na	na
Lithium	3.8	3.0	2.0	na	na
NiCd	1.35	1.2	1.0	1.4	1.8
NiMH	1.4	1.25	1.0	1.45	1.8
Lead Acid	2.1	2.0	1.75	2.4	2.6

Table I-The BCDA supports *several* common *battery chemistries [2]*

The current differential and gain amplifiers are biased at half of V_{ref} or about the 2.33 V provided by buffer U7b.

With a ±5-A input current, the voltage swing into the ADC is ±1.956 V centered around 2.33 V. Each amplifier's voltage gain is 6 dB at about 300 Hz for a combined 12 dB. This attenuation results from the low-pass filters in the feedback which are created by the parallel combinations of R18 and C22 at U7a and R7 and Cl9 at U7c.

Battery voltage is monitored at the battery-connection side of R6. The voltage divider of R19 and R20 attenuate the battery voltage by 4.4 before buffering by U7d. The ADC input voltage range is 0–V_{ref} for battery voltages up to 20.5 V. Battery voltage is always connected to the ADC regardless of the state of relays K1 and K2. This feature enables the software to detect battery connection.

Note that U7 needs a very high CMRR of 80 dB or higher with a rail-to-rail input common-mode voltage range and a rail-to-rail output capability.

V_{ref} is provided by D2, a TL43 1 programmable precision reference set for about 4.66 V. The reference is heavily filtered to minimize the system noise affecting the ADC conversion.

U8 is a Linear Technology LTC1293 ADC containing a built-in sample-and-hold and a six-input multiplexer. It performs 12-bit unipolar conversions on the conditioned voltage and current inputs. The LTC1293 connects via a direct four-wire interface to the SPI port on U6, a Motorola MC68HC-705C8. Table 2 gives the input voltage, current range, and resulting resolution.

The microcontroller interfaces to LCD1 via port B and three outputs on port C. Software controls the direction of port B for access to all features of the display. The direction of port B is changed to read the LCD busy flag so that timing loops are not required. The four user input switches interface to four inputs of port A.

Relays K1 and K2 control the charge and load currents, respectively. The relay coils require 40 mA at 5 V to pull in, so they are controlled by the microcontroller through Q2 and Q3, which

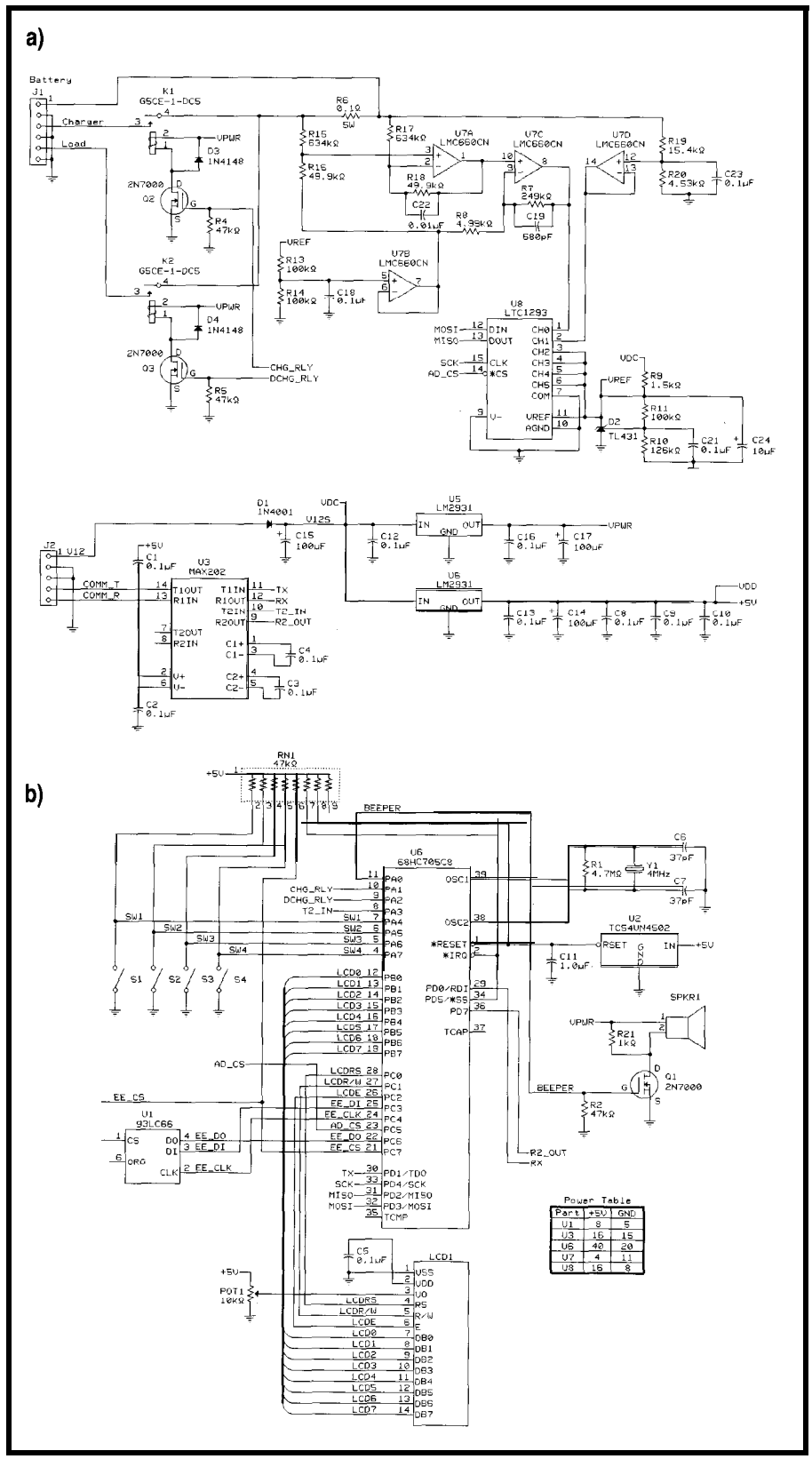


Figure 4-The *BCDA* **performs** *both* **voltage and current conversions** *with the LTC1293 ADC. If is* **a** *12-bit analog-to-digital converter that includes a 6-input multiplexer and sample* **and** *hold.*

are 2N7000 JFET transistors. The gate-to-ground resistors, R4 and R5, keep the relays off during power-up and power-down transitions.

Digital and analog power is supplied by regulator U4, an LM293 1-5A0. Regulator U5, also an LM2931-5A0, provides separate power to the relays

Input	Range	Resolution
Current	-5.00 to 5.00 A	2.44 mA
Voltage	0 to 20.00 V	4.9 mV

Table 2—*The BCDA* digitizes both the voltage and current *inputs with* a *12-bit ADC.*

and the beeper since all three can be active simultaneously.

The serial EEPROM U1, a 93LC66, is interfaced to four signals on port C of the microcontroller. U1 has 256 x16 words of nonvolatile storage for coefficients, setups, and temporary data.

RS-232 level serial input and output are provided by U3, a Maxim MAX202 +5-V-powered driver and receiver. U3 connects to the SCI port on U6. The other driver and receiver, which can be used for auxiliary control or debugging, connect to the unused ports on U6, but are left unconnected on the RS-232 side.

Photo 1 shows the hardware implementation of the BCDA. A plastic box eases assembly, but metal enclosures provide better shielding. I installed components on both sides of the double-sided PCB. The key switches, LCD, beeper, and contrast adjustment pot are on the circuit side, while the rest are on the component side.

Adjusting the ratio of R7 and R8 changes the gain of U7c, thereby altering the input current range. Setting the range to ±2 A provides a resolution of slightly less than 1 mA per bit. A more sensitive current input would be valuable for low-current applications.

The divider built with R19 and R20 sets the voltage-input range. Almost any voltage input is possible if the input limits on U7a are not exceeded. This constraint should not be a problem as long as the resistor values of R15, R16, R17, and R18 are maintained.

Function	S1	s2	s3	s4
Display Software Version		x	x	
Toggle Charge Relay	X			X
Toggle Load Relay	X		X	
Initialize Logging Values	X	X	X	x

Table 4—*The* four soft switches *can* be used in combination to directly *access* certain functions.

The corresponding routines that process the voltage and current values, including the calibration and linearization functions, need to be changed accordingly.

CALCULATIONS

All relevant values are calculated and accumulated twice per second. The ADC voltage (V_{raw}) and current (I_{raw}) values are the result of averaging 500 ADC samples. The summing operation is performed in the 1-kHz interrupt routine, while the averaging calculation is completed in the foreground user interface routine. Calculate V_{raw} by:

$$V_{raw} = \frac{SUM\{v[x]\}}{500}$$

Mode	S1	s2	s3	s4
Menu	Previous	Scroll Up	Scroll Down	Next/Execute/Clear
Select	Exit	Scroll Up	Scroll Down	Select/Deselect
Edit	Exit/Abandon	Inc Value	Dec Value	Accept/Save

Table 3—*The four soft switches are used* individually *to scroll* through menus, select values for display, and edit input values.

where *v[x]* is the individual ADC voltage sample over the range of x from 0 to 499. Calculate I_{raw} by:

$$I_{raw} = \frac{SUM\{i[x]\}}{500}$$

where *i[x]* is the individual ADC current sample over the range of x from 0 to 499.

Each *v[x]* and *i[x]* value is compared to the corresponding prevailing peak value, and if it's greater, it replaces the last recorded value. Peak valued are recorded for voltage and charge and discharge currents.

The V_{raw} and I_{raw} values are used as arguments in a first-order linear equation for a straight line to calculate V_{DC} and I_{DC}, respectively. V_{DC} and I_{DC} are in real-world units of volts and amps. I_{DC} is a signed value. The negative current value denotes charge current while positive stands for discharge current. Calculate V_{DC} by:

$$V_{DC} = (V_{RAW} \times Mv) + Bv$$

where Mv and Bv are coefficients stored in EEPROM during voltage calibration.

Calculate I_{DC} by:

$$I_{DC} = (I_{RAW} \times Mi) + Bi$$

where *Mi* and *Bi* are coefficients stored in EEPROM during current calibration.

The power, P_{DC}, is the product of V_{DC} and I_{DC}. The P_{DC} value carries the current's sign with the same designations on polarity. Calculate P_{DC} by:

$$P_{DC} = I_{DC} \times V_{DC}$$

Ampere-hours (Ah) and watt-hours (Wh) are calculated by accumulating a running total of the current (I_{total}) and power (W_{total}) values calculated every 0.5 s and dividing by 7200. Calculate ampere-hours by:

$$I_{total} = I_{total} + I_{DC}$$

where I_{DC} represents the amps calculated every 0.5 s, then:

$$Ah = \frac{I_{total}}{7200}$$

where 7200 is the number of readings per hour.

Calculate watt-hours by:

$$W_{total} = W_{total} + P_{DC}$$

where P_{DC} represents the watts calculated every 0.5 s, then:

$$Wh = \frac{W_{total}}{7200}$$

where 7200 is the number of readings per hour.

Separate ampere-hour and watt-hour values are accumulated for charge and discharge.

A battery's charge efficiency is a measure of the use of input energy during charge to replace the energy expended during discharge. During charging, the active materials are converted into a charged form.

Charge efficiency depends on both the charge and discharge rates. The

charge efficiency is the ratio of discharge to charge times 100. Once the charge efficiency is calculated, the appropriate battery can be selected according to operating requirements.

Efficiency is calculated for both ampere-hours and watt-hours. Calculate the charge efficiency based on amps by:

$$\text{Charge efficiency (A)\%} = \frac{\text{Discharge Ah}}{\text{Charge Ah}} \times 100$$

Calculate the charge efficiency based on watts by:

$$\text{Charge efficiency (W)\%} = \frac{\text{Discharge Wh}}{\text{Charge Wh}} \times 100$$

Once this information is collected, the appropriate charger and battery is obvious. The BCDA aids and simplifies the acquisition of the pertinent data necessary to make that decision.

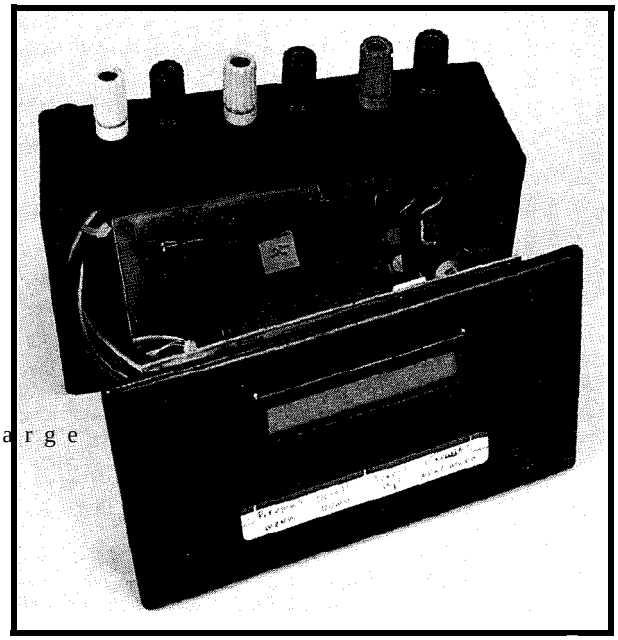


Photo I--The *BCDA* is *built on a sing/e double-sided PCB with components on both sides. A metal enclosure would provide better shielding and noise immunity.*

SOFTWARE DESCRIPTION

The BCDA program is written in assembly, which provides the most control and flexibility without sacrificing speed. The MC68HC705 does not easily lend itself to high-level languages like C, but I do like a stack-organized machine language.

The BCDA program includes a collection of functions that implement 16- and 32-bit math, data and parameter stack operations, timer/counter operations, EEPROM store and recall functions, and LCD and serial output routines.

The program consists of two main sections:

- a background interrupt
- a foreground user interface

The BCDA executes a repeating interrupt at a 1-kHz rate that performs the ADC conversions, key scanning, annunciators, and basic counting for the timers. The foreground loop provides the user interface and real-time calculations.

The code for the BCDA is simple and straightforward, except for the double-buffered summing registers. To continuously sample the analog in-

puts, these registers are controlled by the 1 -kHz interrupt routine. The interrupt routine is triggered by the output-compare function of the MC68HC705 microcontroller. The output-compare registers are reloaded on entry each time the code is executed for the next interrupt.

The 1-kHz interrupt performs many real-time operations such as:

- sampling and digitizing the voltage and current inputs. It compares and records the peak voltage and current values.
- summing the digitized voltage and current values into the selected bank of summing registers. It checks the sample counter for 500, sets data-ready flag, switches banks, and zeros the sum registers.
- scanning and debouncing the key inputs
- toggling the annunciator output if on and processing the annunciator on timer
- processing 1-ms and 100-ms countdown timers ii active
- processing the I-ms countdown timer chain and incrementing the l-s real-time clock

The voltage and current inputs are sampled at a 1-kHz rate and summed into a currently active set of 32-bit summing registers. After 500 samples, the active sum registers are switched, the data-ready flag is set, the new sum registers are cleared, and the summing process starts over.

The sums are double-buffered with one set actively under interrupt control at all times. The other set is used by the foreground program to calculate real-world values. Listing 1 shows the code that samples and sums the values into the currently active bank of sum registers.

The user interface is basically a table-driven menu system where each menu item is defined as a record in a table. Every item's actions are defined by the item flag bits. An item could be the title of a menu, a static value, or real-time calculations (at a 0.5-s update rate).

The value-editing and list-selection operations are also simple table-driven

Listing *l--The digitized analog samples are summed in-line to the current/y active set of summing registers.*

```
tmrntr:



tmrntr_atod
  lda   sample_counter+1        ;is sample count = O?
  ora   sample_counter
  bne   tmrntr_same_bank
  lda   a2d_flag                ;yes, toggle register bank select
  eor   #BANK.
  sta   a2d_flags
  lda   #{SAMPLES/256}          ;set sample counter = 500
  sta   sample_counter
  lda   #{SAMPLES&$Off}
  sta   sample_counter+1
  bset  READY!,a2d_flags        ;set data ready flag

* zero new selected bank
  brclr BANK!,a2d_flags,tmrntr_clr_bank0;load address of
  ldx   #current1               ;present register
  bra   tmrntr_clear            ;bank
tmrntr_clr_bank0
  ldx   #current0
tmrntr_clear
  lda   #8
sums
tmrntr_clear_loop
  clr   ,x
  incx
  deca
  bne   tmrntr_clear_loop

* same register bank as last pass
tmrntr_same_bank                ;no, continue collecting readings
  lda   sample_counter+l        ;one sample closer to calculation
  sub   #1
  sta   sample_counter+l
  Ida   sample_counter
  sbc   #0
  sta   sample-counter

* Sample the current and voltage inputs and sum in-line to
* the present register bank. The sums are 32-bit values  The
* ADC is read through the SPI port at 1-MHz clock rate
* channel O-current input
brclr   BANK!,a2d_flags,tmrntr_c_bank0;select present
  ldx   #current1               ;register bank
  bra   tmrntr_current
tmrntr_c_bank0
  ldx   #current0
tmrntr_current
  lda   #ATOD_0                 ;send first ADCcommand byte
  bclr  A2D_CS!,PORTC
  sta   SPDR
  brclr SPIF,SPSR,*             ;wait till SPI transfer complete
  lda   #ATOD_2ND               ;send second ADC command byte
  sta   SPDR
  brclr SPIF,SPSR,*             ;wait till SPI transfer complete
  lda   SPDR                    ;read msb of ADC conversion
  sta   SPDR                    ;send third ADC command byte
  sta   tmrntr_data
  brclr SPIF,SPSR,*             ;wait till SPI transfer complete
  bset  A2D_CS!,PORTC
  Ida   SPDR                    ;read lsb of ADC conversion
  add   LSB32,x                 ;current_sum += ADC valune
  sta   LSB32,x
  lda   tmrntr_data
  adc   NSB32A,x
  sta   NSB32A,x
  bcc   tmrntr_atod0_ncl
```

(continued)

Listing I-continued

```
  inc   NSB32B,x
  bne   tmrntr_atod0_nc1
  inc   MSB32,x
tmrntr_atod0_nc1

* channel 1-voltage input
  brclr BANK!,a2d_flags,tmrntr_v_bank0;select present
  ldx   #voltage1              ;register bank
  bra   tmrntr_voltage
tmrntr_v_bank0
  ldax  #voltage0
tmrntr_voltage
  lda   #ATOD_1                ;send first ADCcommand byte
  bclr  A2D_CS!,PORTC
  sta   SPDR
  brclr SPIF,SPSR,*            ;wait till SPI transfer complete
  lda   #ATOD_2ND              ;send second ADC command byte
  sta   SPDR
  brclr SPIF,SPSR,*            ;wait till SPI transfer complete
  lda   SPDR                   ;read msb of ADC conversion
  sta   SPDR
  sta   tmrntr_data
  brclr SPIF,SPSR,*            ;wait till SPI transfer
  bset  A2D_CS!,PORTC
  lda   SPDR                   ;read lsb of ADC conversion
  add   LSB32,x                :voltage_sum += ADC value
  sta   LSB32,x
  lda   tmrntr_data
  adc   NSB32A,x
  sta   NSB32A,x
  bcc   tmrntr_atod1_nc1
  inc   NSB32B,x
  bne   tmrntr_atod1_nc1
  inc   MSB32,x
tmrntr_atod1_nc1


rti
```

routines. Since the MC68HC705 has only an 8-bit index register, the text display routines are pseudotable-driven. Each call requires the code of the 256-byte page and an offset to the string. Complete source code for these routines and others is available from the Circuit Cellar BBS.

SOFTWARE OPERATION

The BCDA is controlled by the user through six menus and four softkeys. All operations are accessed either by scrolling to the specific function and executing it or by special multikey combinations for direct control. Table 3 details the actions available from the four keys individually. Table 4 tabulates the direct function assignments of specific key combinations.

The user can set up the BCDA, choosing the values to display and the test sequence to perform. The multimeter mode lets you monitor specific measurements.

Once a test is started, the BCDA runs until it either completes the test or is commanded to stop. All menu functions are available, even during a test. The test is actually running during the idle time-much like a multitasking operating system.

FUTURE IMPROVEMENTS

The BCDA works great in its basic form. Overall, it meets all of the original design goals and successfully diagnoses problems with battery-powered equipment. There are several items I want to change that would improve its value as a stand-alone test instrument.

First, I would add an input to measure the battery temperature via an external sensor, which could be either a thermistor or a Dallas 3-pin digital temperature sensor. It records and displays temperature as one of the scrolling values during a test and in the multimeter menu.

Second, I'd add 64 KB of RAM. This addition would allow up to 48 hours of current, voltage, and temperature values to be recorded once every 30 seconds. The RAM could be dumped to a PC for more analysis.

Third, I would provide external access to other ADC inputs, including perhaps a second voltage and current input. This change could be used to measure a regulated voltage output and the resulting load current.

Fourth, I'd provide several digital inputs and outputs that could monitor and control either the charger or the load. This feature would determine if a charger is correctly measuring the battery's fully charged or low-battery condition. The status of each input would be recorded along with the other values mentioned earlier. ❑

David Gaddis (KE4KPC) is a senior hardware and software engineer with Computational Systems where he develops predictive maintenance equipment for the rotational equipment market. He has 17 years of design experience with inspection systems for nuclear reactors, medical X-ray machines, power utility metering devices, laboratory microwave heating systems and laser-based alignment systems. David may be reached at gaddis@ix.netcom.com.

REFERENCES

[1] Sanyo Electric, *CADNICA Sealed Type Nickel-Cadmium Batteries,* Engineering Handbook SF-6336, 3-5, 1991.

[2] *Calculations,* McGraw-Hill, New York, NY, 11.2-11.3, 1988.

[3] Gates Energy Products, *Sealed* Applica-

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

Sniffing EMI in the Near Field

FEATURE ARTICLE

David Prutchi

Noncompliance with FCC and EMC regulations affects everyone! So, David shows designers how to construct probes that identify and isolate sources of EMI that inevitably make it into a circuit.

The technologies involved in modern circuit design have considerably blurred the boundaries between digital and analog worlds. Suddenly, 100-MHz clocks became commonplace in high-performance digital circuits. Now it's necessary to consider every connection between components as an RF-transmission line.

As the need for higher performance pushes designers toward high-speed technology, the market demands more compact, lighter, and less power-hungry devices. With smaller size, analog effects again enter into consideration.

As components and conductors come into close proximity, coupling between circuit sections becomes a problem. Obviously, self-interference must be eliminated to make the product workable, but this still doesn't make the product market worthy.

To ensure that devices do not interfere with each other, strict regulations concerning electromagnetic compatibility are now enforced worldwide.

In the U.S., the FCC regulates the testing and certification of all electronic devices which generate or use clock rates above 9 kHz [1]. In principle, the FCC's charter protects communications from unwanted electromagnetic interference (EMI).

In the European Union, an Electromagnetic Compatibility (EMC) Directive prohibits undue interference to radio and telecommunications equipment. Equipment must possess sufficient immunity to operate as intended in the presence of interference [2].

Designers must follow these requirements. Failure to comply with EMI and EMC regulations has a serious impact for everyone associated with the product—designers, manufacturers, marketing, distribution, and even customers.

Noncompliance can halt manufacturing and distribution, incur fines, and cause the public posting of notices of noncompliance to warn potential customers and other agencies [3].

Assuring compliance involves an extensive series of tests. The EMI and EMC standards clearly define the construction of test sites, as well as the necessary test procedures. Even a fairly Spartan facility capable of conducting these tests can cost over $100,000 just to set up. Most companies, therefore, hire an outside test lab at $1000–2000 per day to conduct testing.

Considering how fast charges accumulate during testing, it's obviously not smart to simply hire a test lab and wait for the results. Rather, designers should familiarize themselves with the relevant EM1 and EMC standards and consider the compliance requirements at every stage in the design process.

In INK 61, Jeff Bachiochi presented the basics of designing digital circuits for compliance, as well as some methods for troubleshooting circuits to reduce potential problems at the time of testing for compliance.

In this article, I delve deeper into the theory of how digital circuits produce EMI. I also describe some low-cost tools and methods so you can identify and isolate the EM1 sources that inevitably make it into a circuit.

RADIATED EMISSIONS FROM DIGITAL CIRCUITS

Digital circuits constantly switch the state of lines between high- and low-voltage levels to represent binary states.

Figure la shows that the resulting time-domain waveform on any single line of a digital circuit can be idealized as a train of trapezoidal pulses of amplitude A (either current $[I]$ or voltage $[V]$), risetime (t_r), falltime (t_f) at 10–90% amplitude, pulse width (τ) at 50% amplitude, and period (T).

The Fourier envelope of all frequency-domain components generated by such a periodic pulse train can be

approximated by the nomogram in Figure I b. The frequency spectrum is mainly a series of discrete sinewave harmonics starting at the fundamental frequency $f_0 = 1/T$ and continuing for all integer multiples of f_0.

The nomogram identifies two frequencies of interest. At f_1, the locus of the maximum amplitudes rolls off with a $1/f$ slope. At f_2, the locus rolls off at a more abrupt rate of $1/f^2$. These frequencies are located at:

$$f_1 = \frac{1}{\pi\tau} = \frac{0.32}{\tau}$$

and

$$f_2 = \frac{1}{\pi t} = \frac{0.32}{t}$$

where t is the faster of (t_r, t_f).

The envelope of harmonic amplitude (in amps or volts) then simplifies to:

$$(\text{V or I}) = \frac{2A(\tau + t)}{T}$$

where f is less than f_1,

$$(\text{V or I}) = \frac{0.64\,A}{Tf} = -20\,\text{dB/ decade roll off}$$

where f_1 is less than or equal to f, and f is less than f_2, and

$$(\text{V or I})\left[\frac{\text{dBV}}{\text{MHz}} \text{ or } \frac{\text{dBA}}{\text{MHz}}\right] = 20\log\left(\frac{A}{t\,[\mu s]}\right) - 14 - 40\log(f\,[\text{MHz}])$$

where f_2 is less than or equal to f.

Depending on its internal impedance, a circuit carrying a pulse train creates a field in its vicinity which is principally electric or magnetic. At a greater distance from the source, the field becomes electromagnetic, regardless of the source impedance.

If there is a conduction or radiation coupling mechanism, some or all of the frequency components in the digital pulse train's spectrum are absorbed by a "victim" receiver circuit.

To illustrate the extent of the problem, imagine a microcomputer motherboard consisting of a CPU, glue logic, and memory ICs in an unshielded plastic case.

Assume that a number of these ICs toggle states synchronously at a frequency of 100 MHz. Also, assume the total power switched by the circuit at any instant during a synchronous transition is approximately 10 W. Since a real circuit's efficiency is not 100%, a small fraction of this 10 W does no useful work, is not dissipated as heat by the ICs and wiring, but radiates into space.

The power radiated is 10 µW since a reasonable fraction value of 10^{-6} equals the total switched power at the fundamental frequency. If an FM radio is placed 5 m from the motherboard, the field strength E produced by the 10 µW at this distance is approximated by:

$$E = \sqrt{\frac{30\,\text{RadiatedPower[W]}}{\text{Distance [m]}}} = \sqrt{\frac{30\times 10\times 10^{-6}}{5}} = 3.46\left[\frac{\text{mV}}{\text{m}}\right] = 70.79\left[\frac{\text{dB}\mu\text{V}}{\text{m}}\right]$$

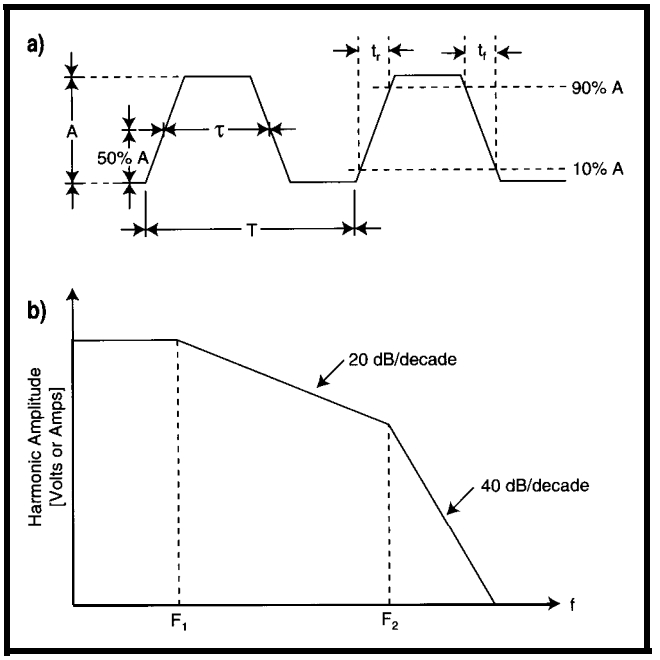


Figure 1—*A* pulse train with the characteristics shown in *(a) produces a spectrum* with an envelope *that can be* approximated *by the* nomogram of *(b).*

Technology	Minimum Voltage Swing [V]	Minimum Transition Time t [ns]	Typical Bit Pulse Width τ [ns]	Equivalent Bandwidth [MHz]	Single-load Input Capacitance [pF]	Output Source Impedance (Low/High)
5-V CMOS	5	70	500	4.5	5	300/300
12-V CMOS	12	25	250	12	5	300/300
HCMOS	5	3.5	50	92	4	160/160
TTL	3	8	50	40	5	30/150
TTL-S	3	2.5	30	125	4	15/50
TTL-LS	3	5	50	65	5.5	30/160
TTL-FAST	3	2.5	25	125	4.5	15/40
ECL	0.8	2	20	160	3	7/7
GaAs	1	0.1	2	3200	1	N/A

Table I--The most popular logic families *have very* different *timing* and *driving* parameters, resulting in radiated emissions spectra with different *characteristics.*

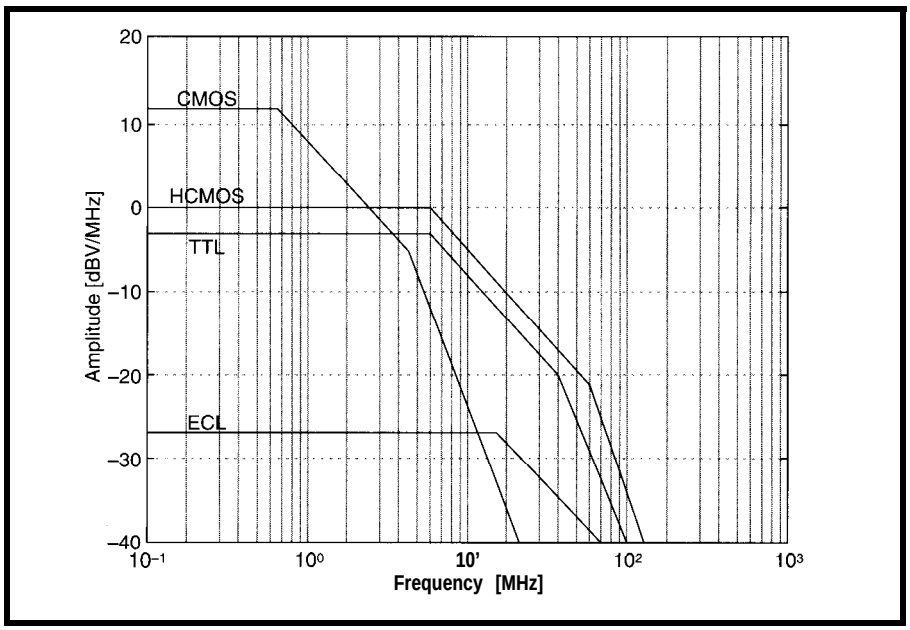


Figure 2—*The* **characteristic voltage spectrum envelope of emissions** *by every sing/e logic gate is high/y* **dependent** *on the technology* **being used.**

Considering that the minimum field strength required for good reception by a typical FM receiver is approximately 50 dBμV/m, the radiated computer clock causes considerable interference to the reception of a same-frequency radio station. In fact, the computer's interference may extend up to 50 m away or more!

It'c easy to see that one reduces radiated emissions by maintaining low clock speeds, slowing rise and fall times as much as possible, and keeping to a minimum the total power per transition.

Transition times and powers depend on the technology used. Table 1 shows how the AC parameters of each technology strongly influence the equivalent radiation bandwidth.

The voltage swing, source impedance, and load characteristics of each technology also determine the amount of power used, and thus the power of radiated emissions on each transition. Figure 2 shows how technology selection is crucial in establishing the bandwidth and power levels of radiated emissions that require control all during design.

ELECTROMAGNETIC FIELDS

EM1 standards require test measurements to be performed at distances of 3–10 m, depending on the frequency range. At these distances, radiated emissions have their electric-field $\vec{E}$ and magnetic-field $\vec{H}$ vectors orthogonal to each other, but in the same plane. Under these conditions, electromagnetic propagation occurs as a plane wave.

If the test probe is brought closer to the device under test, however, the nature of the electromagnetic field changes. Near the source of the radia tion, the field produced is mostly a function of the source's impedance.

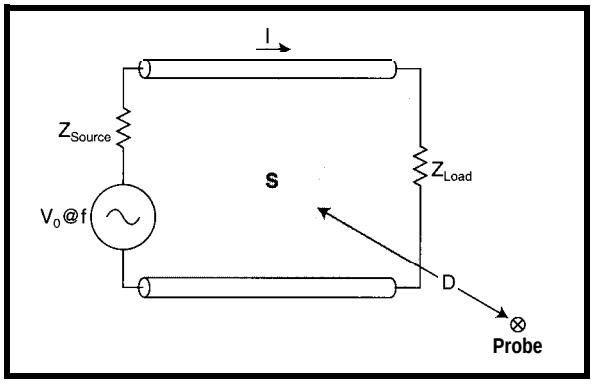


Figure 3-A simplified *but realistic* **model of a** *circuit which* **radiates electromagnetic emissions.** *In it, an AC voltage source causes the flow of a* **current** *I in a* **rectangular loop enclosing** *an area S. The voltage seen by the load depends on the source and load impedances.*

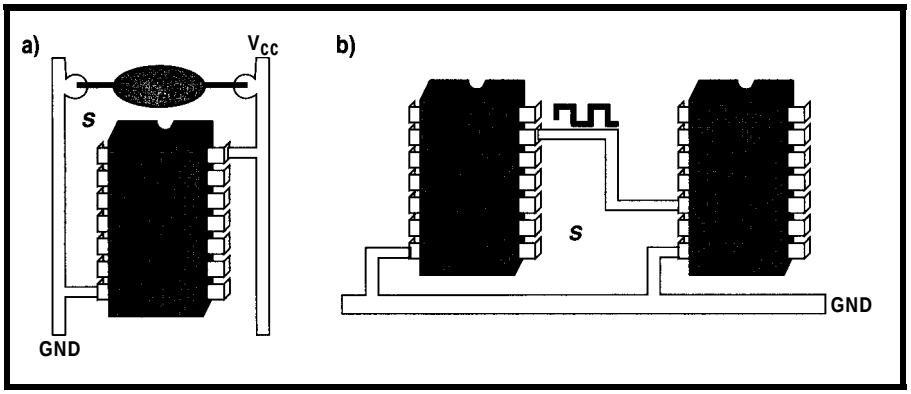


Figure 4-Simple differential-mode radiating-circuitconfigurations are *created when an AC current flows on a current pafh that forms a loop enclosing a certain area S. a)* **Transient** *power demands of an IC are supplied by a decoupling capacitor, causing* brief *strong currents that circulate on a loop formed by the supply-bus PCB tracks. b) Fast digital signals driving low-impedance inputs form EM-radiating loops when current returns through distant ground paths.*

If the field is generated by a high-current, low-voltage circuit, the field is mostly magnetic in nature. If, on the other hand, the field is produced by an element placed at high voltage with little or no current, the field is mostly electric in nature. This is the domain of the near field, while the plane wave is in the domain of the far field.

The ideal generator for a magnetic field (H-field) is thus a circular loop of area $S[m^2]$ carrying an AC current I of wavelength λ.

Note that, although a static field is generated by a DC current and can be calculated with the following method, static H-fields do not cause radiated emissions. They are thus disregarded for EM1 purposes.

If the loop size is smaller than the observation distance D, the magnitudes of the $\vec{E}$ and $\vec{H}$ vectors are found using the solutions derived from Maxwell's equations.

In the near field, the simplified values for these magnitudes are:

$$H\left[\frac{A}{m}\right] = \frac{IS}{4\pi D^3}$$

and

$$E\left[\frac{V}{m}\right] = \frac{Z_0 IS}{2\lambda D^2}$$

where Z_0 equals the impedance of free space, 120π, or 377 Ω.

Inspecting these equations, we find that in the near field, H is independent of λ and decreases drastically with the inverse of the distance cubed. At the same time, the electric field increases as frequency increases, and it falls off with the inverse of the square of distance.

The wave impedance may be defined as the division of E by H because:

$$Z_{wave}[\Omega] = \frac{E\left[\frac{V}{m}\right]}{H\left[\frac{A}{m}\right]}$$

thus, in the near field,

$$Z_{wave} = \frac{Z_0 2\pi D}{\lambda} \qquad (1)$$

where

$$D < \frac{48}{F[MHz]}$$

In the far field, on the other hand, both E- and H-fields decrease as the inverse of the observation distance as described by:

$$H\left[\frac{A}{m}\right] = \frac{\pi IS}{\lambda^2 D}$$

$$E\left[\frac{A}{m}\right] = \frac{Z_0 \pi IS}{\lambda^2 D}$$

which maintains a constant impedance equal to Z_0. You can therefore directly calculate the radiated power density in W/m^2 by multiplying E and H.

E and H, and thus power, increase with the square of frequency. Limiting the bandwidth of radiated signals by a pulse train is therefore of utmost importance in controlling EMI.

The region dividing the near field from the far field is called the *transition region* **(i.e., at** $D = 48/f\,[MHz]$**)**. Abrupt transitions occur in near-field characteristics until a smooth blending leads to far-field characteristics.

Electromagnetic fields are also created by passing an alternating current through a straight-wire dipole, just as with a radio antenna.

In this case, the near-field electric and magnetic vector amplitudes are:

$$H\left[\frac{A}{m}\right] = \frac{Il}{4\pi D^2}$$

and

$$E\left[\frac{V}{m}\right] = \frac{Z_0 Il\lambda}{8\pi^2 D^3}$$

where l is the dipole length in meters.

Contrasting with the near-field H of a loop which falls with the inverse of D^3, the near-field H of a dipole falls off as $1/D^2$. Similarly, the near-field E of a

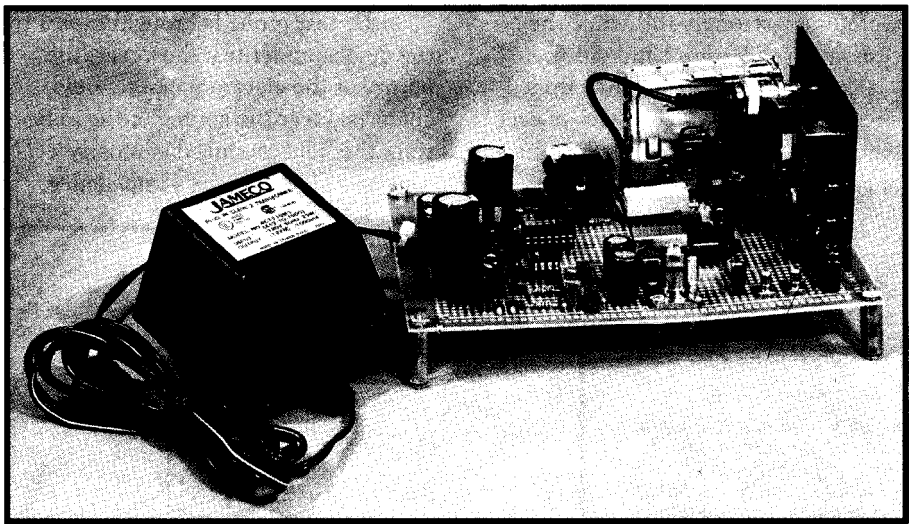


Photo 1-A *simple circuit can convert any triggered oscilloscope info a 100-kHz to 400-MHz spectrum analyzer suitable for near-field EMI sniffing.*

dipole falls off as $1/D^3$, in contrast to that of a loop, which falls as $1/D^2$.

The wave impedance of emissions radiated by a dipole is also affected differently by frequency:

$$Z_{wave} = \frac{Z_0 \lambda}{2\pi D} \qquad (2)$$

Compare Equation 2 with Equation 1. The change in wave impedance as a function of frequency in the case of a dipole is inverse to that of a loop.

In the far field, the behavior of E- and H-fields is again similar to that of electromagnetic radiation from a loop. That is, they decrease as the observation distance increases:

$$H\left[\frac{A}{m}\right] = \frac{Il}{2\lambda D}$$
$$E\left[\frac{V}{m}\right] = \frac{Z_0 Il}{2\lambda D}$$

Beyond the transitional point, the wave impedance again remains constant at the value of Z_0. The result of a constant impedance in the far field means that the ratio of E to H components remains constant regardless of how the field generates.

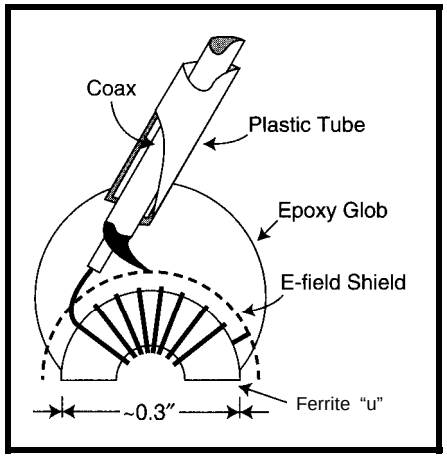


Figure 5-A *useful H-field probe can be constructed from a small, halved ferrite* bead. Approximately 40 *turns of thin enameled wire defect the magnetic flux* concentrated *by the ferrife. A* small *portion of the coax cable* braid *is* used as an *E-field* shield *for the* coil. *The assembly is mounted at the* end *of a small plastic tube which serves as a hand/e and is embedded in a glob of epoxy.*

Of course, real-life circuits are neither ideal open wires nor perfect loops, but hybrids of these two.

In a simplified form, as shown in Figure 3, a more realistic model of a circuit which radiates electromagnetic emissions assumes that an AC voltage source causes the flow of a current I in a rectangular loop enclosing an area S. The source impedance is Z_{source}, and the impedance of the load is Z_{load}, resulting in an overall equivalent impedance of $Z_{circuit} = Z_{source} + Z_{load}$.

In the near field, the electric- and magnetic-field vector magnitudes are given by:

$$E\left[\frac{V}{m}\right] = \frac{V_0 S}{4\pi D^3}$$

where $Z_{circuit}$ is greater than or equal to $7.9\ D[m]f[MHz]$, or

$$E\left[\frac{V}{m}\right] = \frac{0.631Sf[MHz]}{D^2}$$

where $Z_{circuit}$ is less than or equal to $7.9\ D[m]f[MHz]$, and

$$H\left[\frac{A}{m}\right] = \frac{IS}{4\pi D^3}$$

In the far-field, the electric- and magnetic-field vector magnitudes are

given by:

$$H\left[\frac{A}{m}\right] = \frac{35 \times 10^{-6} IS\{f[MHz]\}^2}{D}$$

and

$$E\left[\frac{V}{m}\right] = Z_0 H = \frac{0.013 V_0 S\{f[MHz]\}^2}{D Z_{circuit}}$$

The second lesson of controlling radiated emission leaps out from these equations: keep the area enclosed by loops carrying strong time-varying currents to the minimum possible. Similarly, traces carrying high voltages should be kept as short as possible and be properly terminated.

Besides directing our attention to the parameters affecting radiated emissions, these equations are very useful when designing for compliance with EM1 requirements.

As exemplified by Figure 4, near- and far-field ballpark estimates of EM1 can be obtained from known circuit parameters for a large number of common circuit topologies.

PROBING E- AND H-FIELDS

As shown, the main reason why EM1 standards require testing to be performed in the far field is that a constant impedance in the far field causes the ratio of *E* to *H* components to remain constant regardless of how the field was generated. Hence, measurements can be reproduced with reliability, and standardized methods of testing can be defined with ease.

From the past equations, however, it seems possible to establish a quantitative correlation to estimate far-field values from near-field measurements. Unfortunately, in practice, this is not the case.

Near-field measurements are extremely dependent on the source's exact geometry, the position of the near-field probe, and the interaction between the probe and the source. The variability of these values is too high to enable the exact measurements necessary to calculate radiation behavior in the far-field region.

Although it doesn't predict the outcome of compliance tests, near-field measurements are nevertheless useful for locating potential sources of radiated emissions. Here, near-field qualitative measurements with simple instruments can accurately pinpoint sources of EM1 and identify their basic characteristics.

In essence, if a strong E-field and a relatively weak H-field are detected from a certain circuit section, the culprit can usually be traced to a train of high-voltage pulses on a long wire, an unterminated line, or a trace driving a high-impedance load.

Conversely, if the H-field is strong and the E-field probe is inactive, the source of EM1 is most likely a loop-like circuit where strong currents circulate. Some examples are PCB tracks carrying strong currents, inductors in switching power supplies, and eddy currents induced in metal enclosures by strong internal fields.

Since the same equations describing radiation emission apply to the reception of emissions, it is apparent that a small loop of wire can act as a near-field probe mostly sensitive to H-fields.

In contrast, E-fields are detected preferably by a short exposed wire. Measurements are then taken with a wideband AC voltmeter or a spectrum analyzer.

Even a simple single-turn wire loop at the end of a coax cable can be an effective H-field probe. With this arrangement, maximum output from the probe is recorded when the loop is

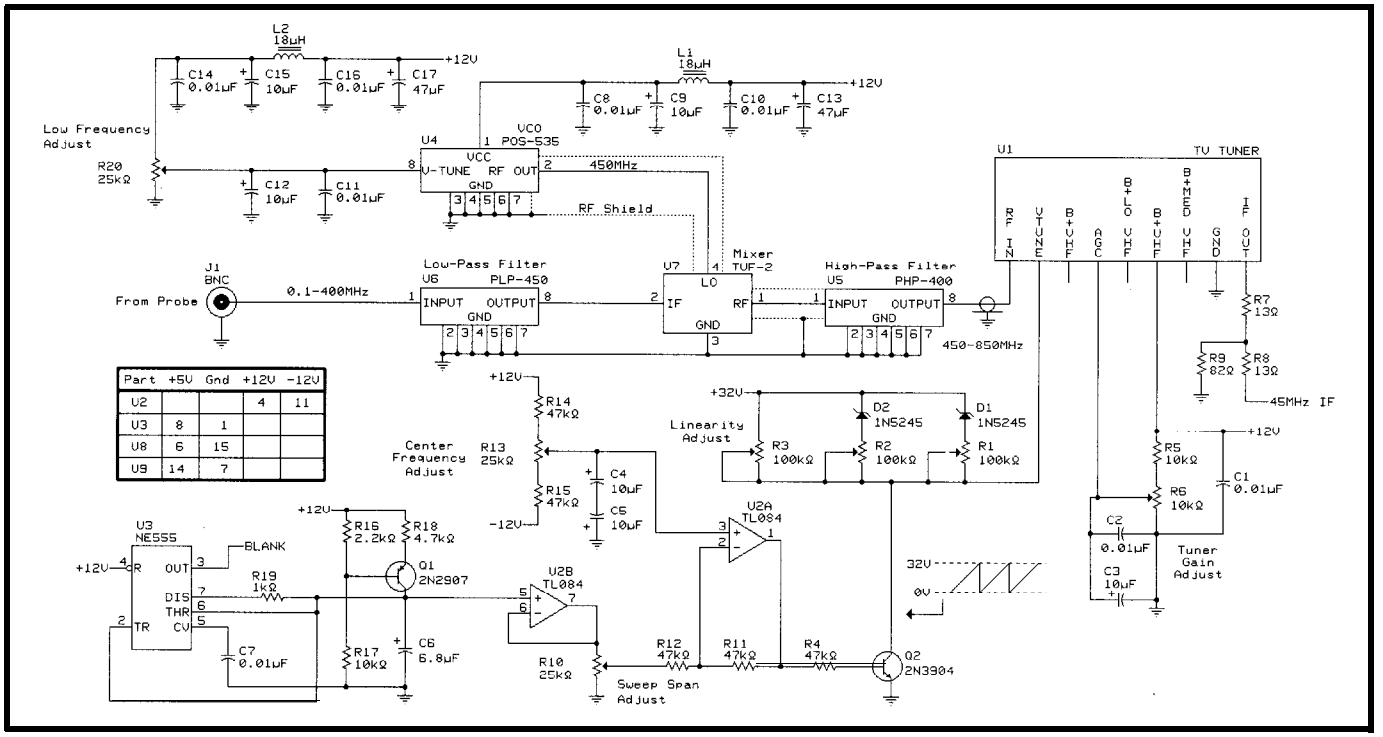


Figure 6—*A varactor-based* **IV** *tuner is the heart of a simple spectrum* **analyzer.** *100-kHz* **to** *400-MHz* **signals from a sniffing** *probe are upconverted to the 450–850-MHz* **UHF band,** *where the tuner can be* **swept** *by a sawtooth* **waveform. The tuner produces** *a 45-MHz* **intermediate frequency which can** *be processed to derive the input signal spectrum. Direct connection* **of** *the probe to the tuner input extends the range of the spectrum analyzer to the high-VHF/UHF region (450–850 MHz).*

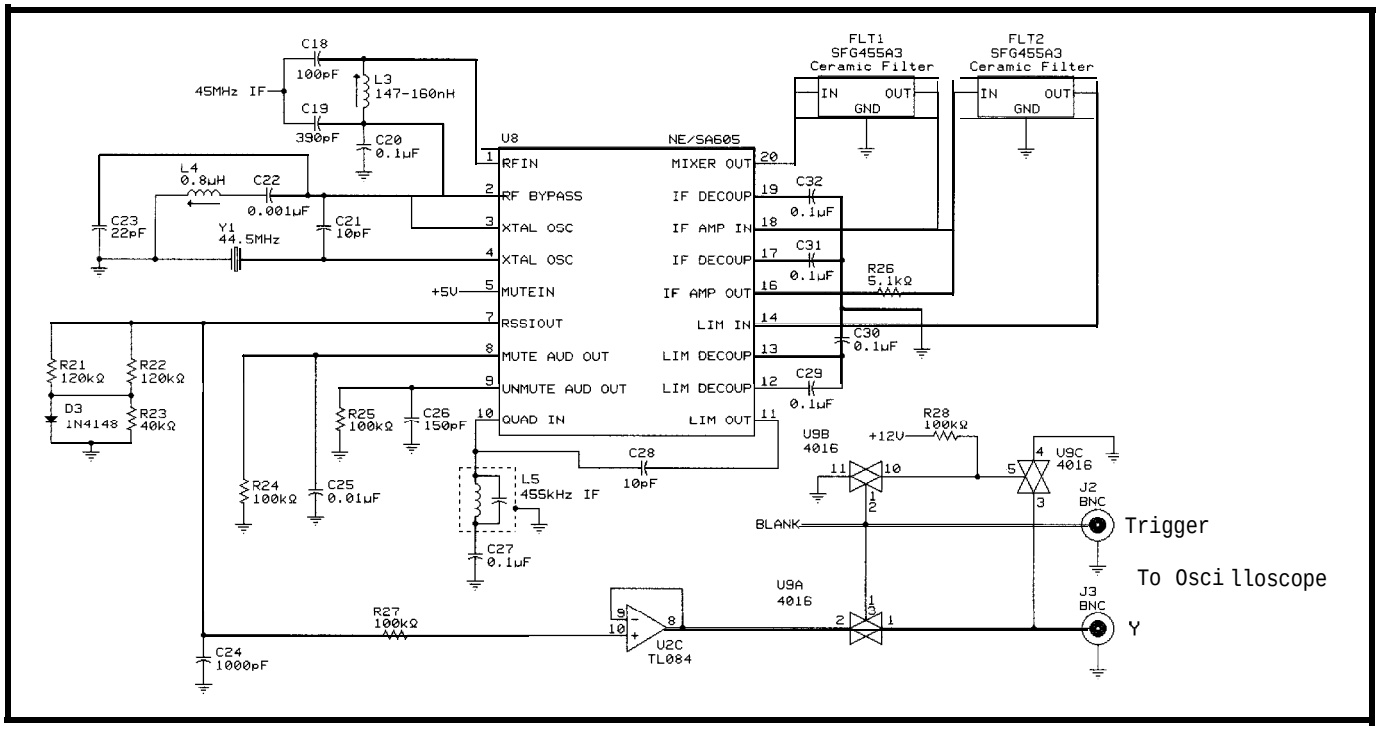


Figure 7-The intermediate frequency output *of the tuner is detected by U8, a sing/e-chip IF* **processor.** *The Received Signal Strength Indicator (RSSI) output, as a function of the sawtooth signal driving the tuner, is a logarithmic representation of the spectrum of the* **signal** *picked up by the probe.*

in immediate proximity and aligned with a current-carrying wire. This directionality is very useful for pinpointing the exact source of a suspicious signal.

The diameter of the loop makes a large difference on H-field measurements [4]. The area enclosed by the loop influences the sensitivity of the probe since it determines the number of magnetic-flux lines which are intercepted to produce a detectable signal.

A larger loop obviously develops a larger voltage at the input of the voltmeter or spectrum analyzer. On the other hand, larger loops have inherently larger self-inductance and equivalent capacitance than smaller loops.

As inductance increases, the network formed with the complex impedance of the measurement setup resonates at lower frequencies, beyond which the probe cannot be used. Moreover, larger loops make it much more difficult to identify the exact source of an interfering signal. Their size does not allow them to selectively pick up radiations from single lines when a multitude of the latter are clustered close together.

Coils with multiple turns increase sensitivity without appreciably increasing the physical size of the coil. However, this solution results in reduced spectral response due to increased self-inductance.

Loop geometry must therefore be chosen each time by compromise. It's a good idea to keep a variety of probes

handy to tackle different problems.

Another convenient H-field probe can be constructed similar to AC-current tongs. In this case, a magnetically permeable material concentrates the magnetic-flux lines created by the circuit under test.

The resulting magnetic flux is detected by a coil with multiple turns. If the tongs completely enclose the conductor in which current is flowing, the voltage developed across the coil is proportional to the vector sum of the conductor's currents.

This is, of course, impractical for the needs of sniffing H-fields. A structure with open-ended tongs is more suitable for probing a circuit without modifying it.

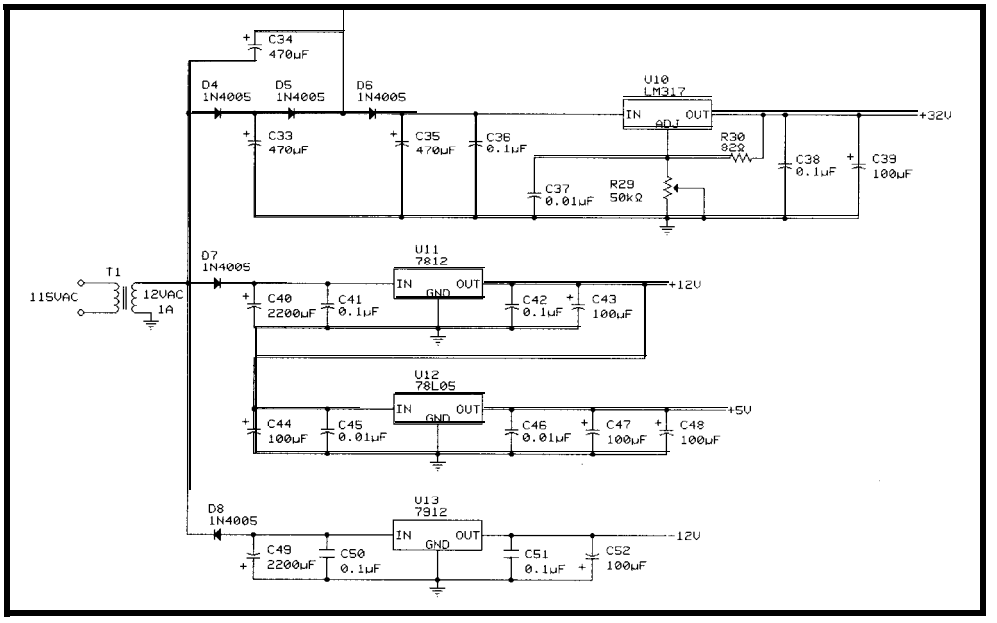


Figure 8—*DC power for the various circuits of the spectrum analyzer is derived from a single 12-VAC input.*

The probe can be built as shown in Figure 5, using a small ferrite bead (e.g., 0.1" thick, 0.3" outer diameter) cut in half. The construction depends on the ferrite selected, but generally, 40-50 turns of thin enameled copper wire provides suitable sensitivity.

Solder the coil terminals to the center and shield of a coax cable. After insulating the central conductor connection, use a portion of the braid to cover the assembly. This shields the coil from the E-field.

The assembly can then be mounted at the end of a small plastic tube and embedded within a glob of epoxy. For the prototype probe, I measured a virtually flat bandwidth from approximately 600 kHz to 10 MHz.

Better bandwidth is achieved with a VCR magnetic head. Video heads detect broadband magnetic fluctuations. For this reason, they are useful for sniffing H-fields from 2 to 120 MHz with relatively flat response.

To construct the probe, carefully remove one of the magnetic heads from a discarded drum. Even a worn-out head works well.

Soiled heads should be cleaned with a swab and pure alcohol. Degaussing also improves an old head's sensitivity. All other aspects of constructing and using this probe are the same as for the ferrite-bead probe.

For E-fields, the simplest near-field probe is a coax cable in which a short segment of the center conductor extends beyond the braid at the unterminated end of the coax. Similar to the loop probe, a longer wire picks up a stronger signal at the expense of specificity and bandwidth.

In general, select a wire length which gives a sensitivity of approximately 3 mV/m. At this level, potentially problematic emissions can be identified without causing undue concern about low-level emissions.

Constructing the ideal H- or E-field probe for a specific job takes some trial and error, since the effort of electromagnetic modeling required for proper design is overkill in most applications.

You may nevertheless want to check whether a probe resonates within the desired spectral range. Connect a wideband probe to an RF generator set to track the tuning frequency of a wideband spectrum analyzer. The probe should be located in close proximity to the emitting probe, and connected to the spectrum analyzer's input. The limit of the probe's useful bandwidth is the point at which the first abrupt resonance appears.

Before plugging the spectrum analyzer into the powerline, however, the first step in conducting a near-field EM1 study is to draw the assembly's component placement diagram. The diagram should indicate circuit points identified in the mathematical circuit harmonic analysis as potential sources for EM1 radiation. Only after this preliminary work should you try bench testing.

Conduct a coarse near-field sweep at relatively high gain to identify EM1 hot spots. Log the frequencies at which strong components appear when scanning the unit under test. Detailed scanning, using a more discriminating probe, can then concentrate on the hot spots identifying the culprit circuit generating offending emissions.

You gain a valuable source of clues for future troubleshooting by printing the spectral estimate at each point the measurements highly agree or strongly disagree with the circuit's harmonic analysis.

In any case, keep detailed and organized notes of the near-field scans. They prove invaluable when attempting quick fixes while the clock is running at the far-field compliance-testing facility.

A BARE-BONES SPECTRUM ANALYZER

While an AC voltmeter indicates the field strength a probe is exposed to, it doesn't indicate an emission's spectral content. A spectrum analyzer is a tool that certainly can't be beat in the search for offending signals.

Unfortunately, spectrum analyzers are often beyond tight budgets. For

near-field sniffing, however, even the crudest spectrum analyzer does a magnificent job.

Photo 1 shows a simple home-brewed adapter that converts any triggered oscilloscope into a spectrum analyzer that gives qualitative spectral estimates of 100 kHz to 400 MHz.

As shown in Figure 6, a voltage-controlled TV tuner U1 forms the basis of the simple spectrum analyzer. Most any voltage-controlled tuner works. You may be able to get one free from a discarded TV or VCR PCB. The connection points and distribution vary from device to device, but the pinout is usually identified by stampings on the metallic can of the device.

Varactor-controlled TV tuners receive signals on their RF input at a frequency determined by the voltage applied to the VTUNE input. With power applied to the UHF section of a tuner, typical control voltages between 0 and 32 V span a frequency range of approximately 450-850 MHz.

Tuner sensitivity can be adjusted through the AGC input. The output of the tuner is a standard 45-MHz IF.

However, the 450–850-MHz range is not directly applicable to the large bulk of EM1 sniffing. For this reason, a more appropriate range of 100 kHz to 400 MHz is converted up to the tuner's input range through a circuit formed by U4–U7.

Here, signals from the probe are low-pass filtered by U6 and injected into the IF port of a TUF-2 mixer. The LO input of the mixer is fed with the output of U4, a self-contained voltage-controlled oscillator tuned to 450 MHz by potentiometer R20.

The RF port of the mixer outputs signals with frequency components at the sum and difference between the IF input and the LO frequency. This output is high-pass filtered by U5 to ensure that only up-converted components are fed to the tuner input.

Sweeping the tuner across its range is accomplished by a sawtooth waveform which spans approximately 131 V. The basic sawtooth is generated by U3 and Q1, and buffered by U2b. The span of the sawtooth is set by attenuator R10, while the center of the sweep is adjusted by introducing an offset on U2a by means of R13.

The output of U2a is amplified by transistor Q2, which should be selected for a gain of 50 or less. The final span and linearity of the sweep is adjusted in three ranges by R1, R2, and R3.

The IF output of the tuner is attenuated to a level suitable for processing by the circuit in Figure 7. Select the actual value of the resistors for this attenuator based on the output level of the specific tuner you use. U8, an NE/SA605 single-chip IF processor, detects the signal and produces a logarithmic output of signal strength.

In this portion of the circuit, the 45-MHz IF signal is coupled to the input of a RF mixer internal to U8 via a tuned circuit formed by C18, C19, and L3. The LO input of this mixer is fed from a 44.5-MHz crystal-controlled oscillator. The resulting 455-kHz IF is filtered by two ceramic filters, FLT1 and FLT2.

An internal Received Signal Strength Indicator (RSSI) circuit is used as a detector and linear-to-logarithmic converter. The RSSI output, as a function of the sawtooth signal driving the tuner, is thus a logarithmic representation of the signal's spectrum picked up by the probe.

RSSI is a current signal which requires conversion to a voltage by the network formed by R21–R23 and D3. C24 low-pass filters the RSSI output to produce a smooth display, and U2c acts as a buffer and impedance transformer for the current-to-voltage converter. Finally, U9a blanks the output during retrace.

Figure 8 presents the power-supply circuit for the adapter. The +12-V power supply powers most of the circuitry, including the upconverter, tuner, and sawtooth generator. The IF processor is powered by +5 V.

The +32 V to drive the tuner's varactors is obtained by stepping up the 12-VAC input to +48 V and then reaching the desired voltage through U10, an LM317-adjustable linear regulator.

To operate the spectrum analyzer, the Y output of the adapter is con-

nected to the vertical input of the oscilloscope, and the TRIGGER output is connected to the trigger synchronization input of the scope.

The horizontal frequency of the oscilloscope is set so one full sweep caused by the sawtooth fits the full graticule on the oscilloscope's screen. Fine-tune it by either trimming the time base of the scope or by appropriately adjusting the value of R18.

Alternatively, a two-channel oscilloscope can be operated in the X-Y mode by injecting the sawtooth available at pin 7 of U2a to the appropriately scaled x-axis channel.

The comb generator circuit in Figure 9 calibrates the adapter. The circuit is simply a TTL-compatible 40-MHz crystal-controlled oscillator module feeding a synchronous binary counter.

It is called a comb generator because the spectral pattern of any of its outputs resembles a hair comb with its prongs pointing up. Because these spectral components occur at harmonic multiples of the selected fundamental square-wave frequency, it follows that the frequency difference between consecutive prongs must be the same as the value of the fundamental frequency of the square wave.

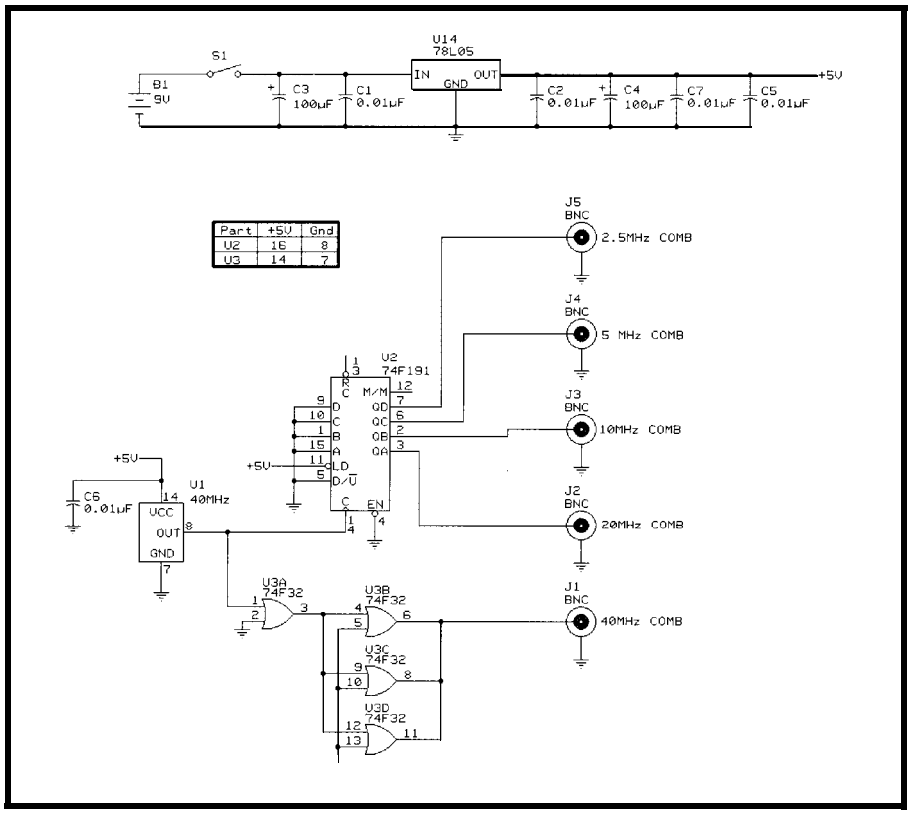


Figure 9—*High-frequency* **clocks and fast logic generate broadband signals extending** *well* **info** *the hundreds-of-megahertz region.* **This generafor** *produces various comb patterns which he/p calibrate spectrum analyzers.*

The graph in Figure 10 presents the pattern obtained when the 20-MHz comb output of the generator is probed by a commercial-grade spectrum analyzer.

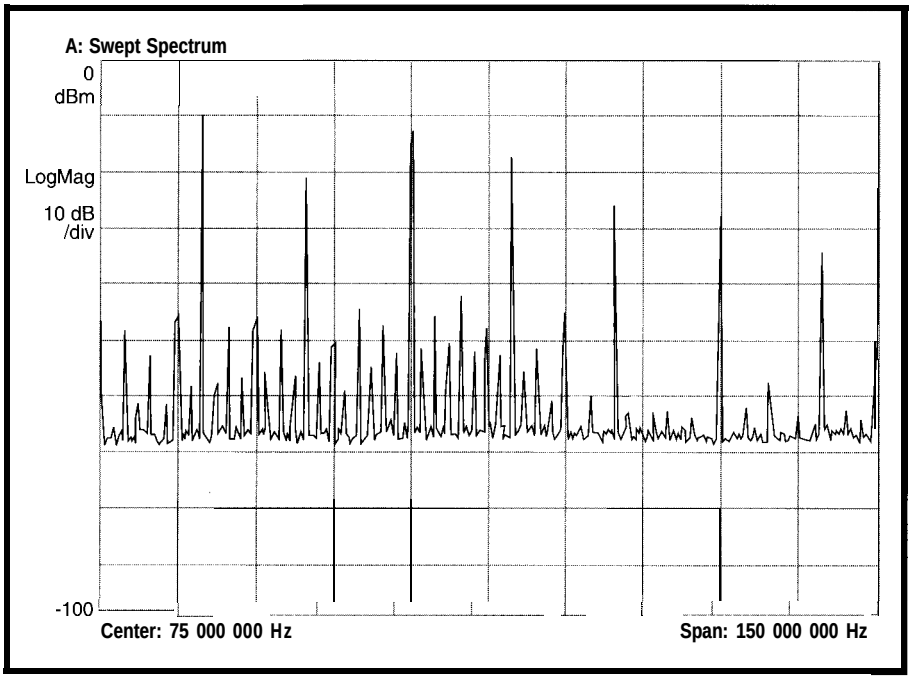


Figure 1 *O-The spectra/pattern obtained from the output of the comb generator serves as a frequency ruler because* **if presents** *strong spectral lines at every harmonic of the fundamental square wave. Notice the similarity between the envelope formed by the spectral components of this 20-MHz comb and the nomograms of Figures 1 and 2.*

I accomplished AC coupling through a series-connection 15-pF capacitor. The capacitor's output side is terminated to ground through a 50-Ω noninductive resistor. This is the gold-standard against which the adapter should be calibrated.

Start testing the adapter by setting the sawtooth generator to vary the voltage at the VTUNE input of the tuner between 1 and 31 V. Initially, set R6 to apply 2.5 VDC to the AGC pin of U1.

Adjust the upconverter LO frequency to 450 MHz by trimming R20. At U4's VTUNE input, 9.6 VDC typically results in the desired LO frequency. L4 should be trimmed to achieve stable oscillation of the 44.5-MHz IF LO oscillator.

With a 40-MHz comb applied to the input of the adapter through a 15-pF coupling capacitor and with 50-Ω termination, adjust L3 to approximate the expected 40-MHz comb pattern on the oscilloscope.

After achieving a satisfactory display for the 40-MHz comb, calibrate the linearity of the adapter using a 20-MHz comb by first trimming R3 to

produce equal spacing between spectral lines throughout the lower third of the display. Then, linearize the mid-range by trimming R2, and finally the high range by trimming R1.

PASSING THE TEST

Designing equipment that passes EM1 and EMC compliance testing without fixes or delays never happens by mistake. Rather, it involves considering compliance with EM1 and EMC regulations from the very beginning of product formulation and design.

Developing a first prototype free of foreseeable trouble gives a head start in the battle against EMI. Such development is possible by carefully selecting the technologies which fulfill the product requirements while minimizing EMI, using good design and construction practices, and making extensive use of circuit simulation tools.

Near-field probing of the first prototype should reveal real-world EM1 effects that escape from the limited view of initial modeling. Correcting any problems through filtering, shielding, or redesign is inexpensive at early design stages. The second prototype already has a good chance of passing compliance testing with minimal rework.

This article presented only a few of the ways in which technology selection, circuit design, and layout techniques influence the generation of EMI. Many more books [5,6,7] and articles [8] disclose the secrets of the EM1 and EMC world, all the way from Maxwell's equations, through the legalities of regulation, and into the tricks of the trade for taming EMI.

Considering the stiff economical, technical, and legal penalties brought by manufacturing a product that does not comply with EM1 and EMC regulations, you should be motivated to keep EM1 in sight at every turn of the design process.

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

REFERENCES

[1] G. Dash and I. Strauss, "Inside Part 15: Digital Device Approval," ***Compliance Eng., 1995*** Annual Reference Guide, Al 1–A18.

[2] R.W. Gubisch, "The European Union's EMC Directive," ***Compliance Eng., 1995*** Annual Reference Guide, A55–A64.

[3] G. Dash and I. Strauss, "How the FCC Enforces Part 15," ***Compliance Eng., 1995*** Annual Reference Guide, A21–A27.

[4] V. Kraz, "Near-Field Methods of Locating EM1 Sources," ***Compliance Eng., 43-51,*** May/June 1995.

[5] M. Mardiguian, ***Con trolling Radiated Emissions by Design,*** Van Nostran Reinhold, New York, NY, 1992.

[6] M. Mardiguian, ***Interference Control in Computers and*** *MicroprocessorBased* ***Equipment,*** Don White Consultants, Gainesville, VA, 1989.

[7] ***C.*** Marshman, ***The guide to the EMC Directive,*** IEEE Press, Piscataway, NJ, 1992.

[8] I. Strauss and G. Dash, Various Articles on "Design for Compliance," ***Compliance Eng., 1995*** Annual Reference Guide, Al 15–A154.

SOURCES

POS-535, PLP-450 and PHP-400
Mini-Circuits
P.O. Box 350166
Brooklyn, NY 11235-0003
(718) 934-4500
Fax: (718) 332-4661

NE/SA605
Philips Semiconductors
811 E. Arques Ave.
Sunnyvale, CA 94088-3409
(408) 991-2000
Fax: (408) 991-2311

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

Easy-to-Use Serial EEPROMs

FEATURE ARTICLE

Ingo Cyliax

There seem to be as many flavors of serial EEPROMs as there are manufacturers. Interfaces using 2, 3, and 4 wires abound. Now, Atmel has introduced a family of EEPROMs that does it all.

Volatile FPGAs and CPLDs traditionally use serial OTP PROMs to store their hardware configuration (also called the configuration bit stream).

However, Xilinx has their own line of OTP serial PROMs for their FPGAs. These FPGA-compatible serial PROMs are special because they do not impose any special protocol when downloading their contents into the FPGA.

Atmel has developed a line of FPGA configuration memories which are EEPROM-based and FPGA-compatible. They work with Xilinx FPGAs (XC4000 and XC3000), AT&T's ATT3000 FPGAs, and Atmel's own line of FPGAs (AT6000).

Over the last year, I've worked with several design projects involving SRAM-based FPGAs (Xilinx XC3000 and XC4000). Since these FPGAs are volatile, they have to be initialized with an onboard CPU or a serial PROM every time power is applied to them. During debugging, it is often convenient to use a download cable directly connected to a computer hosting the FPGA CAD environment.

While these methods are the most flexible and easy way to implement and test programmable hardware in a system, it just wasn't enough. Situations arose where I wanted to use serial EEPROMs to configure the FPGAs, but standard serial EEPROMs are incompatible with FPGAs.

Enter Atmel's series of AT17Cxxx serial EEPROMs, designed to interface with FPGAs. Great, I thought-another useful device that's hard to get in small quantities. As it turns out, a call to my nearest Atmel vendor produced a tube of these devices at a price comparable to the OTP version of the FPGA-configuration PROMs.

Now, with the EEPROMs in one hand and properly formatted bitstream on a floppy in the other, I approached our device programmer only to find out that this type of EEPROM is not supported (yet).

After obtaining the appropriate data sheets from Atmel's automatic fax line and spending a day hacking together a simple interface and a basic program to program these devices, I successfully wrote a configuration EEPROM which initialized my FPGA design on the first try.

For commercial applications, I always recommend using a programmer which has been certified by a vendor to

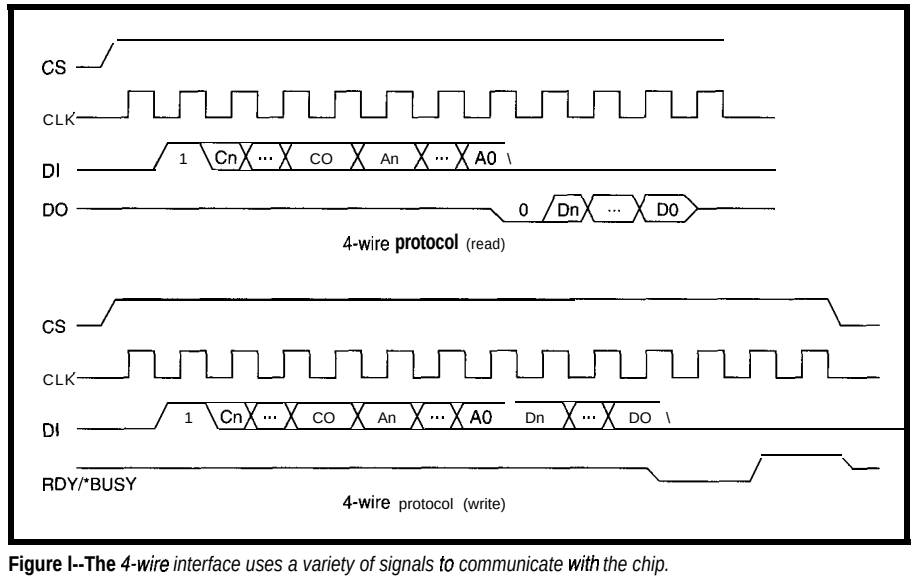


Figure 1--The *4-wire* interface uses a variety of signals to communicate with the chip.

Instr.	Opcode	Address	Data	
READ	1 10xx	A6-A0		Readdataataddress
EWEN	1 0011	xxxxxxx		Enable write mode
WRITE	1 xlxx	A6-A0	D7 -DO	Write byte at address
ERAL	1 0010	xxxxxxx		Eraseallmemory
WRAL	1 0001	xxxxxxx	D7-DO	Writebytealladdresses
EWDS	1 0000	xxxxxxx		Disable write mode

Table 1--Some typical *4-wire* commands including *the* enable *(EWEN)* and disable *(EWDS) write* mode commands.

work with their devices. However, most EEPROM and flash devices have their own on-chip programmer and programming voltage generator which make programming them not as critical. Atmel has a low-cost programming solution for these parts.

vary by the type of interface used (4-, 3-, and 2-wire) and internal organization. Some devices are page oriented (i.e., they only write in fixed-size pages, typically 16-64 bytes) and come in 8- or 16-bit internal word size.

Serial EEPROMs have bit-serial interfaces that connect to a parallel EEPROM structure. They contain a serially loaded address register and shift registers which clock data values in and out of the memory array over the serial data interface. Some devices also autoincrement the address register for sequential read and write page operations.

straightforward. In many cases, they hold configuration information and—since large-capacity serial EEPROMs have become available-even executable code.

The BASIC Stamp uses serial EEPROMs to hold the user's compiled basic code. Peripheral cards use them to store soft configurations, such as I/O port base address and IRQ level, to reduce the number of jumpers. Using surface-mount, small-capacity serial EEPROMs to replace jumpers is cheaper and requires less board space, which is very important in high-volume applications.

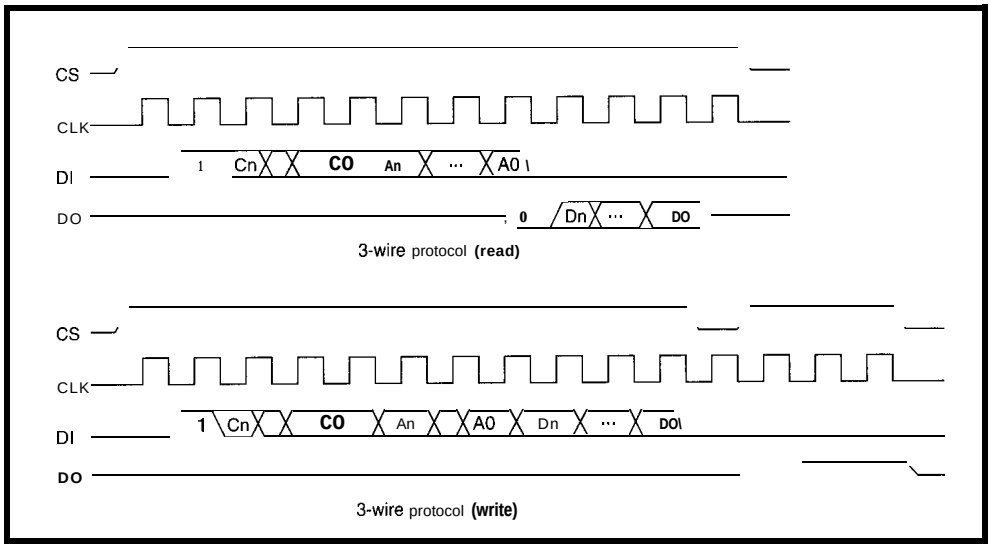


Figure 2—*The 3-wire* interface eliminates one signal *(ready/*busy)* from the *4-wire* interface by overlaying the *function* on *the data-out* signal.

Before I go on about the construction and programming details, let me describe some of the generic serial EEPROM techniques.

SERIAL EEPROMS

These serial EEPROMs feature a serial data path, rather than a byte- or word-wide data path. The serial data path greatly reduces the pin count and thus the package size and cost at the expense of a slower interface. Internally, these devices still feature a parallel architecture (8 or 16 bits) and, like their parallel cousins, also feature on-chip programming voltage generators. They can operate from a standard S-V (or even lower) power supply.

Serial EEPROMs come in many memory sizes (128 bits to 256 Kb) and

USES AND APPLICATIONS

Serial EEPROMs are useful in many situations where parallel EEPROMs and flash memories are overkill and expensive or where PCB real estate is prime. They are very easy to interface to CPUs and interfacing them to hardware (ASICs and FPGAs) is straightforward. In many cases, they hold configuration information and—since large-capacity serial EEPROMs have become available-even executable code.

The BASIC Stamp uses serial EEPROMs to hold the user's compiled basic code. Peripheral cards use them to store soft configurations, such as I/O port base address and IRQ level, to reduce the number of jumpers. Using surface-mount, small-capacity serial EEPROMs to replace jumpers is cheaper and requires less board space, which is very important in high-volume applications.

PROTOCOLS

When talking about protocols, the EEPROM is the slave and the CPU or device trying to read from and write to the EEPROM is the master. The most common protocols used with serial EEPROMs are 4-, 3-, and 2-wire protocols.

The number of wires refers to how many signals are needed to communicate with the EEPROM. The 4-wire and 3-wire protocols are very similar, while the 2-wire protocol is a little more complex and is used to program and verify the AT-17C128.

The master device selects the chip with a chip-select line and possibly some external address lines. After the slave is selected, the master uses the clock to strobe command and address information to the slave.

When performing a write operation, the master also sends the data to the slave following the command and address. During a read, the master reads the data back from the chip while strobing the clock line.

Instr.	Opcode	Address	Data	
READ	1 10	A6-A0		Read data at address
EWEN	1 0011	xxxxx		Enable write mode
ERASE	1 11	A6-A0		Erasebyteataddress
WRITE	1 01	A6-A0	D7-DO	Write byte at address
ERAL	1 0010	xxxxx		Eraseallmemoty
WRAL	1 0001	xxxxx	D7-DO	Write byte all addresses
EWDS	1 0000	xxxxx		Disable write mode

Table 2—*The 3-wire* commands *are very similiar* to the *4-wire* commands.

Since write and erase operations may take several milliseconds to complete, the slave may need to signal the master that it is busy performing a write or erase cycle and can't respond to any requests until it's done.

In a system, there is minimally one master and one slave device. However, several masters and slaves can be present on a single bus. If more than one master is present, some kind of bus-access arbitration has to be used. All the devices on a bus share the wires that are defined in the interface used.

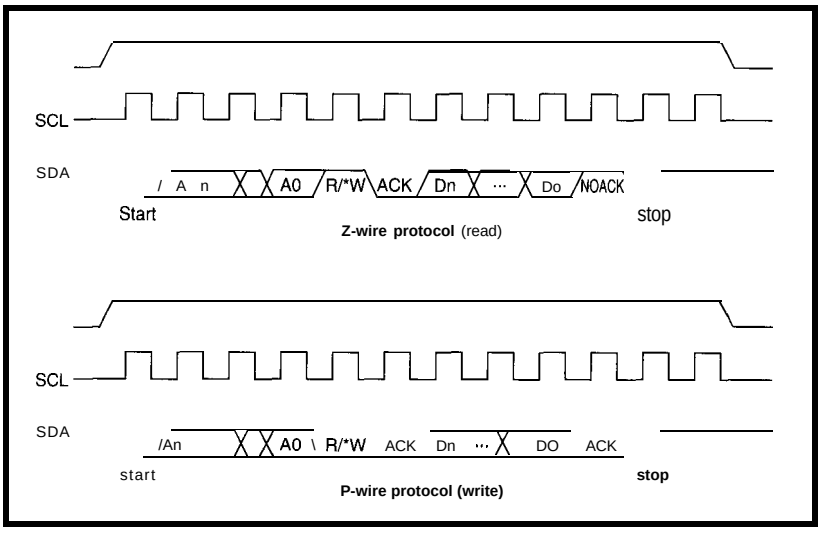


Figure 3—*By making the data bus bidirectional, the P-wire interface only requires two signals to communicate with the chip.*

WIRE PROTOCOLS

The 4-wire protocol uses four interface signals, the serial clock line, the data-in line, the data-out line, and a line to signal if the chip is busy programming. The serial clock is provided by the master (or the system). All data is latched synchronously to the rising edge of the clock.

The chip uses the READY/*BUSY line to signal the master that it cannot respond to any operations. The master either waits until the slave is done writing, in which case it drives the READY/*BUSY line high, or releases the chip and tries again later (see Figure 1).

The data-in line receives the opcode and address from the master and, if writing to the chip, the data to be written. The data-out line is used to send read data back to the master. The slave typically sends a 0 bit before the data to indicate the start of the data byte.

The 4-wire devices implement a rich set of operations besides reading, such as write a byte, write all memory, and erase one byte. To protect against accidental writing, these devices have to be write enabled (by sending an opcode) before they can be programmed or erased. Table 1 shows some of the opcodes implemented by the AT59C11 device.

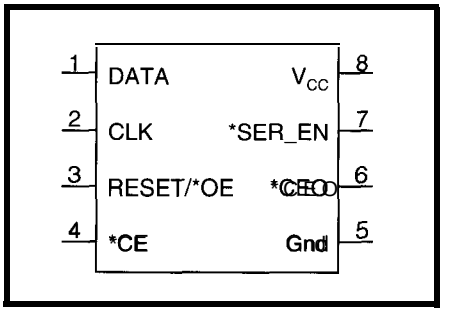


Figure 4—*The 8-pin DIP pinout for a AT17C64/128/256 chip. This chip is pin-compatible with the Xilinx OTP serial-configuration PROMs.*

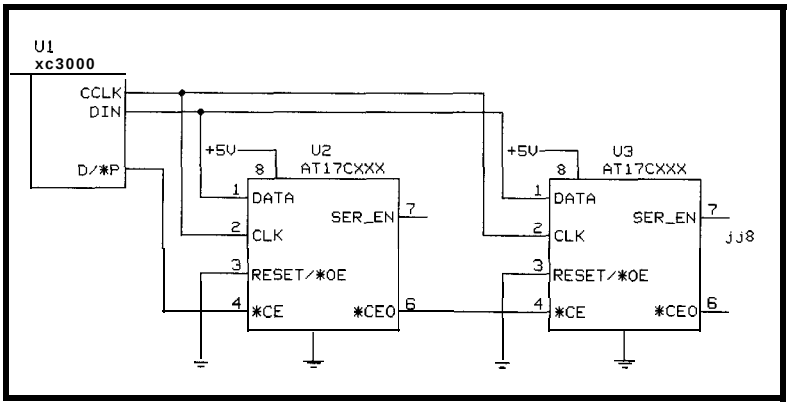
The 3-wire protocol eliminates the READY/*BUSY line and signals a busy condition by driving the data-out line low in response to the chip-select condition when busy. The same protocol is used when reading and writing data over the data-in and data-out lines. The 3-wire operations are very similar to the 4-wire parts as shown in Figure 2. Table 2 shows the opcodes used by the AT93C-46.

Finally, the 2-wire protocol uses a bidirectional data bus to communicate with the chip. The clock behaves a little differently. The rising edge still latches the data into the slave. However, the falling edge is used by the master to latch the data sent by the slave.

In addition, the start and stop conditions are signaled by making the data-line transition while the clock is held in a high state. A start condition is a high-low transition, while the stop is signaled by a low-high transition. In Figure 3, the device receiving the data acknowledges each byte by pulling the data line low during a ninth clock cycle.

The commands implemented by 2-wire chips differ from the 4- and 3-wire devices. Read, read sequential, and write page are usually the only commands implemented. While 4- and 3-wire devices implement byte write operations, 2-wire devices are written one page at a time. The page size is between 4 and 64 bytes.

A page write is accomplished by sending the beginning address of the page to be written and then sending all of the data that is necessary to write a page. If less than one page of data needs to be written, the master must first read the page into its memory and then write out the page with the appropriate changes.



'igure 5-*A typical system configuration using a daisy-chain of serial configuration PROMs in a FPGA design.*

1	0	1	0	A2_pin	1	1	R/*W	ACK	0	0	A13...A8	ACK	A7...A0	ACK

Figure 6—*The command structure of the ATR17Cxxx is similiar to standard 2-wire interface devices. There is a single-byte control followed by a two-byte address when writing.*

The stop condition, if data has been sent after the command and address byte, initiates the program cycle. During programming, the device doesn't respond to any requests from the master.

ATMEL AT17C*XXX* EEPROM

This device is a serial EEPROM specially developed to emulate the kind of OTP PROMs normally used with FPGAs. It comes in three sizes: the AT17C65 (65 Kb), AT17C128 (128 Kb) and AT17C256 (256 Kb). The memory is organized in 8-bit bytes with 64-byte pages.

AT17Cxxx parts are available in 8-pin DIP and various surface-mount packages. Reset polarity (RESET/*OE or *RESET/OE) is programmable for maximum flexibility. Finally, the best feature is the ability to select which protocol the chip uses [i.e., FPGA mode or 2-wire] (see Figure 4).

In FPGA mode (*SER_EN high), the DATA pin is enabled as output when RESET/*OE is low and the device is selected (*CE low). The FPGA uses the CLK line to shift out the contents of memory on the DATA pin. The internal address register starts at address 0 after powerup and whenever the RESET/*OE line goes high.

While reading, the address counter autoincrements after each byte is read. If the address counter overflows, the DATA pin goes into tristate condition, and the *CEO is driven low. Observe in Figure 5 that this change allows chaining of multiple devices by daisy-chaining the *CEO and * CE outputs and inputs and busing the DATA, CLK and RESET/ • OE pins.

In 2-wire mode (● SER_EN low), some of the pin functions change. The DATA pin is now the bidirectional signal, and the *CEO is called the A2 pin. In Figure 6, the level of the A2 pin is compared to one of the bits in the control byte sent to the chip.

RESET/* OE and • CE must be low for reading and programming the chip. The protocol is now the standard 2-wire protocol as described before. The master sends a control byte followed by two address bytes (high and low order).

The chip understands two commands, read and write page, which are signaled by the R/* W bit on the control byte. If the chip is ready and has interpreted the control byte and address bytes correctly, it responds with a standard 2-wire ACK to each byte. If the master wants to write (R/*W=O), it sends up to 64 data bytes after the second address byte. When reading (R/* W=1), the chip responds by sending bytes to the master until it receives a NAK (i.e., a 1 in the ninth bit) and a Stop condition.

PC-PARALLEL PORT

Serial EEPROMs usually consume very little power. In the case of the AT17Cxxx, it's less than 10 mA. We can therefore use the data lines on the PC parallel port to directly power the chip. To make this more reliable, two signal lines drive the V_{CC} pin.

Since the data line of this device is bidirectional in 2-wire mode, we need to interface this to the PC printer port. We do this by adding a resistor in series with one of the data lines. Now, the PC-port data line is only able to weakly pull the DATA pin high or low. If the device wants to drive the DATA line, it overrides the parallel port.

One of the status lines on the PC printer port senses the state of the DATA pin. The *CEO pin receives the same interface, since it is an input pin in 2-wire mode and an output pin in FPGA mode. As you see in Figure 7, the remaining pins are wired directly to the data lines of the parallel port.

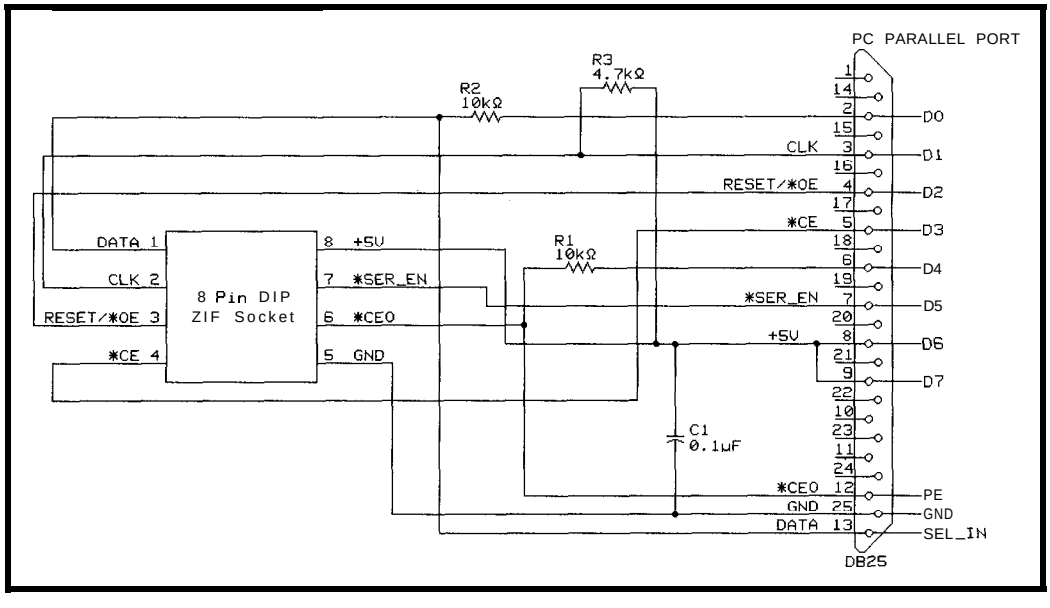


Figure 7-A *simple* **PC** *parallel port serial EEPROM programmer for programming A T17Cxxx serial configuration EEPROMs. Since the the chip draws less than 10 mA, power can be scavanged from the unused data lines on the parallel port*

Listing 1—After setting the start address, the contents of the chip is read sequentially with readbyte (see Listing 3).

```
x=cndstart:  x=writebyte(&HA6): x=writebyte(&H0): x=writebyte(&H0)
x=cndstart: x=writebyte(&HA7)
FOR i = 0 TO 4095
      pdat(i) = readbyte
NEXT i
x = cndstop
```

Listing 2-Using the writebyte function in Listing 3, a 64-byte page is programmed by first setting the address and then writing the data sequentially into the chip. A stop condition initiates the programming cycle on the chip, which may fake as long as 10 ms to complete. The last WHILE loop prints '.' until the programming cycle is finished.

```
x = 1
WHILE x = 1
     x = cndstart: x = writebyte(&HA6)
     IF x = 1 THEN PRINT ".";
WEND
x = cndstart: x = writebyte(&HA6): x = writebyte(INT(paddr/256))
x = writebyte(paddr MOD 256)
FOR i = 1 TO 64
     x = writebyte(pdat(paddr))
     paddr = paddr + 1
NEXT i
x = cndstop

x = 1
WHILE x = 1
     x = cndstart
```

(continued)

SOFTWARE

There are three levels to the software. The low-level code deals with all of the interfacing to the PC parallel-port registers and implements the bit-level communication. At the middle, there are routines which transfer one byte of data and handle the ACK bit. At the highest level, the byte-level routines send the commands and read and write the data necessary to implement reading and programming the chip.

The high-level code first initializes the memory, reads all of the chip into memory, reads the hex file into memory, programs the chip from memory, and finally verifies the content of the device against memory.

READING AND WRITING

Reading a block of memory is fairly trivial, as Listing 1 shows. First, we set the address by doing a write (A6), but not transferring any data. Then we do a read (A7) and start sequentially reading data until we have enough (4096, in this case).

Similarly, programming of the device (given in Listing 2) is accomplished by sending a write command (A 6) followed by the address and 64 bytes of data. The stop condition starts the internal write operation. While the device writes the page, it does not respond to any other commands.

The routines readbyte() and writebyte() shown in Listing3 transfer one byte of data and handle the ACK bit. Readbyte() transmits the ACK bit (which is zero), where the writebyte() routine just reads the ninth bit as returned by the slave. This means writebyte() returns 0 if successful and 1 otherwise. When programming the chip, the return value of writebyte tests whether the last write cycle is still in progress.

Listing 4 gives the low-level routines cndstart() and cndstop(), which signal the start and stop condition needed for 2-wire operation by making the data-line transition while the clock line is driven high.

Writebit() sendsabittotheslave by asserting the value on the data line and sending a clock transition low-high-low. Readbit() does the inverse by sensing the data line during the high part of the low-high-low clock transition as you see in Listing 5.

WHERE TO GO WITH THIS

Since Atmel FPGA-configuration EEPROMs can be switched in 2-wire mode at any time and 2-wire mode supports random access reads, the unused space in the configuration PROM can store parameters used by the FPGA during operations. The FPGA may even use 2-wire protocol to store information or to rewrite its configuration bit-stream, allowing in-circuit programming without a CPU.

Very complex state machines are implemented by storing the state tables in the unused space in the EEPROM. One variation on this is to design CPU architectures in the FPGA and use the EEPROM as program memory. This alternative would make for some very compact designs.

Also, wave-shape tables can be stored in EEPROM for signal synthesis. FPGAs are finding their way into DSP applications because of their flex-

Listing *2-continued*

```
    x = writebyte(&HA7)
    IF x = 1 THEN PRINT ".";
WEND
x = cndstop
```

Listing 3—*Using the low level readbit and writebit routines in Listing 5, readbyte and writebyte transfer one byte of data and hand/e the ack bit.*

```
FUNCTION readbyte
    x = 0
    FOR i = 1 TO 8
        b = readbit
        x = x * 2 + b
    NEXT i
    b = writebit(0)
    readbyte = x
END FUNCTION
FUNCTION writebyte (b)
    FOR i = 1 TO 8
        a = INT(b/(2 ^ (8 i)))
        a = a MOD 2
        x = writebit(a)
    NEXT i
    a = readbit
    writebyte = a
END FUNCTION
```

Listing 4-Routines *cndstart and cndstop handle sending the special clock and data sequence to signal a start and stop condition necessary for the 2-wire interface.*

```
FUNCTION cndstart
    OUT &H378,&HC3: OUT &H378,&HC3
    OUT &H378,&HC2: OUT &H378,&HC2
    OUT &H378,&HC0
    OUT &H378,&HC1: OUT &H378,&HC1
END FUNCTION

FUNCTION cndstop
    OUT &H378,&HC0: OUT &H378,&HC0
    OUT &H378,&HC2: OUT &H378,&HC2
    OUT &H378,&HC3: OUT &H378,&HC3
    OUT &H378,&HC1: OUT &H378,&HC1
END FUNCTION
```

Listing 5—*Readbit and writebit use the clock and data lines to transfer one bit of data.*

```
FUNCTION readbit
    OUT &H378, &HC3
    OUT &H378, &HC3
    a = INT(INP(&H379) / 16)
    a = a MOD 2
    OUT &H378, &HC1
    OUT &H378, &HC1
    readbit = a
END FUNCTION
FUNCTION writebit (bit)
    OUT &H378, &HC0 + bit
    OUT &H378, &HC0 + bit
    OUT &H378, &HC2 + bit
    OUT &H378, &HC2 + bit
    OUT &H378, &HC0 + bit
    OUT &H378, &HC0 + bit
    OUT &H378, &HC1
    OUT &H378, &HC1
END FUNCTION
```

ibility and speed. One application for wave-shape tables would be custom PWM control for electric motors.

Ingo ***Cyliax has a B.S. in Computer and Electrical Engineering from Purdue University. He now works in computer science at Indiana University where he does system administration and spends a lot of time in the analog VLSI and robotics labs working on Stiquitos*** *(http://www.cs.indiana.edu/robotics/colony.html).* ***He is also a partner at*** *EZComm* ***Consulting. He may be reached at*** *cyliax@EZComm.com.*

REFERENCES

Atmel Corp., ***AT1*** *7C65/128/256* ***FPGA Configuration EEPROM Datasheet, 1995.***

Atmel Corp., *Atmel* ***Configurable Logic Databook, 1994-1995.***

Atmel Corp., *Atmel* ***Data Book, 1993-1994.***

Atmel Corp., ***Programming Specification for*** *Atmel's* ***Configuration Memories,*** App-note, 1995.

Eggbrecht, L.C., ***Interfacing to the IBM Personal Computer,*** SAMS, Carmel, IN, 1990.

Motorola Semiconductor, ***Motorola Memory Data,*** 02/89 DL113 Rev. 5, Datasheet, 1989.

Xilinx, Inc., ***The Programmable Logic Data Book, 1994.***

SOURCES

XC3000/XC4000 FPGAs
Xilinx, Inc.
2100 Logic Dr.
San Jose, CA 95124-7114
(408) 559-7778
Fax: (408) 559-7114
http://www.xilinx.com/

ATT3000 FPGAs
AT&T Microelectronics Customer Response Center
Dept. AL-500404200
555 Union Blvd.
Allentown, PA 18103
(215) 439-4331
Fax: (215) 778-4106
http://www.attme.com/fpga/

AT17Cxxx serial EEPROM, AT6000 FPGAs
Atmel Corp.
2125 O'Nel Dr.
San Jose, CA 95 13 1
(408) 441-0311
Fax: (408) 436-4200
http://www.atmel.com/atmel/products/products3.html

SOFTWARE

To keep things simple, the programming software was written in Microsoft Quick Basic for DOS. A complete version of the program can be obtained at ftp://ftp.cs.indiana.edu/pub/goo/eeprg.zip.

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

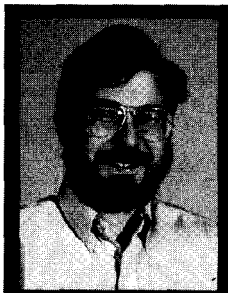
DEPARTMENTS

FIRMWARE FURNACE

Ed Nisley

80x86 Performance

I/O Bound



The next time you're starting the design of an embedded PC product, don't put a lot of weight on processor speed until you look at what it will spend its time doing. Ed explains why.

A friend called, asking whether I had any bicycle parts catalogs in my collection. His bike needed new chainrings, but he discovered, much to his surprise, that current components don't fit. We figured his bike might be eight years old, which is not at all a long time measured in bicycle generations.

Do you recall 198%vintage PCs! Hints: no PCI, no VLB, no EISA, no SVGA, no SCSI, no JPEG or MPEG, no CD-ROM. To a good first approximation, no GUI, no PnP, no EEPROM, no flash ROM. Basically, every acronym we take for granted today lay slightly beyond the horizon.

You can't buy a '386 desktop box today, but you can build a fine embedded system around a '386 CPU. While '286 boxes may be roadkill in the desktop market, clone chips live on in embedded PC systems. Even the 8088 lingers, driven more by software compatibility than raw go-power.

Only the PC architecture, whatever that may be, lets you choose performance over about two orders of magnitude on essentially the same hardware platform with essentially the same source code. If your application fits the PC model and cost requirements (and you can't afford lots of nonrecurring engineering charges), there aren't that many other choices.

After I laid out the last two columns on cache performance, I thought it would be interesting to run that Direct Digital Synthesis (DDS) code on a range of CPUs. At about the same time, I ran into the local Megatel rep at a trade show here in Raleigh. We kicked a few ideas around and, as they say, the rest is history.

This month I'll run the DDS loop code on three different Megatel embedded-PC boards and my desktop test system. We'll examine the effect of clock speed, CPU type, and I/O bandwidth on the DDS code's performance. Because the DDS loop spends most of its time in I/O operations, the results are quite surprising.

First, let's take a look at the machinery.

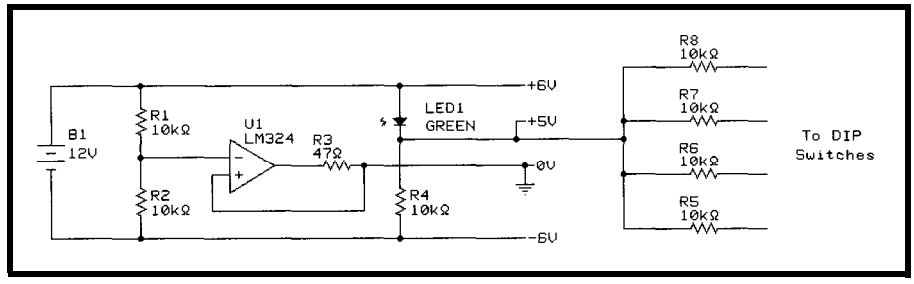


Figure 1—*Adding a low-current +5-V supply to the DDS-output* **circuitry requires nothing** more **than a green LED and** *a 10-k* **resistor.** *If* **you need more current or better regulation, spring for a** *3-terminal* **regulator.**

MEET THE BOARD(S)

The Megatel PC/+*i* board features a 25-MHz Intel '486SL CPU which, despite what "SL" might imply to you, has an on-chip math coprocessor. The system does not have an external memory cache between the CPU and the 4 MB (optionally 16) of DRAM. The BIOS resides in part of a 2-MB flash ROM that can be used as a file system.

The PC/+*i* crams essentially everything you could ask for into 16 square inches. In addition to the standard serial, parallel, and keyboard ports, you get a floppy controller, SVGA or LCD controller, Ethernet adapter, SCSI interface, and all the usual AT-style system-board peripherals. That's overkill for our purposes, but having a system that boots DOS directly from a floppy certainly simplified things.

Mmmm, that last sentence pretty well sums up the entire embedded PC biz....

The Megatel PC/II+ differs from the PC/II+*i* mostly in the CPU, a 25-MHz Intel '386SL. In this case, "SL" means what you think-the CPU lacks a math coprocessor. You can get 2, 4, or 10 MB of DRAM along with 1 MB of flash memory and the same assortment of peripherals.

Further down the power curve, the Megatel PC/+*v* sports a 16-MHz NEC V40 CPU, which is basically a tuned-up '286 with several on-chip peripherals. The board includes 640 KB of DRAM, a VGA or LCD controller, floppy and SCSI controllers, and the standard serial, parallel, and keyboard ports.

The PC/II+ and PC/II+*i* boards I'm using carry Megatel QTB/104 breakout boards that convert a fearsome array of vertical pins into a PC/104 header and a set of I/O connectors. An optional 96-pin rectangular DIN connector gives you single-plug access to all those signals, although you probably don't want to run them all together through any length of cable.

Photo **1** shows the three CPU boards, along with their I/O expanders. I intended to pull the '386 and '486 boards apart so you could see the circuitry underneath the I/O boards. After one attempt, I performed a simple experiment with an 8-pin connector and a kitchen scale. Quiz: if it takes 2.5 lbs. to move an 8-pin strip, how much force will you exert on a board with 260 pins?

Extra credit: write an essay explaining why the boards no longer work after you crush them back together.

If you opt for the DIN connector, you can then use the QTB/II paddle card shown attached to the PC/+*v*. You'll still need Megatel's hydra cable to break out the serial and parallel ports from the card's minuscule 50-pin high-density connector.

Note to all manufacturers: if you expect users to plug ribbon cables onto vertical pin headers, unambiguously identify pin 1, right there on the board, in big letters! The QTB/II has four unmarked headers, one of which is backwards from the others. I got it right, but only because I'm a compulsive manual reader. The QTB/104 boards do identify pin 1, much to my relief.

The video hardware drives either a VGA-class display or a bitmapped LCD panel. If you recall our adventures with big LCD panels a few years ago, you won't be surprised at this sentence from the Megatel manual: "Since there are no standard signal names for LCDs, the following table lists some of the compatible signal names used by various panel manufacturers."

Tempting though it was to rummage through my LCD panel collection, I simply mooched the VGA display from my desktop testbed PC, along with its keyboard, power supply, and floppy drive. All the boards fired up, once I remembered that the PC/+*v* requires an XT-compatible keyboard and flipped the appropriate switch inside my keyboard.

Getting the DDS code running, however, required a few tweaks and twiddles.

COMPATIBILITY POTHOLES

Despite the relentless standardization of desktop-PC hardware, there remain many undocumented, unexplored features that may or may not work the same way on all systems. My experience has been that the Big Picture algorithm works fine, while the grubby details get wedged. When you move code, even from one 'x86 CPU to another, pay careful attention to trivial hardware interfaces and expect to spend more than a little time with a scope, debugger, and the board documentation.

For example, when I first ran the code on the '386SX board, two of the four DIP switches didn't work. I probed around and found that, as with

many laptops, the LSI blocks on the Megatel boards omit the "standard" pullups on some (not all!) printer-port control inputs, presumably to save power. Those pins normally connect to pullups inside a printer, so the design decision made perfect sense at the time.

Because the power for my DDS DAC board comes from a 12-V battery split into ±6-V supplies, I couldn't simply run pullups from the positive supply to the port pins. Not, that is, unless I wanted to risk smoking a rather expensive loaner. Refer back to the schematic in Figure 5 of *INK 69* for the complete power-supply circuit.

Everybody knows that a forward-biased diode drops about 0.5–0.7 V. Surprisingly, fewer folks recall that a green LED drops more than 1.5 V at 1 mA, which is just what's needed for 5 V. A 12-V lead-acid battery normally supplies about 13 V across the terminals, making the nominal 6 V weigh in closer to 6.5 V.

Thus, the green LED and resistor shown in Figure 1 provide about +5 V relative to the split common reference driven by the op-amp. The LED's bias current bypasses the common point, reducing the load on the op-amp. Check your LEDs to verify their forward drop, as it depends on the exact LED color and bias current.

Sure, go ahead, use a fancy low-dropout regulator instead of an LED. Just don't forget those decoupling capacitors!

When I fired up the 16-bit version of the DDS code on the PC/+V V40 board, the analog output didn't change at all. I dug out Debug to see how the hardware differed from my expectations and found that the BIOS didn't recognize the printer port at all. The LPT1 address at 0040:0008 was zero, not 378 as the documentation claimed.

It turned out that the hydra cable breaking out the serial and parallel ports was miscrimped, shorting several adjacent conductor pairs. The BIOS ran its usual power-on tests, found that the port didn't respond correctly, and did not insert the port address into the table. That's the way the BIOS should work!

After installing a new cable and verifying that the port worked with manual outputs from Debug, I fired up DDS16 again. This time the port output stuck at -6 V.

The manual clearly states that setting bit 5 in the printer-control port circuitry disables the data-port drivers. That comes as no surprise, being a fairly standard way to implement a bidirectional parallel port. The trick dates back to the original IBM PC's parallel-port hardware.

Unfortunately, the documentation *doesn't* **say** that the LSI blocks on this board always return bit 5 high, no matter what value you write. The first time my DDS code changed a bit in the control port, it read bit 5 as a 1, flipped the other bit, and wrote the whole byte back. That immediately disabled the data-port drivers. The DAC inputs pulled the port bits low, forcing the analog output to -6 V.

Blap!

After tracking those glitches down, the DDS code snapped into focus. Running good old DOS Debug from a diskette simplified matters, although for a more complex program, deploying a more potent debugger might be reassuring. In any event, I didn't spend a lot of time working on the truly obscure hardware and timing bugs that come with an entirely new design.

That's another sentence summing up the embedded PC biz... .

BEHIND THE NUMBERS

Measuring system performance raises some exceedingly tricky issues. When your project requires the resources of an 80x86-class CPU, your choice of algorithm and compiler generally have more influence on the outcome than the CPU's low-level, bit-twiddling ability.

Typical microcontrollers execute their classic input-compute-output cycle without much computing. Given their (relatively) scarce memory, there's a stiff upper limit on how complex a problem you can stuff into their program space. For example, figuring inverse kinematics for a three-axis robotic arm joint using an 8051 just isn't in the cards, unless your arm is in no particular hurry for the results.

Conversely, assigning a '486 to the same task makes at least some sense. You get a decent CPU that can run a high-level language, a math coprocessor for fast(er) floating-point computations, and at least on the Megatel boards, a built-in Ethernet interface that can link your 'bot together without serial port hassles. Given the right software, you can even slap the CPU into protected mode and use all that memory without 64-KB segment contortions.

Bearing all that in mind, the numbers from my simple DDS loop measure the boards' lowest-level I/O capabilities and instruction timings rather than their flat-out, compute-bound limits. Your programs probably depend less on I/O performance than

Listing 1—*The comments indicate the clock cycles required for* **each instruction on '386 and'486** *CPUs.* **The '486 CPU imposes a** *1-cycle penalty for 32-M operands! The values reflect the idea/ rate, neglecting prefetch delays, cache misses, wait states, and so forth.*

```
AND  AL,NOT FLAG-MEMORY   ; signal table lookup
OUT  DX,AL

MOV  ESI,EBX              ;22 get high word of phase
SHR  ESI,16               ; 3 3 . . . into SI for addressing
AND  SI,0FFFCh            ; ?? . . make dword offset
MOV  EAX,[ES:SI+OFFSET SineTable]  4 2 get table entry
SHR  EAX,24               ; 3 4 move MSB to AL
MOV  DX,[DataPort]        ; 4 1 send out the new value
OUT  DX,AL                ; 14 16

MOV  DX,[ControlPort]     ; 4 1 fetch control port address
IN   AL,DX                ; 13 14
OR   AL,FLAG_MEMORY       ;21 signal table lookup
OUT  DX,AL                ; 14 16
                          ; 65 52 Total cycles
```

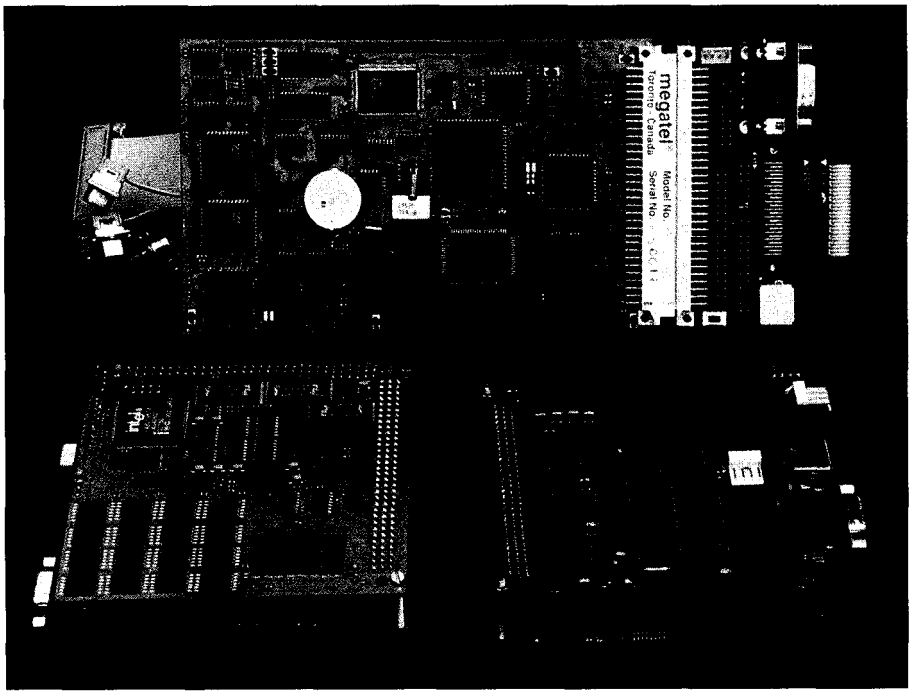


Photo 1—*These* **three** *Megatel* **embedded-PC boards sport** *I/O* **expanders for standard peripherals.** *The PC/II+v has a* **QTB/II paddle card on ifs** *96-pin DIN connecter, while the PC///t and PC/II+i* **boards** *carry QTB/104 breakout* **boards, You** *can choose from a* **bewildering array** *of* **options** *that* **adapt** *the* **boards to** *your system.*

the DDS loop, but this code serves as an important limiting case.

We'll tilt the scales the other way next month with a test program that's entirely CPU-bound. That foray will establish another performance limit that your code probably won't reach either. Locating the boundaries, though, ought to give you a good idea of the range available in the hardware.

But, for now, you've probably noticed a trend, at least in the ads, that calls for the biggest, fastest, most costly embedded PC, perhaps regardless of the real requirements. After all, if you need a PC, why not throw a big one at the problem and be done with it?

Well, let's find out.

TRACING OUTPUTS

We covered the details of the DDS code last month, so this time around I'll spend more time on CPU comparisons. I modified the code slightly from last month's listings to focus on the I/O operations rather than the cache. You can download the complete source code from the BBS.

Listing **1** shows the table lookup part of the 32-bit DDS loop, bracketed by **OUT** instructions that generate a low-going pulse on a parallel port bit. Each comment includes the number of CPU cycles required on '386 and '486 CPUs, as specified in my collection of manuals and books.

If you just glanced quickly at Listing **1,** glance again. Although the three I/O instructions represent only 27% of the code, a '486 CPU spends nearly 88% of its time executing them!

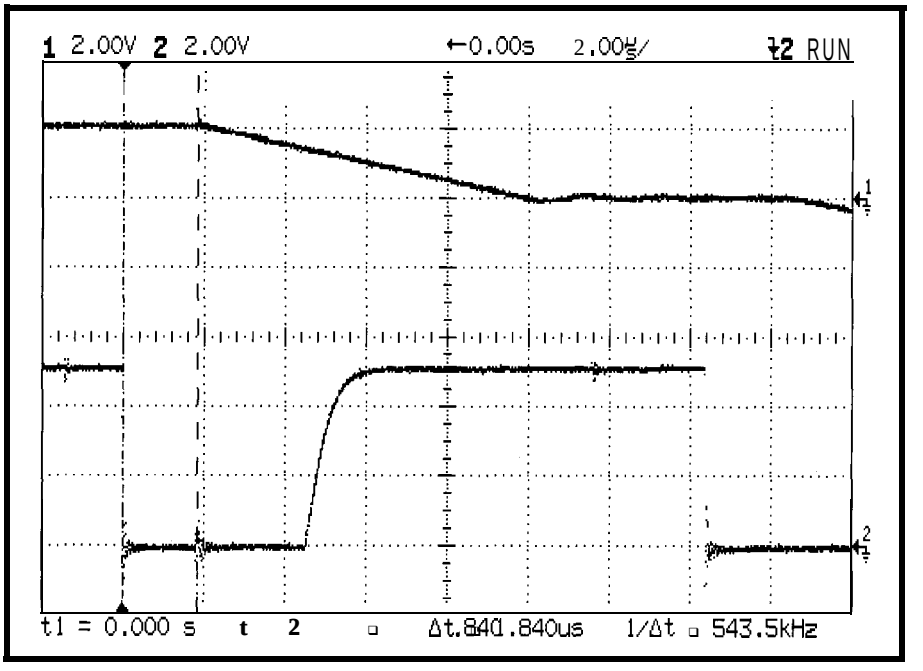
It's tempting to invoke the Heisenberg Uncertainty Principle here, even if it doesn't apply directly. The simple act of marking the memory lookup with I/O instructions decreases the performance to perhaps 30% of normal. If we didn't watch the code quite so closely, it would definitely run faster.

Quiz: if you don't care at all how fast your code runs, does it matter if it runs at all?

The '486 manual points out that its lists of instruction-execution times exclude real-world effects like instruction prefetching delays, memory cache misses, I/O wait states, and so forth. Under those ideal assumptions, the table lookup code should run in 52 clock cycles or 2.1 µs at 25 MHz.

Photo 2 shows how the real world, as implemented in the Megatel PC/II+*i*, differs from the handbook's opti-

Photo *2-The 25-MHz'486SL CPU on the Megatel* **PCIIIii** *board executes the* **1** *l-instruction sine-tab/e lookup code in about 4.5 µs. The analog output occurs about 40% of the way through the sync pulse because arithmetic and memory instructions run much faster than I/O instructions.*



mism. The op-amp produces the slew-rate-limited ramp in the top trace after the DAC output changes at the right-hand cursor. The pulse in the bottom trace lasts about 4.5 µs, more than twice what you'd expect.

Gotcha!

The culprit here, it turns out, is the familiar accretion of Compatibility Barnacles on any ISA design. Because the PC/II+*i* includes a PC/104 bus interface, the I/O timings date back to the original AT. Each I/O operation comes encrusted with enough wait states to reduce the bus bandwidth to about one access every microsecond.

The scope cursors mark the seven instructions executing between the falling edge in Trace 2 and the start of the ramp in Trace 1. The remainder of the pulse-more than half of it—includes just four instructions. Hmmm?

If your application does lots of thinking between I/O operations, a fast CPU speeds things up dramatically. On the other hand, if you are just twiddling a few bits, the I/O bus performance dominates everything else.

A scope shot like that can be a real eye-opener, particularly for suits who equate fast CPU clocks with blazing I/O performance.

MAKING WAVES

With that in mind, I ran the revised DDS loop on the three Megatel boards and my 80-MHz'486DX2 desktop test system. Photo 3 shows the code on the Megatel PC/II+*i* board, producing a rather tatty-looking, 16-point sine wave at about 4.3 kHz.

I also measured the effect of various memory-caching combinations. The Megatel BIOS does not include a cache-control function, an entirely reasonable omission for an embedded-PC board that will run a single program forever. Listing 2 shows the few instructions in **D D** S32CD.**E X E** that disable and flush the internal cache on '486 CPUs.

Table 1 summarizes the results. Using the 16-point DDS sine-wave frequency as a measure of execution speed, the four CPUs differed by a factor of 3.9. That's rather less than you would expect for a 16-MHz '286 versus an 80-MHz'486DX2, but it reflects the I/O-dominated nature of the DDS code.

The "Minimum output pulse width" row shows the duration of a three-instruction output pulse. The ISA Compatibility Barnacles set this value, so boosting the clock speed and hitting the cache just won't help.

The "Memory access pulse width" row reports the width of the pulse marking the memory access, as shown in Trace 2 of Photo 2. Given a few more instructions to work on, memory

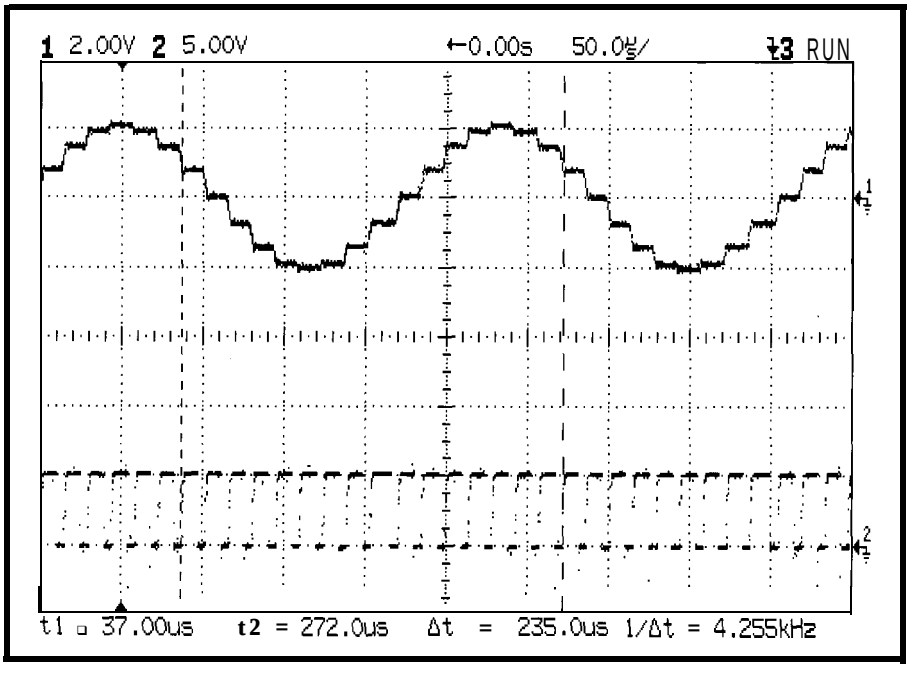


Photo 3-A *Megatel PC/II+i board with a 25-MHz '486SL processor produces a lb-point, 4.3-kHz sine wave. The lower trace marks the memory accesses preceding each output step and is somewhat spotty at this relatively stow sweep rate.*

caching provides a boost reaching 21% for the DX2 processor. Because the entire DDS loop and all the lookup table data fit neatly into the cache for this test, the external cache has no measurable effect.

Knowing both the number of instructions and their execution time, it's easy to calculate the execution speed for both the short stretch of memory access code and the entire DDS loop. You can see how critical memory performance is for a DX2 processor. Even for this I/O-bound code, caching increases the speed by almost 50%.

The last line of the table gives the average number of clock cycles per instruction in this code. A casual reading of the '486 manual might convince you that each instruction requires only a few cycles. Diluting that CPU-bound optimism with real wait states means each instruction requires about ten clock cycles or 400 µs.

Now, the fallacy of throwing a faster CPU at a simple problem becomes obvious. The DX2 in my test system has an 80-MHz internal clock, three times faster than the 25-MHz clocks on the PC/II+ and PC/II+i boards. Unfortunately, it also spends three times longer on each instruction in the DDS loop. The net gain? Essentially zero!

The extension to clock tripling and quadrupling CPUs should be obvious to the casual observer.

Moral of this month's story: if you're doing lots of I/O and not much computing, beware the standard benchmarks.

Listing 2—*Turning off the* '486 internal cache requires *setting* two *bits in Control Register 0.* **I used the Bit Test and Set instruction to emphasize the bit** locations, **even though** *that* **code is** *slower than* **a** *simple* AND **with** *a bit mask. The last instruction flushes the infernal cache to ensure that the timings reflect main memory accesses.*

```
IF      1 EQ KILL-CACHE
MOV     EAX,CR0
BTS     EAX.30          ; disable cache updates
BTS     EAX,29          ; disable writethrough
MOV     CR0,EAX
WBINVD                  ; invalidate all cache entries
ENDIF
```

RELEASE NOTES

The files this month include **DDS32.EXE** for '386 and '486 systems, DDS32CD.EXE to disable the '486 internal cache and **DDS16.EXE** for the Megatel V40 board. They produce an analog sine-wave output through a DAC on the printer data port and timing pulses on the printer control port. You'll need a scope to view the results.

These programs disable all interrupts and (attempt to) shut off RAM refresh. You must reset the system to regain control, so don't run them on

CPU	V40	'386SL	486SL	486SL	'486DX2	'486DX2	'486DX2
Memory caching	none	none	none	internal	none	internal	int + ext
Clock MHz, ext/int	16	25	25	25	40/80	40/80	40/80
DDS frequency kHz, 4 points	4.76	13.9	14.7	16.6	12.14	18.7	18.7
DDS frequency kHz, 16 points	1.22	3.56	3.80	4.26	3.14	4.61	4.79
Minimum output pulse width µs	1.54	1.34	1.34	1.36	1.38	1.38	1.38
Memory access pulse width µs	13.0	5.04	4.82	4.50	5.20	4.08	4.08
MIPS during memory access	0.85	2.2	2.4	2.4	2.1	2.7	2.7
MIPS for complete DDS loop	0.90	2.2	2.4	2.7	2.0	2.9	3.0
Clock cycles per instruction	19	11	10	10	38	30	30

Table 1—*These four* **different PC-compatible systems** *cover the range* **from an enhanced '286** *to* **a '486** *desktop box.* **Dividing** *the* **clock frequency by** *the MIPS* **rating gives clock** *cycles* **per** *instruction, a much-touted figure. The DDS loop,* **a** *severely I/O-bound* **program, spends considerable fime in** *bus* **waif states!**

your desktop box or, ahem, your network server!

If you wonder how and why the Intel **80x86 architecture** stomped the competition flat, see Nick Tredennick's paper in the December '95 Proceedings of the IEEE. His "Technology and Business: Forces Driving Microprocessor Evolution" bursts many of your illusions, no matter which side of the debate you start from. In short: volume drives economics, which drives development, which drives volume. Winner takes all!

Next month: we'll run some CPU-bound code on the same collection of hardware with some interesting results. ❑

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

SOURCE

PC/*+i*, PC/II, PC/II+*i*
Megatel Computer Corp.
125 Wendell Ave.
Weston, ON
Canada M9N3K9
(416) 2452953
Fax: (416) 245-6505

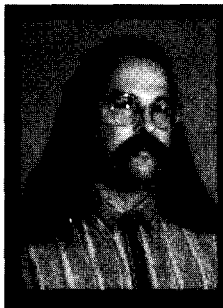
For datasheets, contact
Kurt Tugman
EMJ America
(800) 548-2319
Fax: (919) 460-8861

I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

Printer Sharing

Automatic Parallel Printer Switch



Stow your assembler, compiler, and programmer. This month's project doesn't need a micro. Jeff creates a smart printer switch that shares your favorite printing device between two computers.

FROM THE BENCH

Jeff Bachiochi

In the course of the day, I use two PCs. The PC on my desk is used for a number of simple tasks: word processing, schematic capture, and the daily jaunt to the onsite BBS.

Across from my desk are a couple of work benches. One of these is home to a CAD station which is mostly used for PCB layouts. I flit back and forth between the systems whenever a print routine takes longer than a trip down the hall to the Coke machine.

Everyday, I print reams of documentation in this paperless workplace. You know-there's nothing quite like a printout to get your mind going in the morning.

To facilitate the ability to print from both stations, I've installed a simple A/B switch box. I picked this up from a local computer fair for less than $10. Centrally located, it is only a matter of about five steps to get to the box. Of course, it is on the ground with a rat's nest of other cords which run to and from the PCs.

However, it seems as if the A/B is always in the wrong position, no matter where I'm sitting. If you start a print job with the switch incorrectly set, you lose the first character. Not such a big deal unless the character happens to be a special escape sequence and the printer starts spewing paper after every character in an effort to relieve itself.

I've been bit once too often this week. Like Ed says, "There's nothing like a new problem to take your mind off an old one."

SHARING

Smart printer-sharing devices have been around for a good many years. They started out as print buffers (when our printer had single-line buffers) and became smarter and smarter.

Special codes sent to the smart print-sharing device enable an attached PC to take temporary control of the printer. You could run into trouble if print files happen to have embedded data which matched control codes. This feature became more of a problem

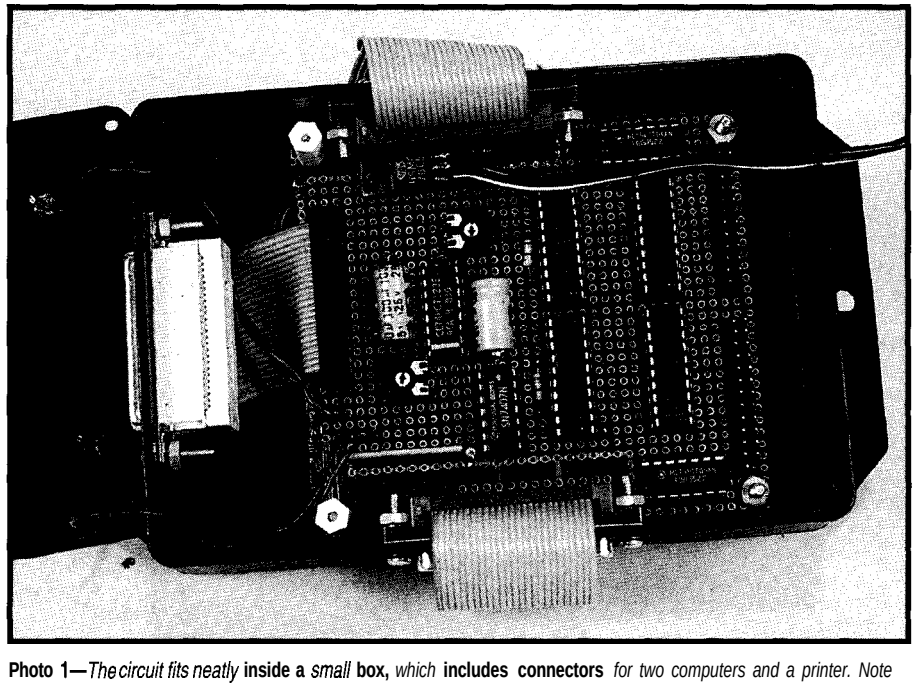


Photo 1—*The circuit fits neatly* **inside a** *small* **box,** *which* **includes connectors** *for two computers and a printer. Note that the cab/es from the board to the connectors have been removed to show the circuit board.*

as printers became more sophisticated and used special codes for changing typestyles and embedding graphics.

Enter the dumBOX. This month's project needs no processor. It doesn't buffer any data. It merely acts like a traffic cop on a first-come, first-serve basis.

Let's take a look at the Centronic's parallel printer-port signals to understand just what is involved.

SIGNALING TO AND FRO

The PC's original parallel printer port was designed to comply with an already-accepted standard. This bit of shortsightedness has led us to one of the most controversial boo-boos in the PC's relatively short history: a unidirectional port.

But, it is not my objective today to discuss the pros and cons of this unidirectional port. Newer equipment is already supplied with "extended mode support" (SPP, EPP, or ECP), so this will be a moot point once all the older machines have been junked.

Besides the eight lines carrying data to the printer, there are four control lines going to the printer and five lines returning status from the printer.

Not all printers use all control and status lines. However, the day you choose to not support one is the day your new printer will require it. (I think I read that on the Murphy's Oil Soap bottle or somewhere.)

Table 1 shows the make-up of the parallel printer-port connections along with an explanation of each signal.

DUMBOX THEORY

The first computer to drop its ● STB line gains control of the printer. This falling edge begins a retriggerable one-shot which in turn enables its data and control paths while locking out the path to and from the other computer. Each computer continues to receive the appropriate error status from the printer even while disabled.

The active ● ACK and BUSY status lines and the INIT control line are disabled to the inactive printer port, preventing the second computer from walking over the active print job.

DB-25M pin #	Centronics pin #	Signal name	Direction	Description
1	1	*STB	to printer	0.5-µs minimum logic low pulse indicating "good" data
2-9	2-9	DATA O-7	to printer	8 data bits, LSB to MSB
10	10	*ACK	from printer	0.5-µs minimum logic low pulse indicating data received
11	11	BUSY	from printer	Logic high signal indicating printer cannot receive data
12	12	PAPER OUT	from printer	Logic high signal indicating printer is out of paper
13	13	SELECT	from printer	Logic high signal indicating printer is ready to receive data
14	14	*AUTO FEED	to printer	Logic low signal to add a line feed after each carriage return
15	15	*ERROR	from printer	Logic low signal indicating problem with printer
16	16	INIT	to printer	Logic high signal to clear print buffer and initialize printer
18-25	18-25	GROUND	—	Logic ground

Table 1—*The Centronics signalnames and functions usedbetween a PC's parallel port andprinterare fairly standard.*

In addition, the inactive port's busy status is held busy while the active printer routine is tying up the printer. While the busy line is high, the inactive port does not attempt to begin a print cycle. Let's look at the handshake timing.

Figure 1 shows how the typical • STB signal can be delayed if the printer is busy. Most printers return both an *ACK strobe (recognizing the data byte has been received) and a BUSY status (indicating that the printer is midoperation).

Although there is some minimum pulse-width timing of the strobes, there seem to be some relationship differences between ● ACK and BUSY from printer to printer. Essentially, the status must be returned by the printer before each successive data byte can be transferred.

Referring to Figure 2, you see how the inactive port's busy status is held busy by the active port's one-shot. In addition, notice that the inactive port's INIT line is also prevented from interrupting a print job by the same retriggerable one-shot. If the inactive port was reset, it would generate an INIT. An existing print job would be destroyed.

Each printer port strobe starts its own retriggerable one-shot. The values of R and C used on the '123 offer a maximum pulse width of 45 s. Most printers can process characters with considerably less time between strobes.

The purpose of this time constant is to assure that if the printer is tied up in a particularly long operation, the inactive port is not given the opportunity to interrupt until the entire operation has ceased.

You may want to set these times independently for each computer. The LEDs are an indication of the retriggerable one-shot on time. Increase the time until the LED remains on for the entire print operation.

Two 74HC244s buffer the eight data lines coming from each of the two printer ports. Since these devices can be tristated, their outputs can be directly tied to together.

The one-shots prevent more than one buffer from being enabled at the same time. Two more '244s buffer the status lines. The ● STB, AUTO, INIT, and ● SEL control lines are buffered to the printer and the *ACK and BUSY status from the printer.

Since the INIT control to the printer is not synchronous with character transmissions, its activity must not depend on the data or control buffers being

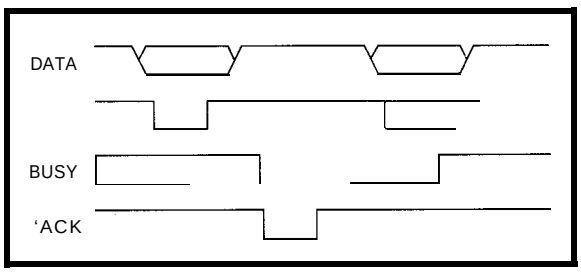


Figure 1-Observe *the basic relationship between the data and handshaking lines of the Cenfronics interface.*

enabled. Instead, it is grouped with the ● ACK and BUSY lines, so it's independent of the data and control buffers and connected directly to the printer (unless the other port disables it).

Three status lines from the printer are not buffered. The *ERR, P_OUT, and SEL status line are tied directly to both printer ports. This connection lets the actual printer status be monitored by both ports even while one port is printing.

When idle, many of the control and status lines are tristated. To prevent floating lines, pullup resistors create green signal conditions.

Notice that along with the BUSY and ● ACK status from printer port 1, its INIT control line is buffered in the same half of the second '244. The one-shot's Q output from port 2 disables this half of the '244, preventing interference from port **1** while printing.

At the same time, port 2's *Q output enables the data and control line buffers on port 2. The printer port requires a *BUSY status before it initiates a transfer. Therefore, the real busy condition must be reflected at each port while the printer is idle.

Many printers provide a 5-V output on the Centronic's connector. Rather than depend on this, I chose to add a 7805 regulator and a wall-wart transformer providing regulated 5-V power to the circuitry.

The PCB fits into a 4" x 6" small plastic enclosure (see Photo 1). The dumBOX's enclosure has mounting ears which enable it to easily mount to any vertical surface.

So, although I have one more widget needing AC power, I can mount it off the floor and tidy up the cables around my feet. Note that this design easily expands so more than two computer printer ports can be tied to a single printer.

Not all products which increase productivity require microprocessors. Although, I have to admit, my first thoughts were, "How can I use a micro to do this?" (I guess my thinking process has been altered by the advent of inexpensive micros. It is a difficult mentality to break free from.)

So, while we all ooh and ahh over products increasing in complexity, let's not forget simplicity has a charm all of its own. Low tech's alive and well in 1996 and living here under my bench. ❑

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

I R S

416 Very Useful
417 Moderately Useful
418 Not Useful

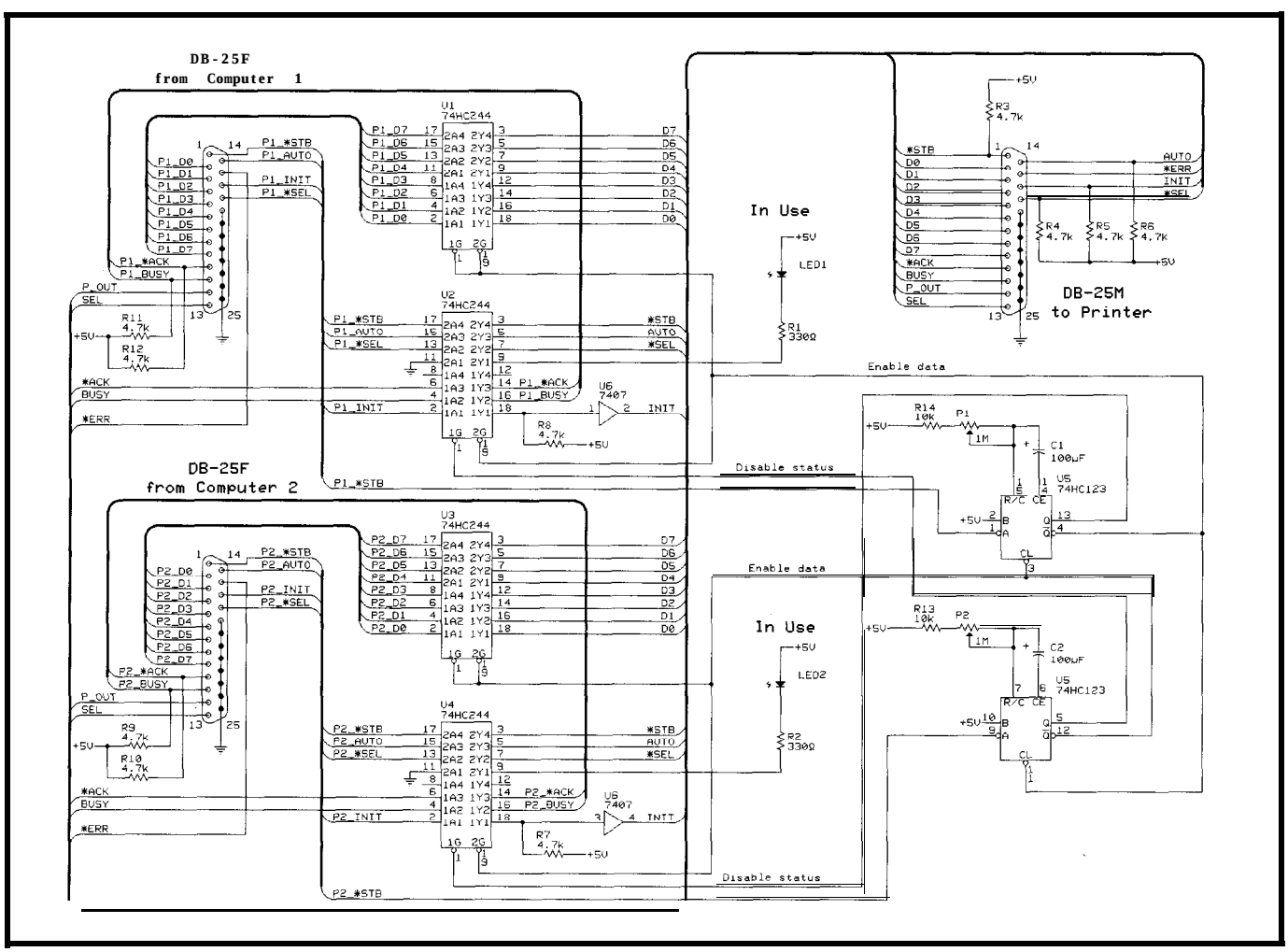


Figure 2—*This circuit supports the connection of two parallel ports to a single printer. Note the "in use" LEDs are actually driven from the Enable data lines of each one-shot.*

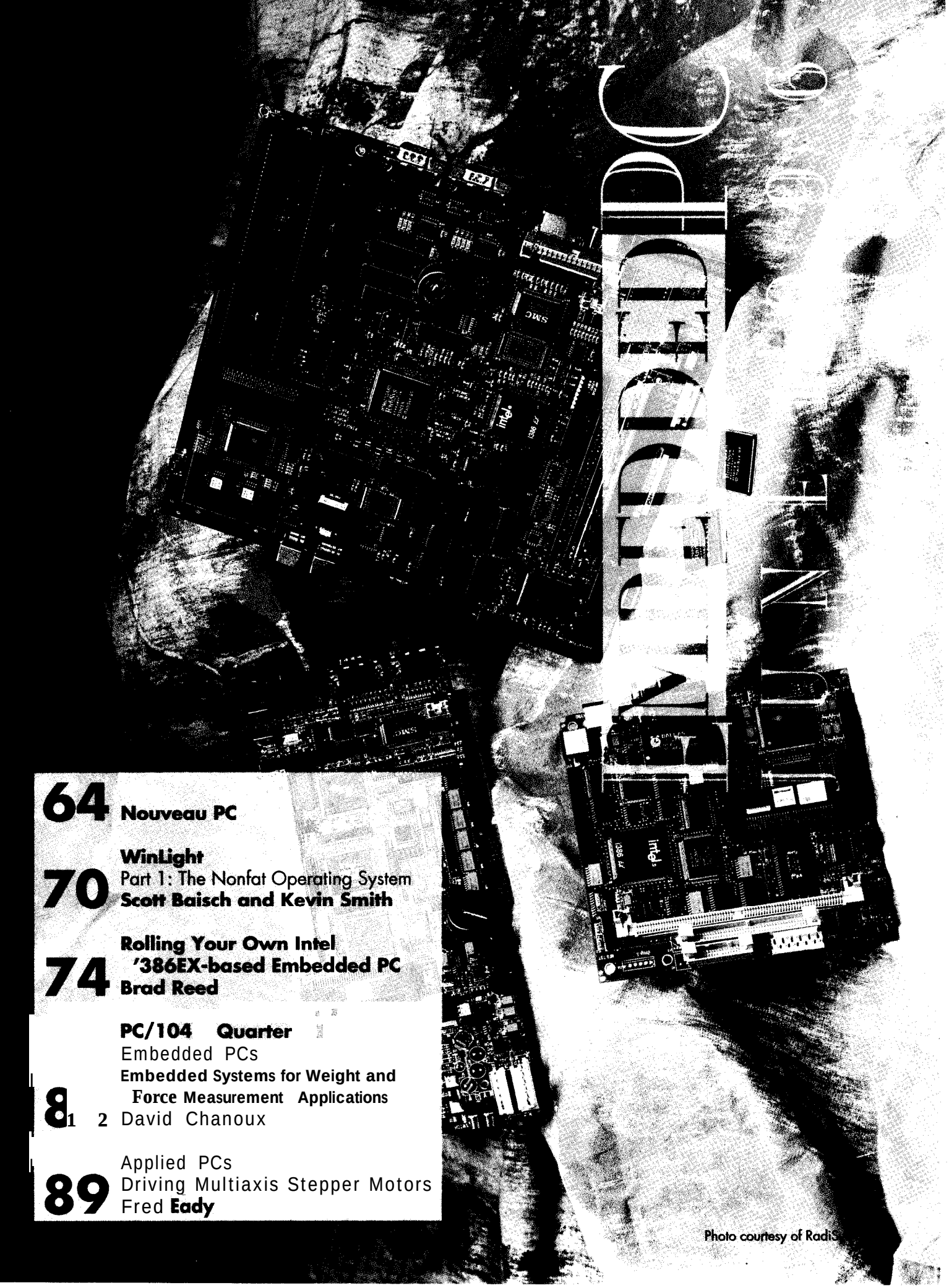
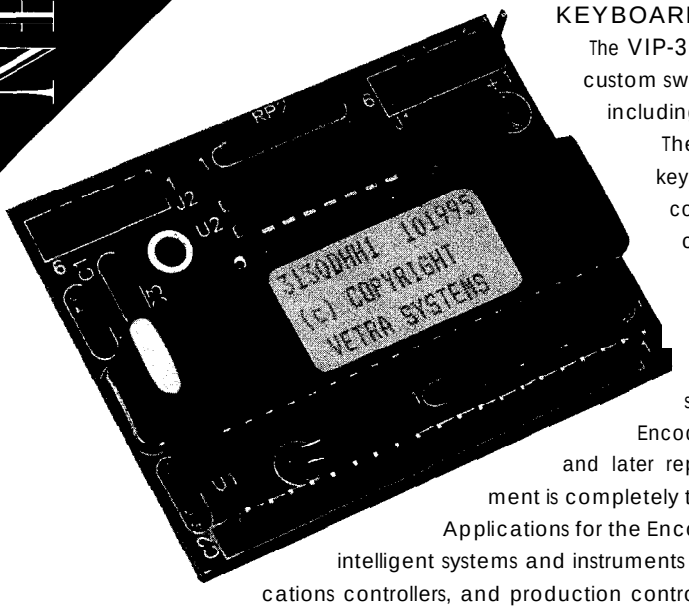


Photo courtesy of Radi

NPC



KEYBOARD ENCODER

The VIP-3 13 PC Keyboard Encoder accepts both a standard keyboard and custom switches, allowing simultaneous operation. Several models are offered, including units that directly interface with standard 3 x 4 and 4 x 4 keypads. The Encoders use a microprocessor to convert contact closures to PC keyboard scan codes that are compatible with the keyboard port of IBM-compatible PCs. Up to 16 contacts connected to common or in a matrix configuration can be used. The Encoders are packaged on 1.8" x 2.35" printed circuit boards that are powered from the PC's keyboard port. They require no external power supply. Custom key codes—such as Alt+key and Ctrl+key-can be supplied.

PC Keyboard Encoders permit system designers to use custom switches, panels, and keypads to replace standard PC keyboards. The Encoders enable a standard keyboard to be used for software development and later replaced with the custom-switch-plus-Encoder combination. This replacement is completely transparent to the application software.

Applications for the Encoders include all embedded and dedicated PC applications such as intelligent systems and instruments controlled by a PC, medical instruments, industrial control, communications controllers, and production control and test systems. The VIP-3 13 PC Keyboard Encoder sells for $49.

Vetra Systems Corp.
2755 Marcus Blvd. • Hauppauge, NY 11787 • (516) 434-3185 • Fax: (5 16) 434-3516 **#510**

INTERACTIVE ANSI C

Mosaic Industries introduces Control C, an interactive ANSI C compiler for the 68HC11-based QED-Board. Running under Windows 3.1 or Windows 95, the compiler makes it easy to program the QED Board using standard ANSI syntax.

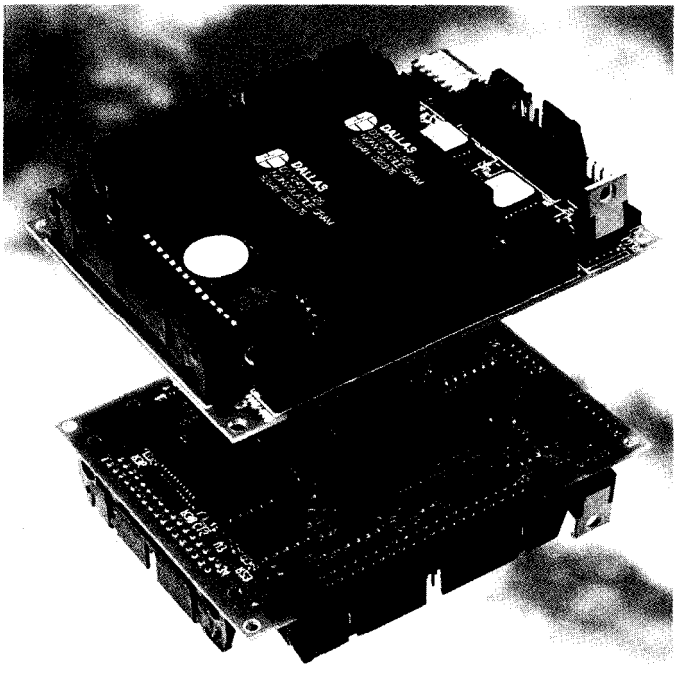
Its integrated development environment provides a high-performance editor and automated compilation. Precoded library routines include hundreds of functions that control the QED board's 8- and 12-bit A/D converters, D/A converter, digital I/O, keypad, scanner, graphicsdisplay interface, real-timeclock, and dual serial ports. A built-in multitasking executive supports fast cooperative and time-sliced task switching to simplify the design of real-time systems.

Precoded memory-management routines allow standard data types to be stored in arrays anywhere in the QED board's 8-MB address space. The operating system in onboard ROM provides system initialization and error handling, and makes it easy to set up an application to autostart on powerup.

A unique debugger facilitates interactive examination and modification of any variable or array element. The user can interactively call any function with any specified input arguments. This ability to thoroughly test each individual function of a program without recompiling the program greatly speeds debugging and testing.

Control C sells for $375. QED-Board quantity-one prices start at $495.

Mosaic Industries, Inc.
5437 Central Ave., Ste. 1 • Newark, CA 94560
(510) 790-I 255 • Fax: (510) 790-0925 **#511**



Nouveau PC

edited by Harv Weiner

EMBEDDED PC

The **2002C** single-board computer from TME combines a high level of integration with a 133-MHz AMD '486DX microprocessor. Included on the board is the Chips and Technologies 65535 CRT/Flat Panel controller with 1 MB of Video RAM. Resolution of up to 1280 x 1024 pixels can be supported.

An onboard 3-MB flash disk supports TME's proprietary QR Flash File system. Providing full load-leveling algorithms and a unique **TSR** which uses only 1 KB of main memory, the QR Flash File system requires no utilities. It operates entirely out of the BIOS and uses standard DOS formatting utilities.

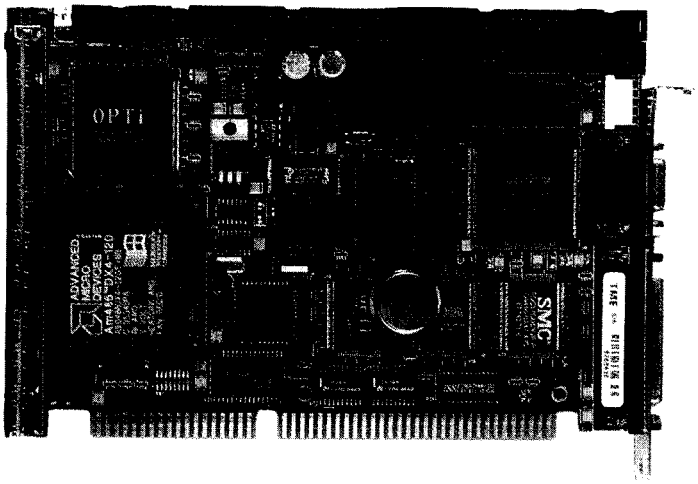
The 2002C provides two full 16C550 serial ports, printer port with ECP support, floppy and enhanced IDE support, and PC/I 04 and ISA interfaces as part of the standard I/O package. A watchdog timer, power-fail reset, and real-time clock are also provided. Memory capacity is up to 64 MB with 32-bit-wide RAM.

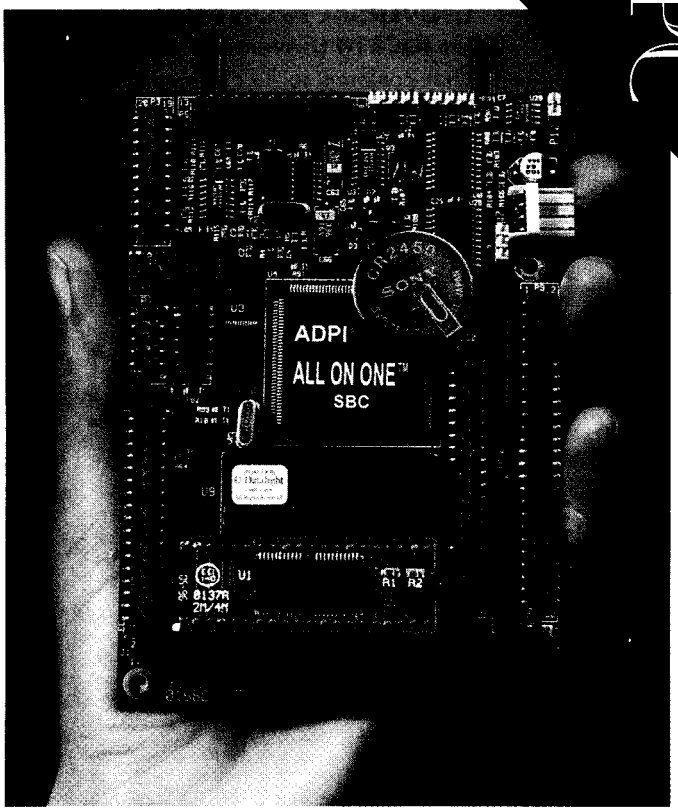
The 2002C also includes embedded PC and system BIOS features such as temperature sensing for processor clock-speed reduction, no fan required for speeds up to 100 MHz, optional fast boot, and boot without keyboard, among other things. The MTBF of the unit is not less than 150,000 hours.

A typical configuration for the 2002C single-board computer sells for less than $600.

Toronto **MicroElectronics,** Inc.
5149 Bradco Blvd.
Mississauga, ON
Canada **L4W 2A6**
(905) **625-3203**
Fax: (905) 625-3717

#512





SINGLE-BOARD COMPUTER

ADPI announces the "Little Guy" All on One computer. The All on One can be used for embedded applications or for smart peripheral control. It is designed around a 14-MHz '386SX microprocessor and measures 3.5" x 5".

The board requires 5 V at 150 mA and includes battery-backed memory and a power-save function. It features two onboard PCMCIA slots (hot swappable) and one PC/I 04 connector. Two serial ports providedata rates up to 1 15 kbps.

A bidirectional parallel port, SCSI port for up to seven devices, two-drive floppy-disk controller, keyboard, LCD and CRT interfaces, and speaker connector are also included on the board. All-on-One functions on any PC as a smart peripheral driver using a combination of ROM-DOS and ADPI's Embedded Peripheral Control Language.

A choice of storage media, I/O, and packaging options can be used for specific requirements. The board can also be used for data acquisition, program loading, and data transfer. All-on-One single-board computer prices start at $595.

Analog & Digital
Peripherals, Inc.
P.O. Box 499
Troy, OH 45373
(5 13) 339-224 1
Fax: (5 13) 339-0070

#513

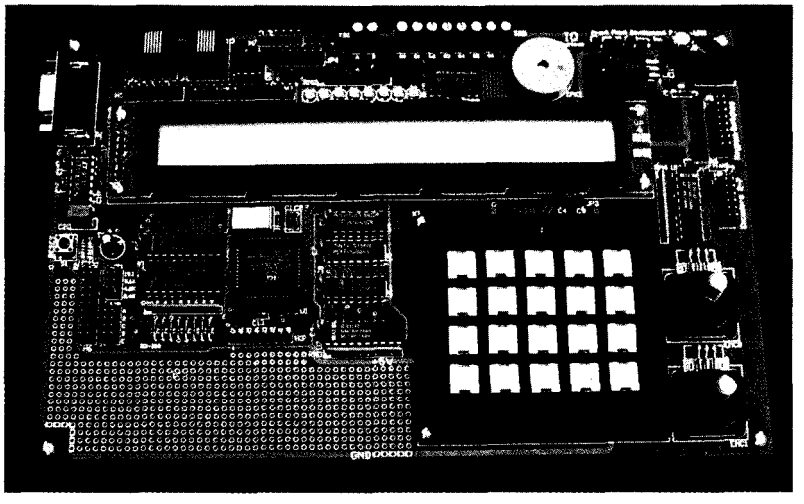
Nouveau PC

NPC

UNIVERSAL FRONT-PANEL CONTROLLER

The **IQC810** Universal Front-Panel Controller is a highdensity integrated circuit that conforms to IQ's Silicon Object architecture. Silicon Objects are smart peripherals that encapsulate real-time functions, are controlled via high-level ASCII messages, and require minimal host attention.

The IQC810 provides comprehensive support for all front-panel functions including a 32-character 7-segment LED display, dual LCDs, a 12 x 4 matrix keyboard, 48 switches, a programmable tone generator, and 8 quadrature rotary encoders. In addition, the unit provides a bus-write capability that allows engineers to easily interface to custom peripherals-discrete LEDs, DACs, displays, relays, and so on-while retaining a standard high-level software interface.



Although communications with Silicon Objects can be carried out over any hardwareconnection capable of conveying ASCII character strings, specialized ports are often unavailable on the host processor. ObjectLink data link protocol solves this problem by using just two general-purpose port pins for I/O and using state rotation rather than dedicated clock and data signals for information transfer. The result is a protocol optimized for soft implementation on a variety of microcontrollers that must contend with the demands of real-time event processing while simultaneously maintaining on-line peripheral communications.

Objectlink's low software overhead results in data rates of 30-50 kbps, even while using standard, economical 8-bit controllers such as the 8051 and 6805. An Objectlink Development/Analyses System consisting of an RS-232-to-Objectlink translator, timing and state analyzer, and LCD status display is available. The system can evaluate the performance of a direct ObjectLink connection when configured as a passive analyses and debugging tool.

Also available is an IQC8 10 Development/Evaluation System that comes with complete documentation, software, and application examples. The IQC8 10 Universal Front-Panel Controller **sells** for $695 in quantity. **The** ObjectLink Development/Analyses System costs $295 and the IQC8 10 Development/Evaluation System lists for $395.

IQ Systems, Inc.
20 Church Hill Rd. • **Newtown,** CT 06470 • (203) 270-8667 • Fax: (203) 270-9064

#514

PC/ 104 DATA-ACQUISITION CATALOG

ComputerBoards is offering its 48-page Data Acquisition ***and Control Catalog*** featuring analog-to-digital, digital I/O, analog output, counter/timer, electromechanical relay, and communications cards. Volume 11 introduces twelve new PC/104 cards for data acquisition.

Thisfreecatalog offers low-cost PC/I 04 cards with familiar, standard data-acquisition board configurations.

ADC cards with 100-kHz, 16-channel single-ended or 8-channel differential are available in 12- or 16-bit versions. Digital boards are available in I/O, dedicated input, or dedicated output configurations, supporting 48 digital lines for each card.

Counter/timer boards sup port 10 channels for event counting, frequency measurement, one shot, or frequency output. ComputerBoards designed its PC/I 04 output connectors to be pincompatible with standard 37-pin D-shell connectors to offer connection to the array of terminal, isolation, and multiplexer boards.

A universal library which supplies drivers for all its products and supports all popular programming languagesunder Windows or DOS is available.

ComputerBoards, Inc.
125 High St., Sk. 6
Mansfield, MA 02048
(508) 261-l 123
Fax: (508) **261-** 1094

#515

Nouveau PC

NPC

AM186/88ES EMBEDDED MICROCONTROLLER

The **Am1** 86ES microcontroller delivers 5.35 VAX MIPS-'386-level system performance-by using 70-ns commodity memory for low system costs over a wide range of embedded applications. Both ES-series controllers integrate virtually all of the peripherals required for a complete system.

The new microcontrollers offer several application-specific features including two asynchronous serial ports for versatility in design applications, an enhanced DMA controller for serial-port data transfers, and a pulse-width demodulator for real-time tactical feedback.

Additional ES-specific features include an 8- or 16-bit memory option for software-driven upgrades, two additional external interrupts, and an asynchronous interface that uses 68k-style peripherals and standard x86 peripherals. The ES also features an expanded watchdog timer for improved system reliability.

Based on industry-standard x86 architecture, the highly integrated ES series is 80C186- and 80C 188-compatible and is well suited forapplicationssuch asdiskdrives, hand-held and desktop terminals, set-top controllers, fax machines, printers, copiers, and a variety of telecommunications applications.

The Am 186ES microcontroller is priced at $12.94 for the 20-MHz version in quantity and at $12.29 for the 20-MHz **188ES**.

Advanced Micro Devices, Inc.
P.O. 80x 3453 • Sunnyvale, CA 94088-3453 • (408) 749-5703
http://www.amd.com/ **#516**

NouveauPC

TECHNICAL REFERENCE FOR CARDBUS

Addison-Wesley announces the publication of ***CardBus System*** Architecture, by MindShare Inc., Don Anderson, and Tom Shanley. The latest entry in MindShare's PC System Architecture Series, the book provides hardware and software engineers with the necessary background to design and test systems and components that implement the Cardbus specification.

As leading authorities on PC-system architecture, the authors have produced a detailed and comprehensive treatment of the 32-bit PC-card hardware and software interfaces and their relationships to overall system design.

Using a building-block approach that makes it suitable for readers at a variety of levels of technical expertise, the book explains difficult hardware topics **clearly** and simply, with a wealth of step-by-step instructions, illustrations, and detailed examples. The text provides essential, time-saving information for anyone who designs or tests hardware or software involving 32-bit PC cards.

CardBus System Architecture is available in paperback at local bookstores for a suggested retail price of $29.95 (ISBN O-201 - 40997-6, 432 pages).

Addison-Wesley Publishing **Co.**
One Jacob Way • Reading, **MA 01867**
(6 17) 944-3700
Fax: (617) 944-8243
http://www.aw.com/

#517

NouveauPC

EPC

Scott Baisch
Kevin Smith

WinLight

Part 1: The Nonfat Operating System

For many embedded developers, Windows has *brought more problems than solutions. WinLight, however, targets the embedded market. It provides a leaner, meaner, fully GUI, protected-mode,* Windows-like operating system.

Imagine this! Your group's new project requires you todesign, develop, and manufacture a system that has low memory requirements, operates in protected mode, is compatible with networked PCs, has a small form factor, and uses a touch-screen GUI. Oh, and, by the way, you need to get it done in six months!

Not realistic, right? Until this year, development of a system meeting these requirements probablywasn'tviable, at least not within a reasonable time-to-market.

So, what's happened this year to make development of such a system not only possible, but relatively simple?

WinLight happened.

What is Wintight? Throughout this article, we attempt to answer that question in a way that gives some insight into the relationship of WinLight and the development of the embedded system.

WinLight is a Microsoft Windows 3.1 work-alike operating system designed specifically for embedded platforms. Developed by Datalight, it's a graphical, multitasking, protected-mode operating system that runs Windows programs without modification or the need to recompile.

Since WinLight is a subset of Windows, it provides only those features and functions needed in an embedded environment. WinLight is much smaller and, consequently, less expensive in several ways.

Win tight was born in response to needs in the embedded and industrial systems market for extensions to Datalight's ROM-DOS.

These requests were primarily for the support of protected-mode calls and a graphical user interface. Since PC-based chipsets for embedded and industrial systems continued to become faster and more powerful, the requests made sense.

In 1994, Intel introduced the Intel '386EX chip for embedded systems and Advance Micro Devices announced its Elan SC300 '386-based chipset for hand-held units. As '386 and '486 chipsets lost their appeal in the desktop environment (see Figure 1), they found a home in the high-growth embedded-systems market. Figure 2 depicts the significant growth of these chips.

In fact, Intel's Semiconductor Products Group, which services the embedded and industrial system markets, generated nearly $2.5 billion in gross revenue for the 1995 fiscal year! Clearly, the need for a more sophisticated embedded OS was real.

HOW **WINLIGHT** WAS BORN

At the close of Comdex in Fall 1994, Datalight's research and developmentteam made a commitment to develop a new operating system that took full advantage of '386 and '486 chips in embedded systems.

R&D was convinced that there was a way to develop a new embedded operat-

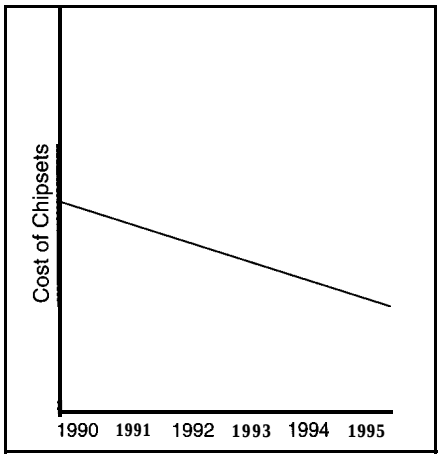


Figure 1: The costs of '386 and '486 *chipsets* has declined *dramatically* as the desktop market has moved to the Pentium.

ing system that didn't require the memory overhead of Windows, but could run an embedded Windows application. They launched the project, determined to write the kernel code in a couple of weeks.

Although the "couple of weeks" was too optimistic, Datalight developed such an operating system. This full-flavor, nonfat system deserves its moniker: Wintight.

In addition to being easy on hardware resources and shortening the time-to-market, WinLight solves several engineering problems typically associated with developing embedded systems.

BREAKING THE 640-KB BARRIER

The first and foremost problem solved by WinLight is the elimination of the infamous 640-KB conventional memory barrier.

Since WinLight operates in protected mode, applications can access up to 4 GB of memory. Software developers no longer need to spend valuable time squeezing their application's object code, drivers, and operating system into 640 KB of conventional memory.

Of course, memory use should always be optimized. How many engineers have spent hours trying to fit everything within the constraints of 640 KB? At such times, the addition of a small amount of memory could significantly shorten the development cycle without significantly increasing the hardware cost of the product.

A PROPRIETARY OS

In any project, hardware cost and size and development time are big issues. Until now, any product needing to take advantage of the speed, power, and price of the '386 and '486 chipsets had to include some sort of proprietary operating system.

Windows is not a practical solution for the embedded system, primarily due to its large size. Although this OS suits desktop PCs, it doesn't work in an embedded environment.

Development of a proprietary operating system that takes advantage of these powerful chipsets is daunting and significantly adds to the development cycle of any product relying on it. Using available proprietary systems ties you to one vendor, restricts the development tools available, and forces you to learn a new API.

WinLight eliminates the need to develop a proprietary operating system. It already uses the architecture of these chipsets and their ability to provide such features as a graphical user interface, protected-mode operation, and networked PC compatibility.

CHOOSING A DEVELOPMENT SYSTEM

Because WinLight is a Windows workalike, its applications can be written, compiled, tested, and debugged by Window developers using the existing Windows development environment.

Tools designed for the development of Windows applications are just as well suited to the development of WinLight applications. Consequently, the familiar desktop PC is the platform of choice for development of WinLight applications.

WHAT ABOUT MEMORY?

WinLight is modular in design, loading only the necessary Windows functions. That is, the basic WinLight operating system is small and scalable, supporting only those functions most commonly used by embedded systems.

Wintight's architecture, shown in Figure 3, keeps the operating system small and compact. This reduction in code size results in significantly smaller memory requirements.

OEMs should be able to configure WinLight to omit unnecessary functions. As WinLight grows, this scalability is key to keeping the minimum memory requirement low. As for memory devices, WinLight operates equally well from ROM, RAM, or flash memory, provided the application does not modify code segments.

WINLIGHT VERSUS WINDOWS

Windows is familiar to not only millions of software users, but also to many software developers. Developers typically want to know:

- since WinLight is so small, what's not supported?
- how compatible is it with Windows?

Wintight's small size is mainly due to the omission of features that are valuable to the desktop PC, but not to the embedded system. Table 1 shows the Windows functions supported by Wintight.

WinLight supports only a necessary subset of the Windows API. Windows often has multiple functions that return roughly the same result, whereas WinLight may only support one or two. It also excludes a program manager; applets such as the clock, calculator, control panel, write, and card file; and helper DLLs which have been added to recent versions of Windows.

Since WinLight was developed from scratch to run new applications, it isn't necessary that it provide compatibility **previously** developed desktop applications.

For Windows 3.1, it was necessary that it maintain backwards compatibility, so it could handle ill-behaved Windows 2.x and 3.0 applications. Needless to say, its code size increased significantly.

What about compatibility with Windows? Any application designed to run on WinLight also runs on Windows. However, only a few Windows applications run on WinLight because it implements only a subset of the Windows API.

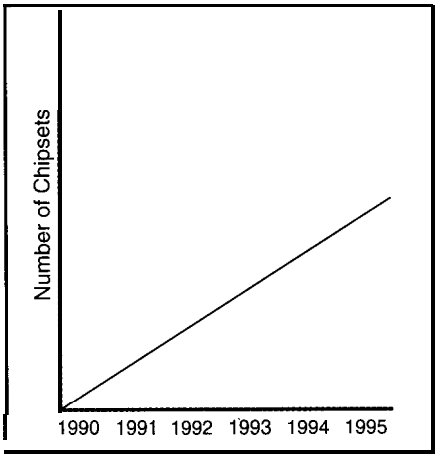


Figure 2: The growth of the '386 and '486 *chipsets* over the past few years has been significant.

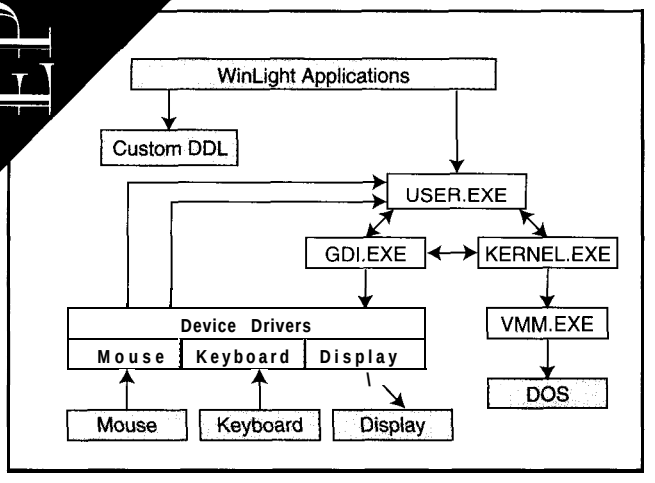


Figure 3: ***WinLight*** components (shown ***non***-shaded) interface ***with OEM-supplied components to compose the embedded system software. Shaded and nonshaded blocks show an architecture similar to Windows 3.1.***

SUITABLE APPLICATIONS

When writing an application for Windows, you're stuck with the entire operating system and all its features, regardless of whether your application needs them or can take advantage of them.

By providing different options for getting things done, Windows enables applications-and their users-to accomplish many tasks in more than one way.

This flexibility and robustness work well for the desktop application development and are acceptable for the embedded system—if the embedded system is taking advantage of the full Windows API. However, most embedded systems do quite well without the large overhead of the Windows API and are fully functional with a scaled-down version.

WinLight applications don't take advantage of the full range of Windows features, nor do they need to. Like the operating system they run on, these applications are typically smaller and more compact than those designed to run in Windows. However, they lend themselves well to the same desktop PC since the same Windows tools are **used** for development.

WinLight applications are written in C or C++ and implement only those functions and features supported by Wintight. The WinLight code is then compiled, tested, and debugged using familiar tools like Borland C/C++ or Microsoft C/C++.

Once the application executes under the Windows development platform, it is simply downloaded to the target system. There is no need to relink or compile since the new "Windows" application is actually a WinLight application-it is just developed as if it were a Windows application.

Using the desktop PC and proven C++ tools results in development-cycle time savings. This savings is due not only to the fact that the development hardware is typically present in the engineering department already, but also because Windows pro grammers are readily available.

A few years ago, programmers with Windows experience were not plentiful. But, the increased popularity of Windows applications in the marketplace and the demands it placed on the software industry has changed all that. Today, a talented Windows programmer is not hard to find.

Although similar, WinLight varies from Windows (see Table 1). In contrast, WinLight applications are totally suitable for the embedded system. These applications are small like the operating system they run on and fit easily within the constraints of embedded system hardware.

On the other hand, WinLight applications can be developed in a familiar environment and on a readily available Windows platform. Wintightalso provides all the functionality required to support a GUI, protected-mode operation, and compatibility with networked PCs.

Since WinLight and its companion, ROM-DOS, reside in a very small memory footprint (256 KB and 48 KB, respectively), the developer realizes some hardwarecost savings. Extra memory components are not needed to manufacture an embedded system that uses WinLight and ROM-DOS.

THE FUTURE

Whether WinLight can do for the embedded system what Windows has accomplished for the PC remains to be seen. However, it's quite clear that an intuitive user interface on any piece of equipment is far more desirable, and thus competitive, than a cryptic arrangement of buttons and a few lines of text.

No support	Provides support
DOS boxes	Event-driven structure
Dynamic Data Exchange (DDE)	Standard Get message
Object Linking and Embedding (OLE)	Mouse and keyboard input
Clipboard support	Window operations (create, move, size, maximize, close)
Multimedia (sound, midi, voice input, and animation)	Dialog boxes and child windows
Multiple Document Interface (MDI)	Standard controls, including static text, push buttons, option buttons, check boxes, edit boxes, scroll bars, list boxes, combo boxes, and group boxes
Printing	Using standard Windows resources, including cursors, icons, bitmaps, and strings
Pen input	Text output
VxD support ('386 virtual device drivers)	Menus and keyboard accelerators
Win32 API	Graphical commands, such as drawing points, lines, polygons, and ellipses
Registry	Full support for DLLs
TrueType	Standard GDI objects, such as DCs, pens, brushes, and regions
Color palettes	Global and local heap memory management
Real mode	File I/O
Metafiles	Timers
Drag and drop	Serial port support
WINHelp help viewer	Bitmap operations including support for ROP3 codes
	Profiles (.IN I files)
	Atoms and Window Properties

Table 1: To maintain its small size, WinLight doesn't provide support for all Windows functions. ***However,*** many features of windows are supported in ***WinLight's initial release.***

Whether the user interface is for car rental check-in, delivery drivers, or ticket-vending kiosks, Wintight's small size enables manufacturers to produce Windows systems affordably and with touch screens or hand-held dimensions.

WinLight enables a doctor to jot notes on a hand-held unit at each patient's bed after registering the patient's bar code information on the system, thereby eliminating the need for a huge administrative and medical records staff. Or, if all cars have a global-positioning system, available at the cost of a stereo, drivers could see their current location and destination.

The possibilities of creating WinLight applications are endless. Its lower hardware and software cost and standard development tools shorten time-to-market. Designers can create affordable applications that everyone can use.

In the next Embedded PC (INK 73), we'll examine the development cycle of a simple WinLight application. We'll provide some answers about:

- what a developer can and cannot do
- using Borland's Turbo and Remote Turbo Debugger for developing WinLight apps
- modifying an application to run on WinLight and how to choose API calls
- writing a Microsoft Foundation Class miniapplication for WinLight
- the effect of WinLight on flash or ROM and how it compares to the ROM version of WinLight EPC

Scott Baisch joined Datalight in early 1995. He has over IO years of sales and marketing experience in the software industry, *most recently with* Walker, *Richer,* and Quinn Inc. and *Microsoft Corp. You can* reach Scoff at *scottb@datalight.com.*

Kevin Smith has 18 *years of software development experience. He* joined *Datalight* in *1* 993 and is current/y Vice President of Engineering. *Kevin may be reached at kevins@datalight .com.*

SOURCES

WinLight
Datalight, Inc.
18810 59th Ave. NE
Arlington, WA 98223
(360) 435-8086
Fax: (360) 4350253

Intel '386EX **chipset** and ULP486 **chipset**
Intel Corp.
5000 W. Chandler Blvd.
Chandler, AZ 85226
(800) 628-8686
Fax: (800) 628-2283

AMD Elan SC300 microcontroller
Advanced Micro Devices, Inc.
P.O. Box 3453
Sunnyvale, CA 94088-3453
(408) 7495703

IRS

4 19 Very Useful
420 Moderately Useful
42 1 Not Useful

EPC

Brad Reed

Rollfng Your Own Intel '386EX-based Embedded PC

Brad introduces the R300EX and R380EX memory and **bus controllers created by RadiSys to support Intel's** *'386EX. A* **highly practical** *sidebar* **gives design advice for integrating the** *R300EX* **and** *R380EX* **with** *PC/104.*

Have you tried rolling your own low-cost embedded PC lately? It's not an easy task. Once you get past the BIOS customization issues that come up when you start adding and deleting standard PC features, you are still faced with a major hurdle.

What CPU and chipset are appropriate for the embedded PC and how long will they be available?

The tail end of the desktop-PC chipset market is just about ready to abandon the '486 and, in most cases, embedding a Pentium is overkill. The "integrated" '486 still often has availability and cost issues.

In addition, some highly integrated '486 chips only run in protected mode, while others provide features which may be unnecessary in the embedded system.

EMBEDDED PC SOLUTIONS

The Intel '386EX is the first processor chip to really address the embedded PC market. Not only does it start well down the road to PC compatibility with its integrated peripherals, but Intel has specifically addressed the embedded market with a commitment to long-term availability of the '386EX processor.

By integrating serial ports, interrupt controllers, timers, DMA, and more, the '386EX provides many of the functional blocks required to build an embedded PC. In addition, the processor includes watchdog-timer and power-management units which are often particularly useful in an embedded system.

Another factor increasing the '386EX's popularity is the **availability of** system BIOSs that specifically address the nuances of the '386EX. The availability of BIOS development kits from multiple vendors enables faster time-to-market with fewer resources than would otherwise be required.

Intel created their original Point Of Sale (POS) reference design as part of their Intel '386EX promotion. Incorporated into that design was a complex PLD (CPLD) that controlled DRAM, IDE, and flash-memory accesses and provided properly timed signals for accessing a keyboard, mouse controller, and real-time clock (RTC).

In addition, the CPLD generated the control signals required by peripheral chips like VGA and PCMCIA controllers. While the CPLD design provides the functionality required, it is expensive.

R300EX: RADISYS'S FIRST CHIP

Knowing the component end-of-life difficulties the embedded designer faces and being familiarwith the Intel '386EX, RadiSys created the R300EX, an enhanced ASIC version of the CPLD used in the POS design. The R300EX was specifically designed for the embedded market.

RadiSys has committed to providing long-term support for its R300EX and R380EX chips, consistent with Intel's commitment to provide the '386EX chips to the embedded market. RadiSys recognizes the need for long-lived components in areas beyond the Intel '386EX support, including VGA/LCD display controllers, PCMCIA controllers, and '486 chipsets.

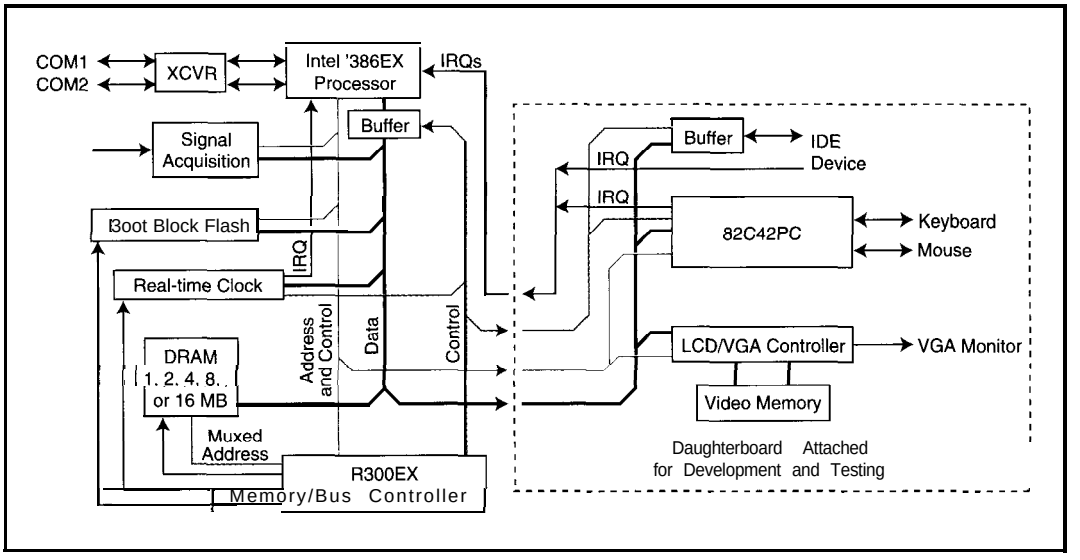


Figure 1: A simple data acquisition system using the Intel '386EX and the RadiSys R300EX: Through careful planning and design, a daughterboard completes the PC-compatible feature set and enables the same hardware to be used in development and testing. The daughtercard eases the burden in manufacturing tests as well.

By having a long-term source for processors and system components, designers can get on with the business of designing something new, without the end-of-life worries common in the embedded PC market.

Using Intel's processor in several designs, RadiSys faced the very same problem solved by the CPLD on the POS board. The drawback of the CPLD's cost, coupled with the opportunity to have a long-term supply of the memory and bus controller function, led to RadiSys' decision to create the R300EX.

The RadiSys R300EX memory and bus controller provides the minimum function set required to build a fully customized mix-and-match embedded PC solution.

R300EX FEATURES

The R300EX memory and bus controller supports EX CPUs up through the full 33-MHz Intel '386EX C-Step processor. It provides a pipelined, zero-waitstate, fast-page-mode DRAM controller that supports 1, 2, 4, 8, or 16 MB of DRAM. Integrated DRAM address multiplexers and CAS-before-RAS refresh circuitry complete the DRAM controller. Control signals are generated for flash and EPROM access, real-time clock (RTC) interface, IDE access, and keyboard and mouse controllers.

Implementing an ISA-like Synchronous Expansion Bus (SEB), the R300EX facilitates hookup of ISA peripherals like VGA or LCD and PCMCIA controller chips. The integrated dynamic bus sizing and READY generation, along with RESET-CLK synchronization and shutdown cycle-NMI generation, ease the embedded-PC designer's task.

The R300EX incorporates a NAND-tree pin configuration which simplifies ATE testing. It also handles some of the "limitations" of the signal timing from the CPU. It is lower cost and lower power, and it operates with better margins than a CPLD with equivalent functionality.

DESIGN TYPE AND THE R300EX

The R300EX's low cost means that it suits a wide range of systems with varying complexity. On the low end, it is appropriate for a system where only the DRAM-controller function is needed. At the high end, it can be used to build a PC-compatible embedded system like the EXPLR1.

A simple Intel '386EX design can use the R300EX as an easy-to-implement DRAM controller. A data logger, for instance, needing only CPU, DRAM, flash or EPROM, RTC, and serial I/O is built quite simply using the R300EX.

Choosing Between the R300EX and the R380EX

With the functional similarities of the R300EX and R380EX, how doesonechoose which device is more appropriate for a design?

Integration-The R380EX includes the RTC, keyboard and mouse controller, EIDE interface, and DIO. The R300EX does not integrate the RTC and keyboard and mouse controller, thereby lowering cost in applications not requiring these features. The R300EX has a regular IDE interface.

3-V operation-The R380EX operates at 5 V or 3.3 V. The R300EX is 5 V (± 10% in each case).

Battery-backed operation-The R380EX selects from either the high-frequency (66-MHz) or the 32.768-kHz clock source for lower-power consumption. Several other power-management functions are also included.

local Bus peripherals-The R380EX directly supports connection of Local Bus peripherals.

DRAM support-The R380EX supports 5 12 KB to 64 MB of DRAM. The R300EX supports 1, 2, 4, 8, or 16 MB.

EDO DRAM support-The R380EX supports both FPM and EDO DRAMs. The R300EX supports FPM DRAM only.

ISA compatibility-The R380EX provides ISA support. The R300EX provides most of the ISA signals.

Intel '386EX DMA support-The R380EX supports Fly-by DMA between the memory and I/O.

Secondary high-speed path to DRAM-The R380EX offers high-speed access to the DRAM from a second processor.

Flash SIMM support—The R380EX supports a mix-and-match of DRAM and flash SIMMs to a total of 64 MB.

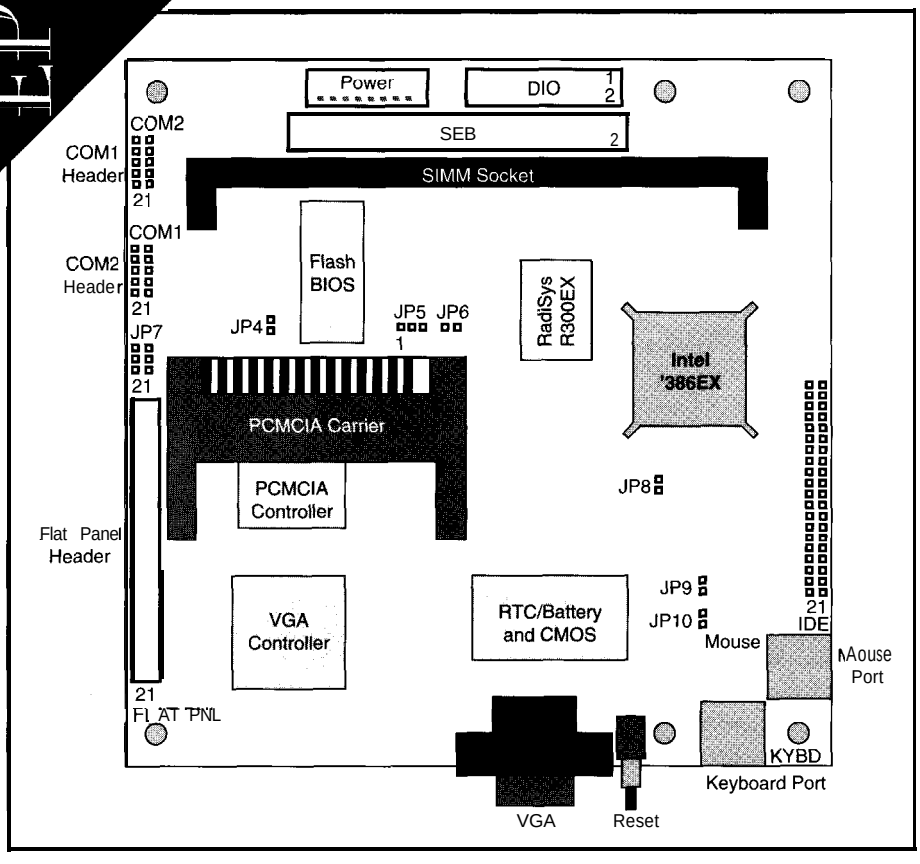


Figure 2: The placement of the primary components on the EXPLR 1/RadiSys EPC-4 1 board shows the board's building blocks.

The beauty of such a system is that it can take advantage of an external VGA plugged onto the SEB interface during development and manufacturing tests. Through planning and design ingenuity, a keyboard controller and even an IDE interface can be incorporated on a daughterboard for development (see Figure 1).

A more fully configured system using the R300EX is the Intel EXPLRI evaluation platform design. RadiSys designed and built the EXPLRI as a proven reference design kit for distribution and promotion of the Intel '386EX and RadiSys R300EX chips.

As Photo 1 and Figure 2 show, the EXPLRI is a single-board computer which demonstrates the Intel '386EX features and showcases its use for low-cost, space-constrained, portable embedded computer applications. The PC-like EXPLR1, which runs most PC-application software, provides a complete, low-cost Intel '386EX system in a proven, easy-to-use, standalone or expandable configuration.

In Figure 3, you can see that the board features a 25-MHz Intel '386EX processor, RadiSys R300EX memory and bus controller, 4-Mb boot-block flash memory, up to a 16-MB DRAM SIMM, as well as a single-slot Type I/II PCMCIA controller.

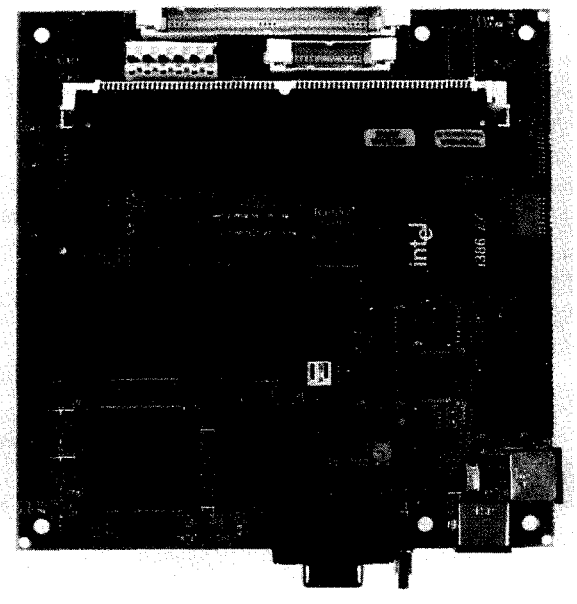


Photo 1: Even with the EXPLR1/RadiSys EPC-4 1's PC-compatible feature set, this 5.5" square board does not need to be densely populated.

The EXPLRI system can stand alone with the onboard VGA/LCD graphics controller, IDE controller, PS/2-style mouse and keyboard ports, two serial ports, RTC with battery-backed CMOS RAM, watchdog timer, and standard PC power-supply connector. Access to system bus signals and digital I/O (DIO) is provided through two expansion headers.

On the software side, the EXPLRI is a PC-compatible system which includes a Phoenix BIOS and Annabooks ROMable DOS 5.0. It runs DOS, Windows 3.1, and even Windows 95 when configured with sufficient memory.

The standard EXPLRI board ships with 1 MB of DRAM in the SIMM socket, but it accommodates 2-, 4-, 8-, or 16-MB SIMMs as well. The supported 2-MB and 8-MB SIMMs are the less common, single-RAS x16-type rather than the more common, dual-RAS x32-type SIMM.

PCMCIA support in the Phoenix BIOS includes SRAM, flash, and some ATA hard-disk cards. The system can boot from PCMCIA, IDE, or the included ROMable DOS in the flash.

While there is no direct facility to connect a floppy disk drive to the EXPLRI board, the ROMable DOS in flash does include **INTERLNK.EXE.** With INTERSRV **.EXE** running on a host PC, **INTERLNK.EXE** provides access to the host's hard disks and floppy drives, allowing a user to load and even run an application from the host's floppy.

The EXPLRI platform easily adapts tointerfacewith many PC/104 cards, thus opening up the wide range of off-the-shelf functionality provided by PC/104card manufacturers (see the sidebar "Connecting the SEB to PC/ 104").

While Intel sells the EXPLRI to individual customers in small quantities through distribution, RadiSys provides the same hardware via the EPC-41 to OEM customers in higher volume. The complete EXPLR 1 Reference Design Kit (RDK) is available from the Intel literature center (see Sources).

The R300EX design was somewhat constrained by its origin in the CPLD of the POS design. Experience with the R300EX led to an enhanced, higher-integration, companion chip for the Intel '386EX—the R380EX.

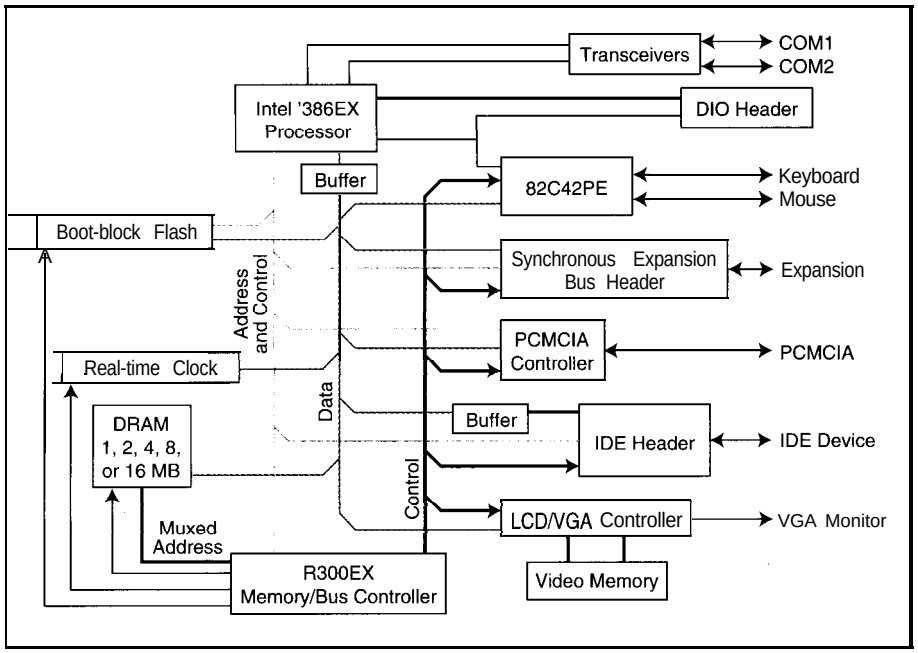


Figure 3: The *block diagram of the* EXPLR ***1/EPC-41*** illustrates the PC-compatible feature *set and* highlights *the* simplicity of interconnection in a system using the ***RadiSys R300EX*** memory and *bus controller chip.*

R380EX'S FEATURE SET

The RadiSys R380EX directly incorporates features needed for an Intel '386EX PC-compatible embedded system design. It provides a simple, low-cost, glueless interface to additional chips like a video controller or a PCMCIA controller, as you can see in Figure 4.

The R380EX functionality directly derives from the PC/AT architecture. It integrates a DRAM controller, keyboard and mouse controller, RTC, enhanced IDE interface (EIDE), and ISA bus controller.

The DRAM controller is compatible with both fast-page-mode (FPM) and extended-data-out (EDO) DRAM and controls 5 12 KB to 64 MB of either DRAM type. It also supports both DRAM and flash SIMMs for greater system flexibility.

The keyboard and mousecontroller and RTC are fully PC compatible. The keyboard controller is a hard-wired state machine, which results in fast response to keyboard commands. While both the RTC and keyboard and mouse controller are integrated into the R380EX, the chip can be configured for an external RTC and/or keyboard and mouse controller if alternate functionality is required.

The R380EXEIDE interface can support the ATA-2 specified programmed I/O mode 3 and 4 for IDE drives at a maximum transfer rate of 8.33 MBps. The R380EX powers up with mode 0 timing to support both older IDE drives that are incapable of interfacing at mode 3 and mode 4 speeds and to support the power-on mode of the newer EIDE drives.

The R380EXISA controller has a separate data bus and implements the "quiet ISA bus" feature. When the R380EX sees an access that is not destined for the ISA bus, it does not drive the ISA data or control signals. Additionally, the ISA refresh cycles may be disabled. This feature reduces overall system power consumption and increases performance.

The power-management capabilities of the R380EX include clock-source switching, halt detection, and SMI event generation. The Intel '386EX's clock is generated by the R380EX. Its source can be switched between the external high-frequency oscillator feeding the CLK2OSC input and the 32.768-kHz RTC oscillator, thereby reducing system power consumption.

When the R380EX is sourcing the 32.768-kHz clock to the Intel '386EX, the external oscillator feeding the R380EX's CLK2OSC input may be powered down to save system power. A programmable clock-restart delay ensures clock-frequency settling when restarting the oscillator from a power-down state.

Connecting the SEB to PC/I 04

The SEB on the EPC-4 1 /EXPLR 1 board provides a set of signals which map directly to the PC/I 04 or ISA bus. Figure i describes the signals involved and the source and destination of each signal on the EPC41 /EXPLR1 board. Information on generating the SEB signals can be found within the EXPLR1 reference design schematics.

There are two sets of address signals on the SEB: the unbuffered address signals routed directly from the CPU (AI 7–A25) and the CPU address signals latched by the BALE signal (SA0–SA16). The SA signals are latched in 74F373 chips by the BALE signal from the R300EX. SAO is a special case in that it is not a latched address signal, but a latched version of *BLE from the CPU, providing the equivalent of AO.

The data signals (DO-D 15) are buffered from the CPU by a 74ACT16245 chip. These same buffered-data signals connect to the data pins of all the chips requiring the data bus. As a result, additional loading of the lines should be kept to a minimum.

The unbuffered control signals that connect directly to the CPU are W/*R, M/*IO, D/*C, *WR, *RD, and *BHE.

The unbuffered control signals that connect directly to the R300EX are BALE, *IOW, *IOR, *MEMW, *MEMR, *IOCS 16, *MEMCS16, and IOCHRDY.

The PWRGD signal is driven by a MAX701 and needs to be inverted to provide the RESETDRV signal. A 74F74 with 22 Ω in series drives the 8-MHz CLKSYS signal.

By including the digital I/O signals in the connections from the EPC41 /EXPLR1 to the PC/I 04, ±12-V and four interrupts are made available. IRQ5 and IRQ12 connect to the CPU.

INT2 and INT3 also directly connect to the CPU and can be configured as IRQ6 and IRQ7, respectively. Note that IRQ12 connects to the 82C42PE keyboard and mouse controller chip. Both IRQ5 and IRQ12 connect to the CL-PD6710 PCMCIA controller chip.

The largest group of unused PC/I 04 signals consists of the DMA signals. The SEB does not include any *DACKx or DRQx signals. As a result, it is recommended that the *DACKxsignals be pulled high and the DRQx signals left floating. The AEN and TC signals also are part of the DMA mechanism and are not supported by the SEB.

The four interrupts available from the DIO are IRQ5, IRQ6, IRQ7, and IRQI2. This leaves IRQ3, IRQ4, IRQ9, IRQ10, IRQ11, IRQ14, and IRQ15 unconnected to the PC/I 04 bus.

While the -5-V signal is rarely used, it can be generated by reregulating the -12 V available as part of the DIO signals if necessary.

The SEB cycle accesses do not use the OWS signal, and it should be pulled high through a resistor.

The *SMEMW and *SMEMR signals can be generated from the *MEMW and *MEMR signals by combining them with a decoded address indicating that the address is below the 1 -MB boundary. These signals can be generated in a PLD.

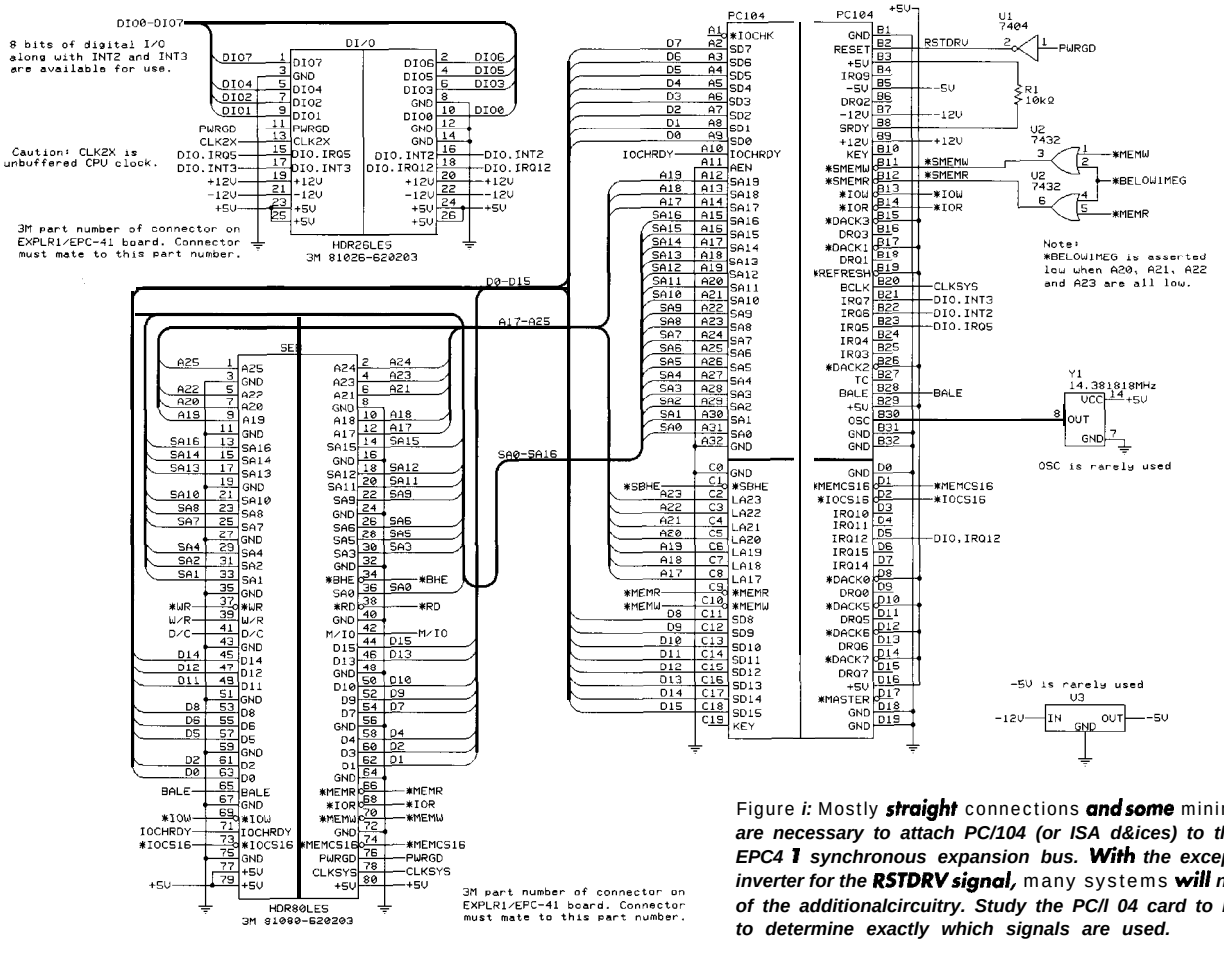


Figure *i:* Mostly ***straight*** connections ***and some*** minimal ***circuitry are necessary to attach PC/104 (or ISA d&ices) to the EXPR 1/ EPC4 1 synchronous expansion bus. With the exception of the inverter for the RSTDRV signal,*** many systems ***will not need any of the additionalcircuitry. Study the PC/I 04 card to be attached to determine exactly which signals are used.***

Signal	Source	Destinations
A25–A17	Intel '386EX	PD6710 PCMCIA controller, RSOOEX, flash
SA16–SA0	74F373	GD8245 VGA controller, PD6710 PCMCIA controller
D15–D8	74ACT16245	GD6245 VGA controller, PD6710 PCMCIA controller, 74ALS244, DRAM SIMM, flash, 74ACT16245, 10-kΩ pull-up
D7–D0	74ACT16245	GD6245 VGA controller, PD6710 PCMCIA controller, 82C42PE keyboard controller, 74ACT16245, DS12887 RTC, 74ALS244, DRAM SIMM, flash, 10-kΩ pull-up (D3–D0 additional GAL16V8)
W/*R, M/*IO, D/*C	Intel '386EX	R300EX
*BHE	Intel '386EX	GD6245 VGA controller, PD6710 PCMCIA controller, RSOOEX, GAL1 6V8
*RD	Intel '386EX	Flash, 2 - 74ACT16245
'WR	Intel '386EX	DS12887, DRAM SIMM through 33Ω resistor
BALE	R300EX	PD6710 PCMCIA controller, 3 - 74F373
*IOW, *IOR	R300EX	GD6245 VGA controller, PD6710 PCMCIA controller, GAL1 6V8, 2 - 74ACT08
*MEMR, 'MEMW, *MEMCS16	RBOOEX	GD6245 VGA controller, PD6710 PCMCIA controller
IOCHRDY	RBOOEX	GD6245 VGA controller, PD6710 PCMCIA controller, IDE Port, 1 -kΩ pull-up
*IOCS16	RBOOEX	GD6245 VGA controller, PD6710 PCMCIA controller, IDE Port, 330-Q pull-up
PWRGD	MAX701	GD6245 VGA controller, PD6710 PCMCIA controller, 82C42PE keyboard controller, RSOOEX, flash, IDE port through 33 Ω, DIO connector
CLKSYS	74F74 through 22 Ω	SEB connector only

Table i: This tab/e shows the signal loading, ***with*** source and destination, ***for*** each of ***the*** signals ***that*** are part of the ***SEB. Even*** without the schematics, by having this information readily available, ***the*** designer can make informed design decisions when ***attaching*** devices to ***the*** SEB.

Included in the DIO signal set are eight signals (DIO0–DIO7) connected directly to the CPU. These signals can be used individually or as a group. The DIO PWRGD signal is identical to the SEB PWRGD signal.

The DIO CLK2X signal connects directly to the CPU and must be used with great care. This signal must be buffered extremely closely to the DIO connector (if it is used) to ensure that CPU CLK2 signal doesn't degrade. Degradation of this signal causes problems which are very difficult to isolate. Use of the CLK2X signal is not generally recommended.

Connection to the DIO signals and the SEB signals on the EPC41 /EXPLR 1 is made by mating to the onboard highdensity 3M connectors.

The EPC41 /EXPLR1 board uses a 25-MHz B-Step Intel '386EX processor. The 25-MHz system frequency results in an equivalent 8-MHz SEB speed. With a 33-MHz '386EX C-Step processor instead, the resulting SEB has a 1 O-MHz equivalent speed. While this is noteworthy, the vast majority of devices one might place on the SEB should have no problem handling the increased speed.

Table i shows the loading placed onto the SEB signals by the EPC41 /EXPLR1 circuitry. This loading must be taken into consideration when connecting to signals on the SEB connector.

Table ii shows the loading placed on the DIO signals by the EPC41 /EXPLR1 circuitry. This loading must be considered when connecting to signals on the DIO connector. Note that the CLK2X signal is directly attached to the CPU and the signal quality must not be degraded.

The *SMEMW signal generates by logically ORing *MEMW with A20, A2 1, A22, and A23. The *SMEMR signal generates by logically ORing *MEMR with A20, A2 1, A22, nd A23.

The *REFRESH and *MASTER signals are rarely used and are not available as part of the SEB.

Another infrequently used signal is OSC, which is a 14.3 181 8-MHz clock. If necessary, it is easily generated by the addition of an oscillator to the board providing the mechanical conversion from the SEB/DIO to PC/I 04.

Signal	Source	Destinations
DIO7–DIO0	Intel '386EX	DIO connector
PWRGD	MAX701	GD6245 VGA controller, PD6710 PCMCIA controller, 82C42PE keyboard controller, R300EX, flash, IDE port through 33 Ω, SEB connector
IRQ12	82C42PE keyboard controller, DIO Port	Intel '386EX
IRQ5	PD6710 PCMCIA controller, 82C42PE keyboard controller	Intel '386EX
INT2, INT3	DIO connector	Intel '386EX
CLK2X	ICD2023SC	Intel '386EX, RBOOEX

Table ii: Hoving signal loading information readily ***available,*** the designer can make informed design decisions when attaching devices to the DIO. ***Particular*** caution should be exercised to ensure signal fidelity of the CLK2 signal since degradation ***at*** the processor can result in many hours of needless ***troubleshooting.***

The R380EX has four user-programmable I/O chip selects in addition to those provided an the Intel '386EX. There are 16 bits of individually programmable digital I/O along with six bits of digital output. Additional function blacks handle the '386EX halt and shutdown cycles, external LEDs, and speaker control.

A PC-compatible Part-B register is included in the R380EX. The main and alternate functions of many of the multiplexed pins are controlled through internal registers. The R380EX operates from either a 5- or 3.3-V supply.

DESIGN TYPE AND THE R380EX

The R380EX can be applied to the simple data lagger discussed earlier, but an enhanced version puts the flash SIMM capabilities to goad use by saving the postprocessed data in the nonvolatile memory. While a simple data lagger can use the R380EX, its greater integration and functionality are better used in designs of greater complexity and that take advantage of the ISA bus.

A high-end system using the R380EX takes advantage of the flash SIMM and large DRAM capability, along with the ability to include bath Local Bus and ISA bus peripherals (see Figure 4).

In addition to the boat-black flash far the BIOS, the system is configured with a combination of DRAM and flash SIMMs totaling up to 64 MB, which enablesdiskless operation while maintaining a considerable amount of program and data storage. **Signals are** provided for updating the BIOS flash under program control, as well as modifying the flash SIMM.

The EIDE interface can support two IDE devices, providing access to a full array of IDE hard drives and CD-ROM drives.

The R380EX DRAM interface lets an alternate master access the DRAM, thereby providing the equivalent of a dual-ported DRAM. This feature is particularly useful in multiprocessor embedded systems.

Alternatively, if a fully dual-ported memory system is unnecessary, the DMA channels in the '386EX can move data between the DRAM and the ISA bus. The R380EX coordinates the state machines for the DRAM and ISA controllers, thus providing the properly timed signals to mate the DRAM cycles with the ISA I/O cycles.

The VGA/LCD display system makes use of local Bus access capabilities of the R380EX, while the PCMCIA and Ethernet controllers use the ISA interface signals.

The R380EX includes PC Port-B function and the capability to route the PCMCIA modem speaker signal through the R380EX and out to the speaker. A pair of LEDs can be driven by the R380EX and used as indicators or'outputs for an optical link.

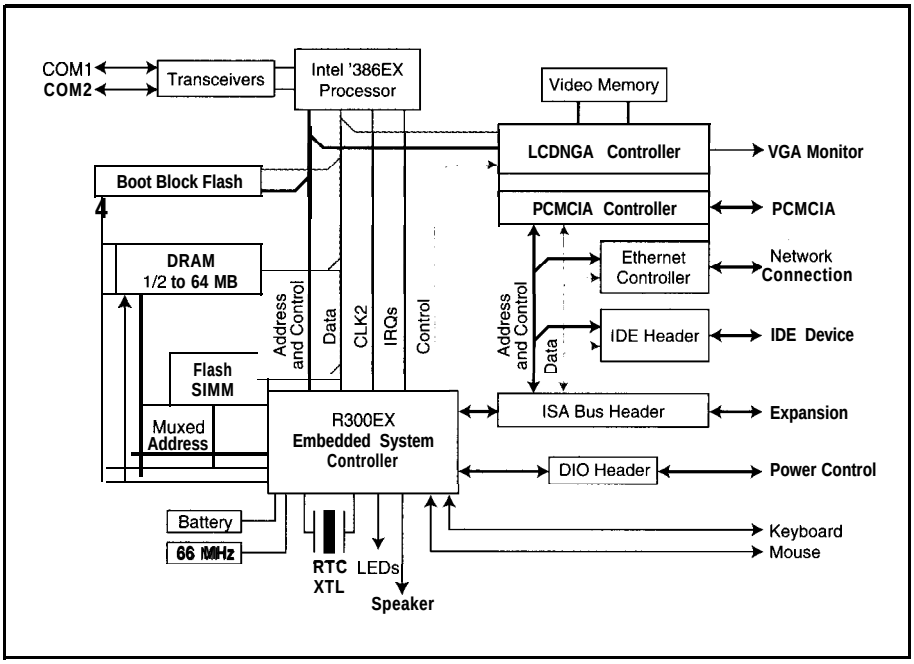


Figure 4: The ***RadiSys R380EX*** embedded system controller is used in a higher-end, embedded PC-compatible system. On the left, you see the core of a complete system with the ***CPU, R380EX,*** and memory. The right side shows the added features which ***take*** the system from a ***PC***-compatible core to a fully configured system complete with PCMCIA, Ethernet, and IDE interface.

Floppy Disk Drives and Intel's '386EX

Your desktop PC has four 8-bit DMA channels as part of its chipset and uses DMA channel 2 when doing a floppy disk transfer. The Intel '386EX only has DMA channels 0 and 1, which causes a problem. A standard off-the-shelf BIOS cannot access a floppy disk drive without DMA channel 2.

In addition, a number of operating systems and applications directly access the hardware, rather than making BIOS calls to access the floppy. Simply modifying the BIOS may not have the desired effect, depending on the software used.

To modify the BIOS, you can change the floppy access routine to use a different DMA channel. While this might work, you still end up with nonstandard floppy accesses.

The second alternative is to modify the BIOS routines for programmed I/O. There are several vendors who sell a BIOS which uses programmed I/O for floppy accesses.

This alternative is usually the best since programmed I/O generally results in the equivalent floppy throughput of the DMA channel under DOS. However, both of these software alternatives have the same problem if the OS or software attempts to access the hardware directly.

The third alternative is to add another DMA machine to the system. An 82C37A can connect to the Local Bus through buffers. A state machine would be required to request the '386EX bus using HOLD/HLDAprotocol. Itwouldalsogenerate *ADS, wait for READY, and control the buffers.

The equivalent of a 74LS6 12 memory mapper would have to be connected to the Intel '386EX Local Bus, too. In addition to the cost and real estate required for an 82C37A and PLD, this is not a trivial task.

Considering these difficulties, let's step back and examine both the need for accessing a floppy and the alternatives. Most of the time, a floppy is used to move data on and off the machine and only on rare occasions actually runs a program. In an embedded system, the need to move data is quite common, but by its very nature, it's unlikely an embedded PC will execute a program from a floppy.

There are two particularly attractive solutions to the problem of moving data on and off an embedded system in a floppy-like manner. The first is to use a PCMCIA SRAM card, flash card, or even ATA hard drive.

The second solution is the one used on the EXPLR 1 board. Use the DOS **INTERLNK.EXE** and **INTERSRV.EXE** programs.By running **INTERLNK** on the embedded PC and **INTERSRV cn** the host system, a serial link can be established that enables the embedded system to access not only the host system's floppy disk drives, but also the host system's hard drives.

This link allows transfer of data between systems, and even allows the embedded PC to execute programs that reside on the host's disks. Obviously, it executes slower than if it were local to the embedded PC, but it is certainly a functional alternative.

The system uses the R380EX's internal keyboard and mouse controller along with the internal RTC. The internal RTC has circuitry to drive an external 32.768-kHz crystal and has a separate RTC power pin for use with an external battery switch-over circuit.

The ISA connectors provide an opportunity to attach ADC, DAC, or isolated digital I/O cards. Parallel printer ports can also be added via the ISA bus.

A floppy controller card can be connected, but a special BIOS configuration handles the programmed I/O floppy access or a nonstandard PC DMA channel. This extra work to handle the floppy is necessary because a plain-vanilla PC uses DMA channel 2 for the floppy, and the '386EX provides only DMA channels 0 and 1.

As you can see in Figure 4 and its system description, the R380EX provides a well-planned set of features. The R380EX can be applied to a wide variety of systems ranging from the simplest system, requiring only the DRAM controller and RTC, to much more complex systems fully loaded with features.

AND ALL THIS MEANS

The main difficulties in building embedded PCs stem from components lacking long-term availability. BIOS issues also surface when one starts modifying features of the "standard" PC featureset. The RadiSys R300EX and R380EX address the long-term availability issue for companion chips for Intel's '386EX and greatly simplify the design of a '386-class embedded PC.

RadiSys recognizes that chipsets are not the only chips in embedded PCs which need long-term availability. They are addressing this issue with additional chips for the embedded market.

RDKs, available through the Intel literature center or local distributors, provide a starting point for designing '386EX-based PC-compatible systems. For companies that would rather focus their resources on their core business than on building a PC-compatible embedded computer, RadiSys happily provides design and manufacturing expertise for production-volume perfect fit embedded PCs. EPC

Brad *Reed works as a component products application* engineer at Radisys. Although *he* worked on embedded *designs at Tektronix, his familiarity with embedded* PC design *launched when he joined first* Microtek and then Radisys. He may be reached at *brad.reed@radisys.com.*

SOURCES

R300EX memory/bus controller and **R380EX** embedded system controller
RadiSys Cp.
15025 SW Koll Pkwy.
Beaverton, OR 97006-6056
(503) 646-1800
Fax: (503) 646-1850
BBS: 15031 646.8290
http://www.radisys.com/

EXPLR1 Reference Design Kit and schematics (272768-001)
Intel Grp.
5000 West **Chandler Blvd.**
Chandler, AZ 85226
(800) 548-4725
Fax: (708) 296-3699
http://www.intel.com/

SEB and DIO connectors on EPC-4 1 **/EXPLR1 board** (8 1080-620203 and 8 1026-620203)
3M
6801 River Place Blvd
Austin, TX 78726
(800) 328-1368
Fax: (512) 984-2211

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

David Chanoux

Embedded PCs

Embedded Systems for Weight and Force Measurement Applications

Until recently, there has been no tool to integrate weighing sensors with embedded PCs. David lets us in on some recent improvements in the link between the PC/104 standard and loud-cell technology.

Suppliersof real-time weighing and force measurement systems now have an integration tool to connect weighing sensors to embedded PCs. Three criteria have been met, paving the way for PC-based weighing applications.

First, good-quality, high-performance, ADCs are changing weighing limits. Applications can be implemented differently.

Second, microcontrollers with easy-to-use bus interfaces allow weighing applications to be segmented, using the vast processing, graphics, storage capabilities, and presence of today's PCs to advantage.

And third, the emerging embedded-PC bus standard, PC/I 04, enables effective industrial packaging, so the PC is installed where weighing takes place.

The Scanning Devices PC/I 04-Compliant Load-Cell Controller shown in Photo 1 uses these new developments to enable PC-based weighing systems.

This article describes weight- and force-measurement applications, tracing their evolution to today's microcontroller-based systems, and projects the effect of these developments on tomorrow's embedded PC-based weighing systems.

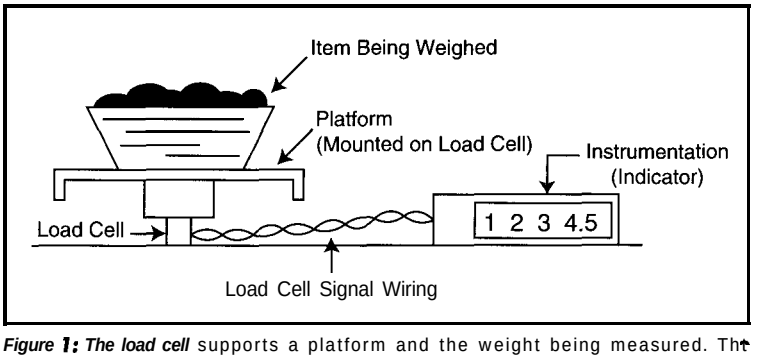


Figure 1: The load cell supports a platform and the weight being measured. The load cell is connected to an indicator which provides bridge ***excitation*** and converts the signal to weight. Notice the load cell is down under ***the*** weight!

TODAY'S PC IN WEIGHING SYSTEMS

Until recently, electronic weight- and force-measurement systems have been the domain of microcontrollers. The PC has been kept at arm's length.

The highly valued portion of a weighing instrument is its ability to precisely measure a relatively small analog voltage signal representing the weight or force. The serial link to a PC for data or process parameters used for measurement was of less interest and often ended up being one-way. From the PC's point of view, it was input only!

System builders requiring the capabilities of both PCs and weighing indicators had no choice but to connect them via either EIA (RS-232, RS-422) or 20-mA current loop interfaces, with the PC system designer accepting the

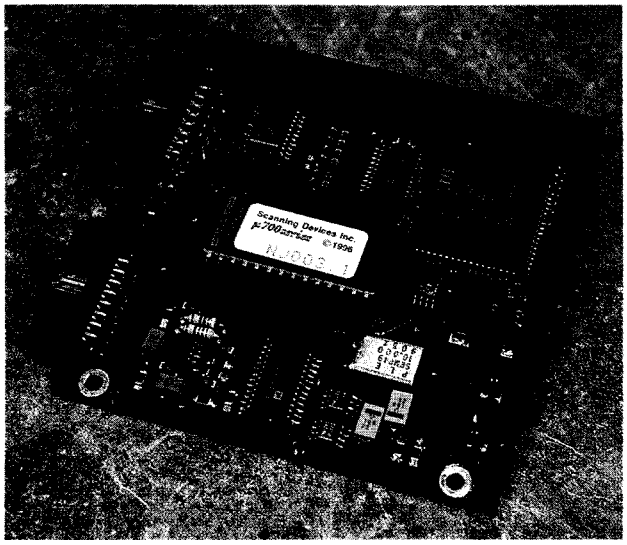


Photo I: Specialized integrated circuits-selected and configured for by application-are controlled with specialized firmware. They keep the part count at a minimum and allow implementation in the PC/104 compact format.

limited data-exchange capabilities found in most digital indicators. (The weighing industry uses "indicator" to describe an instrument working with a load cell to measure and display weight.)

But first, let's get some background on weight and force measurement.

MECHANICAL WEIGHING

Mechanical weighing systems are typically spring-based or lever-based instruments. A bathroom or produce scale is a common spring-based scale which deflects a spring to rotate or displace a dial, thereby displaying the weight. These scales are notoriously inaccurate.

Lever-based instruments balance the unknown weight with a known weight on the end of a lever arm and are accurate enough for legal-for-trade scales. However, reading and recording the weight is left to the viewer's interpretation.

ELECTRONIC WEIGHING

Electronic weighing centers around strain-gauge transducers or load cells, sensing instruments which convert applied force to a resistance change. Figure 1 illustrates a typical electronic weighing system consisting of platform, load-cell transducer, and indicator.

The transducer isconfigured as a Wheatstone bridge made up of four resistive elements. Two of these strain gauges are mechanically react positively to applied force, while the remaining two react negatively to the same force. An excitation voltage applied to one pair of bridge terminals transforms to a measurement-signal voltage on the other.

With no force or weight applied, the bridge is balanced and the signal voltage **is** zero. As force is applied, two strain gauges are put in tension, the other two in compression: their effective resistancechanges in opposite directions. The Wheatstone bridge produces a signal voltage proportional to the applied force.

toad-cell specifications include capacity (the full load, such as 100 pounds) and output voltage ratio. The output is typically 2 mV/V (i.e., 2 mV of signal voltage per volt of excitation at capacity). If the excitation is 10 V, the load-cell signal is 20 mV with 100 pounds applied.

The many capacities, mechanical configurations, and special features available with load cells provide almost unlimited applications. Photo 2 **shows** a typical $350 beam-style load cell applicable to low-profile scales. Excitation and the corresponding measurement-signal voltages are AC or DC, giving further design discretion to the transducer engineer.

By nature, the Wheatstone bridge enables load cells to be used individually or combined for additive weighing. For example, a platform may be supported on four load cells which are electrically connected so that their signals sum to one equivalent load cell. The single measurement signal represents the total weight on the platform, regardless of weight distribution.

THE CHALLENGE

At first glance, the application seems straightforward: pass the signal through an ADC, do some math, drive a display or a printer,and ifcommunicationsareneeded, write to the serial port. Any PC with a general-purpose analog-input channel suffices. Anybody could do this, you say.

Not so fast! Let's introduce some real-world requirements and complications to illustrate why the PC-load-cell combination is not installed in every scale. Let me also describe how a functional indicator deals with these complications. The Scanning Device Load-Cell Controller illustrated in Figure 2 shows the required components. Each has a purpose-bypassing any one compromises performance.

THE REAL ISSUES

The signal is differential and its magnitude is typically measured in millivolts. A typical analog-input channel in data-acquisition systems delivers 12 bits of signal precision, measured with the assumption that the input signal spans a O-5-V range. That is, the digital conversion is a 12-bit representation of a 5-V input.

If the input is only 5 mV, (0.001 of the ADC's input range, but not atypical of a load-cell signal), the 12-bit conversion result would be ten bits equal to zero followed by two bits of significant data.

Clearly, an input-amplifier stage is required. An input-amplifier stage consisting

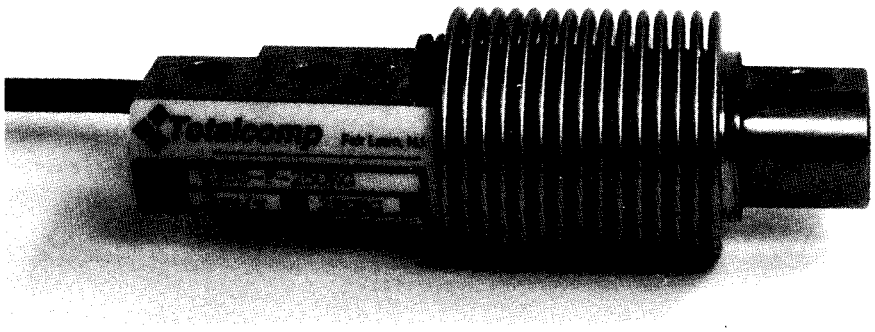


photo 2: The load cell converts force to an electrical signal. This sensor is the heart of electronic weighing systems. *It* consists of a strain-gauge transducer designed to produce a resistive change with applied force. The load cell is configured as a Wheatstone Bridge, allowing the resistive change to be measured precisely.

of an instrumentation amplifier converts the small differential input into a range acceptable to an ADC. The amplifier also provides differential input and gain set by a single external resistor. Its offset control puts the amplifier output in the right range.

The Analog Devices Series 620 is an effective instrumentation amplifier for this application. It is functionally equivalent to the three-amplifier configuration on the right in Figure 3. Fewer parts, reduced noise, and controlled gain favor the instrumentation amplifier most times.

Traditional electronic weighing systems offer input-amplifier stages with dual adjustments. A technician can set the operating points of the input-amplifier circuitry-zero and span-via some means.

The zero adjustment removes dead weight-that not significant for the measurement-from the hardware. The zero adjustment sets the input amplifier to produce a minimum voltage when only the dead weight is applied.

The span adjustment sets the gain of the input amplifier so that the maximum voltage is generated when maximum weight of interest is applied.

But if you are designing an embedded system, you don't want to provide adjustments. Why are zero and span necessary? Precision.

Quality load cells achieve precision to 1 part in 5,000. Good ones do even better. To put that in the perspective of ADC specifications, that's 14 bits of precision in the result. That statistic assumes the load-cell signal spans the full input range of the ADC. In most cases it doesn't.

Minimally, an inputumplifierdesign must allow for overrange voltages and negative measurements while producing a signal so a converter can achieve 16-bit precision. The high-speed 12-bit sample-and-hold converter common on many computer I/O modules is just not the right tool for this task. What should we use?

ADCs using the sigma-delta technique are capable of up to 24-bit precision (e.g., the Analog Devices AD7710 Series converters). They use successive approximation to achieve low-frequency measurements with high precision, exactly the characteristics needed for weighing.

Conversion rates of 50 Hz are often adequate in static and quasistatic weighing. Giving up speed in return for precision is clearly the right choice for weighing applications.

The Analog Devices AD771 2 Sigma-Delta Converter (see Figure 4) with serial digital interface lets you configure performance for applications. Surrounding the converter is linear-input circuitry tuned to achieve high precision with theacceptable speed and digital processing to interpret the conversion results.

Isn't 24 bits overkill? No. Let's examine how we can use the precision.

Scanning Devices tested many good-quality load cells to determine how much precision can reasonably be expected. Without presenting data, let's assume that a goodquality load cell delivers 18 bits of significant data (i.e., 1 part in 262,000).

With a 24-bit conversion on a full-scale signal, we find 18 bits of significant data and 6 bits of random noise. We can argue at length about the data, how much significance can be extracted, and by what means. But, let's assume 18 bits and address the zero and span adjustments.

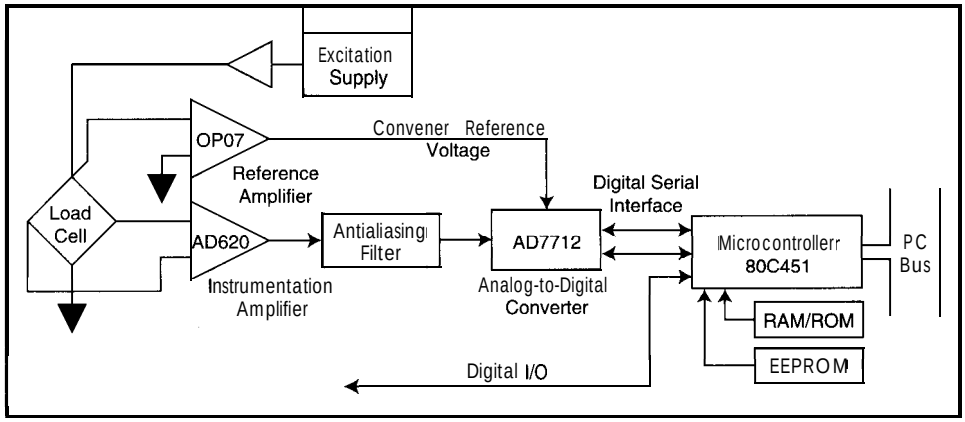


Figure 2: Key components of the Scanning Devices load-Cell Controller genemte load-cell bridge excitation, condition the ADC *with* signal and reference, process the digital measurement signal, and respond to the PC via PC/ *104* bus. *Onboard* processing and memory enable real-time measurement and control under PC supervision.

Making the weighing range span the converter's input-voltage range assumes the converter is the limiting factor in measurement precision. If the weight range of interest spans 4 V (e.g., 0.5-4.5 V at the input to the converter), the converter produces a result with some significance.

If the same weight range of interest spanned 1 V at the input to the converter, the result has only one-fourth the significance. The result is equivalent to taking the conversion from the 4-V case and shifting it two bits to the right, causing the two least significant bits to be lost. Assuming the converter is the limiting factor, the zero and span adjustments prevent this loss of significance, maximizing measurement precision.

But if we have more precision than necessary, why expose the application to adjustments? toad cells are specified so the input range can be determined for a given load cell,

Let's make the input amplifier fixed so it produces a reasonable span over the full range of the load cell. Also allow for both overrange and negative sig-

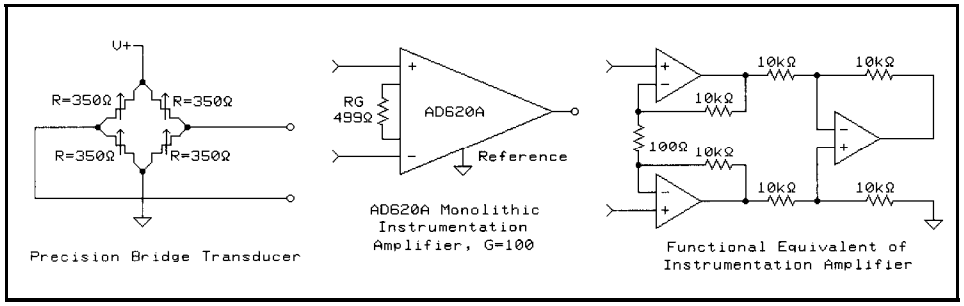


Figure 3: The instrumentation amplifier provides differential input, controlled gain, and offset. *It* is functionally equivalent to the three-amplifier configumtion shown. Performance and cost issues guide selection of the best input-amplifier configuration.

nals. The application's range of interest may be small, one fourth, one eighth, or even less of the load-cell capacity.

We effectively right-shifted the converter result by two, three, or however many bits representthe unused converter input range. But if the least significant six bits are random noise, no significant data is lost.

We used the extra precision in the ADC to eliminate the necessity for input-amplifier adjustments. We can put away the little screwdrivers and treat the load cell like a real computer peripheral!

Except-the measurement signal is at the load cell. The excitation-voltage and millivolt-level measurement signals must be routed to and from the measurement point, which might be some distance and through unfriendly environments. We don't have digital signals at the load cell to insulate the measurement from these conditions. Instead, load cells use excitation sensing.

To compensate for possible voltage drops in the excitation wiring, load cells often have "sense" wires, connections to the excitation terminals which allow the weighing system to measure the applied excitation voltage at the load cell as well as the signal weight.

Remember, the load cell signal is specified as millivolts per volt of excitation. If the excitation voltage changes due to noise or other causes, voltage drops in the cable, the measurement signal changes in proportion even if the applied weight does not.

How to compensate for excitation variation? The sense signal is brought into the measurement system via high-impedance inputs so that little current flows in the sense leads. The sense signal creates a reference for the ADC, which in turn defines the ADC's unit of measure. **Think of** the conversion as a digital number times the unit of measure defined **by the** reference. Changes in theexcitationarecompensated by equivalent changes in the converter's reference.

At last we have a measured signal in digital form. We've used an instrumentation amplifier, high-precision ADC, and excitation-sensing operational amplifiers for the converter's reference. We have converter data in the onboard microcontroller, extracted with the portion of code detailed in Listing 1 and 2.

ENTER THE PC

Traditionally, the process of weighing could be broken down into four more steps: taring, conversion to units, displaying, and controlling. These steps were hard-coded into the indicator with no opportunity for adjustment.

With the entrance of the PC, these steps have become more modular. The developer can now program them to be automated or user directed.

I've broken the steps into more detail so you can see the PC's new, more active role:

- take out the tare

The weighing system usually starts at some nonzero weight: a platform or the load cell's mechanical mount, the box or carton for the item to be weighed, or the

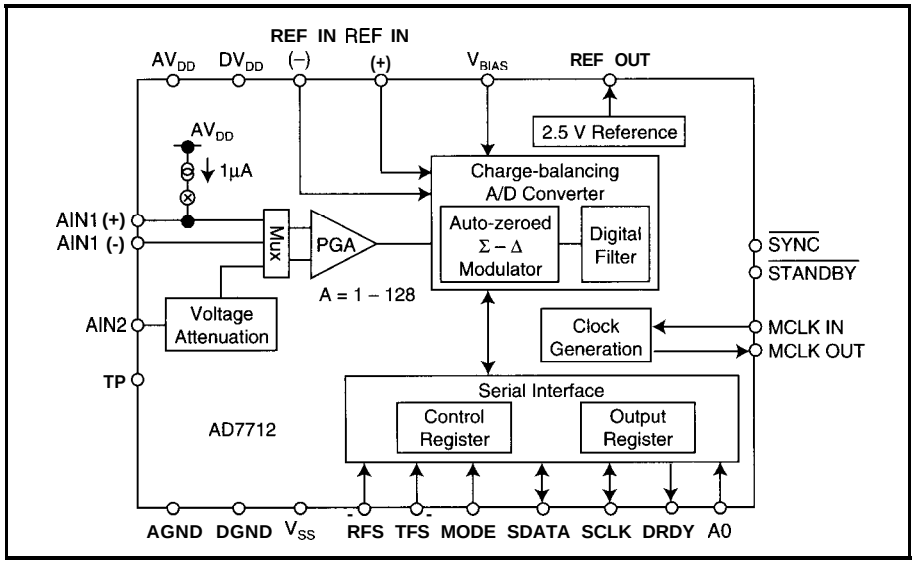


Figure 4: The Sigma-Delta ***converter combines a single-bit*** modulator and DSP filter to achieve very accurate results. The microcontroller interface is serial, consisting of four control lines in addition to clock and data.

first ingredient in a two-ingredient recipe.

The measurement involves taking data from the load cell and storing and computing a tare or zero weight which is subtracted from the gross measurement to obtain the net measurement.

When can the tare be measured? Having the user push a button when the platform is empty works for a microcontroller. Just route the button to a port pin and set up a loop to interrogate the button.

However, if the PC generates a tare command, the system has more flexibility. Tare could come from a PC procedure, the user interface, or any other means.

Here, we encounter the first need for bidirectional data exchange. The PC knows or can easily find out when to tare.

- convert to engineering units

The load-cell output is a voltage. The weight or force is in some other unit, perhaps pounds or kilograms. The digital indicator multiplies to generate a weight in units of interest. But how are the calibration factors used in the derived calculation, and where are they stored?

The calibration procedure of an indicator-load-cell combination specifies the calibration constants for calculating the measurement from raw data. The PC can be programmed to take an operator through the procedure and then store the results, download the data on application startup, or take the raw data and compute the weight itself. Suddenly, we have several options.

- display the data

Display is easy for a microcontroller as long as the display is on the board or in the same box with the microcontroller. Many varied digital displays are available with easy-to-program display-driver chips.

Displaying data remotely is a different problem. let the PC interpret weighing data, add to it date, time, batch number, or whatever else is relevant, and send the data for display, print, or incorporation into a report at another location.

- take a control action

Often, a measurement leads to process action. For example, if the weight is not between 1 and 2, reject mechanism. This kind of control action is easy for micros.

Listing 1: *This* subroutine reads from the AD7712, moving three bytes to the address in *pointer* **r0.** *Note* **the** *signal names correspond to pin designators on the AD7712* **(see** *Figure 4).* **Control signals** *select* **the** *source of the data,* synchronize the devices, *and* clock **the** data.

```
; serin()
        .set    serin,h'08eb

; AD7710 Interface
        .equ    P4,h'c0
        .equ    P4.0,h'c0
        .equ    P4.1,h'c1
        .equ    P4.2,h'c2
        .equ    P4.3,h'c3
        .equ    P4.4,h'c4
        .equ    P4.5,h'c5
        .equ    RFS,P4.0
        .equ    TFS,P4.1
        .equ    DRDY,P4.2
        .equ    A0,P4.3
        .equ    SCLK,P4.4
        .equ    SDATA,P4.5
; operates the AD7712 in external clocking mode, mode=0
; uses r0 as pointer to data memory: r2,r3 as loop counters
; push/pop accumulator
; returns with data in @r0+2, @r0+1 @r0    24 bits
; data transfers MSB first

        .org  .serin
serin:
        clr     sclk
        setb    sdata           ; enable data line
        mov  a,  r0
        push    acc
        inc     r0
        inc     r0
        mov     r1, #3          ;load byte counter

wait:   jb      drdy,wait       ;make sure data ready is low
        clr     rfs             ;enable
mvbyte: mov     r2,#8           ;load bit counter
mvbit:  mov     c,sdata         ;move data from port
        rlc     a               ;build byte in accumulator
        setb    sclk            ;cycle clock
        clr     sclk
        djnz    r2,mvbit        ;test bit cntr, jump back if not done
        mov     @r0,a           ;store byte
        dec     r0
        djnz    r1,mvbyte       ;test byte cntr, jump back if not done
        setb    rfs
        pop     acc
        ret
```

But, where do the setpoint values come from? Are they programmed, stored in a file, or downloaded from the network?

Notice that as we come further down the list, the task extends beyond the traditional microcontroller. The application measures more than just a voltage generated by the load cell. The PC has readily available resources to augment the application while the microcontroller does not.

AN OFF-THE-SHELF SOLUTION

Scanning Devices manufactures a PC/I 04compliant load-cell controller that takes advantage of these developments. Model 754PLC3 uses an 805 1 -derivative microcontrollerwith a handshaking port as the PC bus interface. A bidirectional 8-bit port for transfers data to and from the PC bus with a 2-bit port-control status register.

The load-cell controller is an 8-bit, stack-through PC/I 04 module. The module provides excitation, sense inputs, and measurement inputs for a single load cell. The key to measurement is a high-precision ADC, Analog Devices AD7712. It also offers four digital inputs and four digital outputs for process monitoring and control.

But what takes it 'a step ahead of the system with a digital indicator installed on the serial port is the rich set of data-exchange and control transactions pos-

sible between the PC and load-cell controller. These transactions let the PC control the microcontroller and converter operations so the developer can maximize resources for each part of the application.

In Photo 3, you see the main-menu screen of the Lancelot PC-based demo program. The screen shows buttons for calibration and filter setup, setpoint control, mode selection, digital input and output control, and measurement data transfer and display. All of these can be under user or program control.

Scanning Devices' software runs in the onboard microcontroller to set up and control the ADC, implement postconversion digital filtering, compute weight from raw conversion data, compare weight to four setpoints, monitor four digital inputs, and set four digital outputs. This capability and an onboard EEPROM for permanent storage lets the indicator module run independent of the PC for much of the application.

However, when the PC writes a command to the port address, it's a different story. The 754P-LC3 is passive regarding the PC bus. All transactions are initiated by the PC. When the PC writes a character to the indicator module's address, the micro interprets the command and either accepts data from or transfers it to the PC.

Table 1 shows the thirteen transactions that are currently implemented, all related to setup and data transfers. Compare this level of PC communications and control to the one-way serial output of traditional indicators. The commands in Table 1 refer to the data variables stored and used in the indicator module (see Table 2).

By making the raw data and weight available, users can calculate weight, introduce filters, and compare more setpoints.

Listing 2: This subroutine writes to the AD77 12, ***moving*** three bytes from the **address in pointer *r0.*** The write **destination** is determined ***by A0,*** either control register or calibration register.

```
; serout()
        .set  .serout,h'090f

; AD7710 Interface
        .equ  P4,h'c0
        .equ  P4.0,h'c0
        .equ  P4.1,h'c1
        .equ  P4.2,h'c2
        .equ  P4.3,h'c3
        .equ  P4.4,h'c4
        .equ  P4.5,h'c5
        .equ  RFS,P4.0
        .equ  TFS,P4.1
        .equ  DRDY,P4.2
        .equ  A0,P4.3
        .equ  SCLK,P4.4
        .equ  SDATA,P4.5

;r1 is used as a byte counter, 3 bytes in each transfer
;r2 is used as a bit counter
;r0 is loaded with a pointer to the output data prior to
;calling the serout function
;data is tranferred MSB first

        .org  .set-out
serout:
        mov   r1,#3         ;load byte counter
        inc   r0            ;point to msb
        inc   r0
        clr   sclk
        clr   tfs           ;enable interface
mvbyte: mov   r2,#8         ;load bit counter
        mov   a,@r0         ;load first byte of data
mvbit:  rlc   a             ;move msb to the carry bit
        mov   sdata,c       ;mov the carry bit to the port
        nop                 ;delay one cycle to avoid race
        setb  sclk          ;cycle the clock
        clr   sclk
        djnz  r2,mvbit      ;test bit counter, jump if not done
        dec   r0            ;decrement data pointer
        djnz  r1,mvbyte     ;test byte counter, jump if not done
        setb  tfs           ;disable interface
        ret
```

By providing access to indicator-module digital inputs and control over the outputs, PC users control the whole process.

So, while the module indicator may run independently of the PC, it may also be configured as a special-purpose data-acquisition module with calculationsand logic done completely in the PC. The choice is up to the system designer.

WHY PC/ 1041

Because of the potential for industrial packaging. Scanning Devices is involved in industrial weighing for process control. Such weighing occurs under the material, not on the desktop. Dirt, moisture, electrical noise, and other irritants make short life of unprotected desktop PCs. Protecting desktop PCs is costly.

The PC/I 04 consortium has specified a small form factor with relaxed bus-drive characteristics, so systems can be built with smaller power supplies, limited cooling requirements, and smaller enclosures. Development of a standard has encouraged many companies to build component-level products.

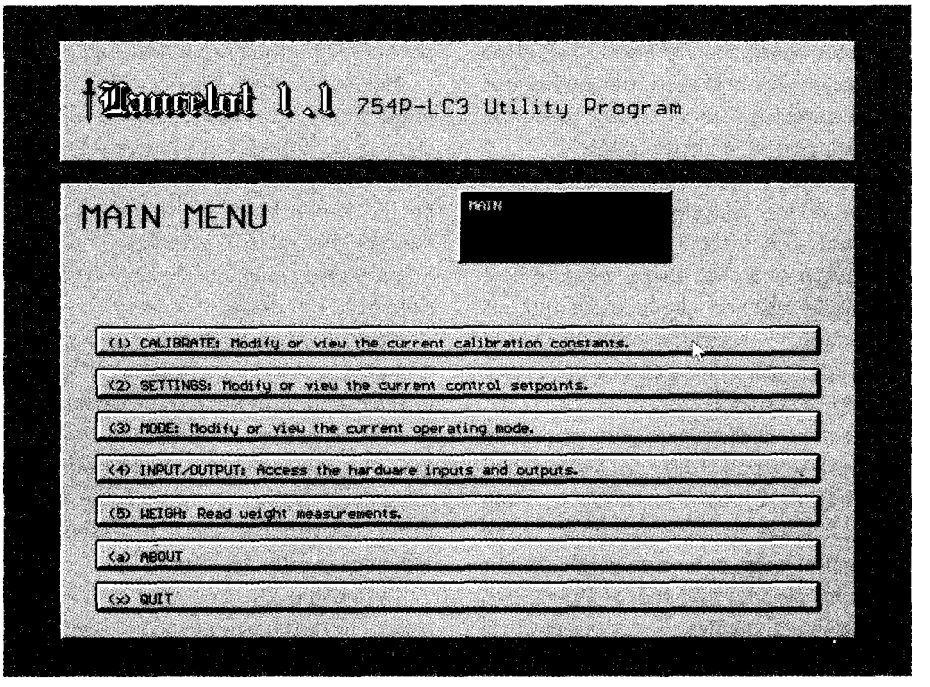


Photo 3: Scanning ***Devices' Lancelot Main*** Menu shows the functions available to the user on one screen-calibration, ***setpoint*** *entry,* mode *selection, digital* input and output *control,* and *weight* measurement. This C++ demo serves as both a diagnostic installation aid for the controller and a programming tutorial on application techniques. Instead of being user- or menu-driven, these functions are *easily* implemented as real time.

By using a micro-based indicator module with extensive PC-data interchange and control capabilities, you can integrate weight- and force-measurement capability with embedded PCs. The load cell becomes justanother PC peripheral. PCQ.EPC

Special thanks to *Totalcomp* ***Inc. for their generous donation of Photo 2.***

David Chanoux is president of Scanning Devices, a manufacturer of sensors, ***instruments, and controls for industrial automation. I-/e's worked with industrial and real-time computers*** **since** *1971* at ***Scanning Devices, Digital Equipment, and IBM. He may be reached at (6 17) 272-5*** *135* ***or*** *scanning@tiac.net.*

GetParameters	transmit the current values of the four process setpoints
SetParameters	receive new values for the four process setpoints, store in EEPROM, and use
GetCalibration	transmit the current value of the load-cell calibration constants
SetCalibration	receive new values for the load-cell calibration constants, store in EEPROM, and use
GetMode	transmit the current operating mode
SetMode	receive a new value for the operating mode, store in EPROM, and use
GetWeight	transmit net weight data in engineering units
GetData	transmit raw weight data (conversion output)
GetIOStates	transmit the state of the indicator module's four digital inputs
SetIOStates	receive new values for the indicator module's four digital outputs and set outputs
GetFilter	transmit the characteristics of the onboard digital filter
SetFilter	receive new values for the onboard digital filter
Reset	restart with parameters in EEPROM

Table l-These C++ functions, which are *easily* integrated into PC applications, give the user control over weighing operations. User-written functions *in the language of choice can achieve the same results.*

Parameters	process setpoints are compared to weight and determine the state of digital outputs
Calibration	variables which calculate weight in engineering units from ADC results
Mode	the indicator module's operating mode, defining the logical relationship between the digital inputs, digital outputs, process setpoints, and measured weight
Weight	measurement in engineering units, used for comparison with setpoints
Data	raw ADC results
IOStates	a byte representing the on or off state of the indicator module's four digital inputs and four digital outputs
Filter	the poles of the onboard digital filter applied to the ADC result
Reset	the PC's last resort when communications to the indicator module is lost "Jump-to-zero" restart with variables loaded from EEPROM

Table 2: The control variables and procedures implemented in the load-cell controller's **onboard** microcontroller are *initiated or altered under PC control. Depending on the application, control variables and procedures are operator or program controlled.*

SOURCES

AD771 0 series converters, AD771 2
Analog Devices
One Technology Way
Norwood, MA 02062-9 106
(6 17) 329.4700
Fax: (617) 326-8703

754P-LC3 *load-cell* controller
Scanning Devices
25 Adams St.
Burlington, MA 01803
(617) 272.5135
http://www.tiac.net/users/scanning/

IRS
425 Very Useful
426 Moderately Useful
279 Not Useful

Applied PCs

Fred Eady

Driving Multiaxis Stepper Motors

Fred ushers us up to motion-control heaven without generating a single line of G code. He's quick to show how easy it is to design and implement even the most demanding *motion-control application* with *off-the-shelf stepper* equipment.

Charles V of Spain needed a clock. Not just an ordinary clock (throne sitters don't do anything ordinary), but a planetary clock. It needed to chime time celestially in Spain and anywhere else in the Holy Roman Empire that Charley deemed necessary. The clockmaker of the hour was an Italian gentleman named Torriano.

This particular clock's design required Mr. Torriano to produce a multitude of gear wheels. Torriano, wanting to complete this job in a **timely** manner, **successfully** adapted a lathe into what we today recognize as a multiaxis milling machine. The year was 1540 and Torriano, with a little help from his Spanish friends, gave birth to primitive motion control.

By the way, the completed clock contained in excess of 1800 gear wheels. Torriano's mill turned out **a** whopping three gear wheels a day. He punched out of the Charles V clock project a little over three years later.

A couple hundred years later, not to be outdone, John Wilkinson introduced the first "modern" machine tool, a horizontal boring machine. Henry **Maudslay followed** by adding an engine to a lathe in the mid-1790s. By 1830, Joseph Whitworth produced measuring devices accurate to one millionth of an inch. Numerical control machining using modern motioncontrol techniques was just over the horizon.

It's too bad these mechanical wizards didn't have access to our embedded-PC technology. Ponder this: Eli Whitney contracted **to** deliver 10,000 interchangeable part muskets to the Feds in 1798. With an embedded PC and associated motion-control components, Eli might have been able to pull off proposing and delivering M-I 6 assault weapons instead of muskets. Hmm...?

Can you imagine the effort that went into developing those early mills? They didn't have the compact form factors or the sophistication that our current embedded PC environment provides. "Off-the-shelf" was not in their vocabulary, and **software**...HA! Hardwire it, machine it manually, or forget it!

As it turned out, weapons making did just fine without computers. The first real computers-numericalcontrol **machines**—came about at the end of World War II. I have it on good word that they were developed for the Pentagon by Heald Machine with some help from good old MIT. By 1950, the CNC machine and modern motion control had arrived [1].

G WHIZ...

Since that first "C" in CNC stands for computer, one would think there must be a programming language or operating system involved somewhere.

Well, sorta.... Although technically not considered a programming language, G code is how most standard CNC control-

APC

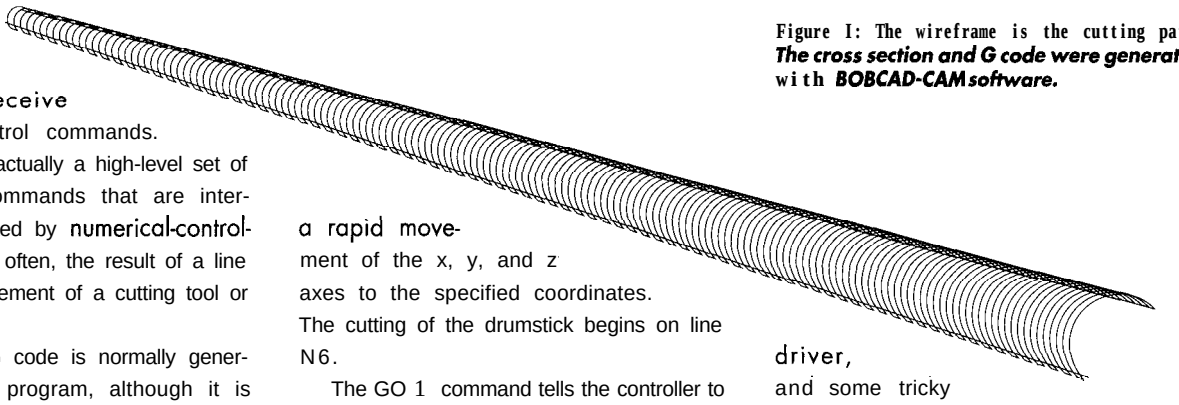


Figure 1: The wireframe is the cutting path. The cross section and G code were generated with BOBCAD-CAM software.

lers receive motion-control commands.

G code is actually a high-level set of EIA-compliant commands that are interpreted and executed by numerical-controller firmware. Most often, the result of a line of G code is movement of a cutting tool or mill surface.

These days, G code is normally generated by a CAD program, although it is possible (but cumbersome) to generate it manually. Instead of describing the entire set of EIA G codes here, let's look at some G code I generated to mill a drumstick.

ENGINEERING A DRUMSTICK

Of course, if I were inventing the drumstick, I would predetermine its initial dimensions such as length, diameter, and contour. But you all know I didn't invent it. So, I'll choose the chicken way out-reverse engineering. I picked up a standard stick and threw calipers at it.

As I took measurements along the drumstick shaft, I loaded the numerical data into my CAD program and produced the cross section you see in Figure 1. I instructed the CAD program to generate the G code necessary to machine this particular view of the drumstick.

The wireframe you see in the drumstick cross section is the path the cutting tool takes. The entire cutting program is in excess of 1600 lines of G code. However, **my editors** would frown on a listing as thick as the Chicago-area phone book, so you gotta get the idea from the G-code snippet in listing 1. OK?

The G code I generated mills only the cross-sectional area visible in the drawing. I personally have never used (or seen) a semisquare drumstick. So, to complete the milling process, the drumstick stock is rotated 180" and the same G-code milling process repeats.

The drumstick milling begins at line N1. line NO is a userdefined part number. The MO3 is a miscellaneous function that tells the controller to start the spindle in a clockwise direction. (The spindle holds the cutting tool.)

The next line of G code is a tool function that sets the tool number, its length, and its radius. The G40 command centers the tool in the programmed path. Line N5 performs a rapid movement of the x, y, and z axes to the specified coordinates. The cutting of the drumstick begins on line N6.

The GO 1 command tells the controller to move all specified axes simultaneously so that they all arrive at their specified locations at the same time. This command results in a straight linear motion known as *linear interpolation.* As you can see, this process is repeated some 3200 more times with the end product being this nice round drumstick.

Granted, this isn'taveryefficientwayto produce drumsticks, but you get the idea. You now have a feel for G code and how it is generated and used.

AT THE CROSSROADS

OK.. .Where have we been?

So far, we took the bus back to the 16th century because Charley V didn't know the time of day. We then traveled to the 18th and 19th centuries to observe the effects of motion control on the beginning of industrial automation.

Then, we wondered if 18th-century production of the automatic rifle would have had any effect on current world affairs. Finally, we demonstrated the use of G code as it relates to motion control.

I saw your lips move. You asked, "Fred, where are we going with this?"

My friend, we are going to a place that Torriano and Eli could only dream about. You could even call it "motion control heaven."

When we get there, we'll move motors without generating a single line of G code. Using off-the-shelf embedded-PC hardware from Octagon, a General Controls motor driver, and some tricky Ability Systems software, we will assemble a compact, highly functional motioncontrol system.

MOTION-CONTROL SUBSYSTEMS

Stepper motors are **easily obtained** and relatively inexpensive compared to other motor types. Therefore, my motioncontrol system is stepper-motor-based.

A stepper-based motioncontrol system consists of a motor or motors, a translator, indexer, low-voltage power supply, and limit switches. To traverse an axis, the stepper motor rotates in a positive or negative series of discrete steps. In most cases, the stepper motor moves an object by either driving a leadscrew or rotating a pulley.

Translator electronics excite the stepper windings to initiate theta motion of the stepper-motor shaft. The low-level translator electronics receive step and direction commands from an indexer. In addition to normal translator duties, the translator I selected controls the power to the stepper windings.

As you know, stepper motors consume greater amounts of current at rest than in motion. Under command of the indexer software, the translator reduces the motor-current consumption during rest or completely removes power from the motor windings. Since the indexer is under user-program control, the user tailors the overall motion-control system's current consumption.

Listing 1—Just 7 of 1600 lines of G code used to mill 50% of the drumstick.

```
NO P23467
N1 MO3
N2 GO6 T10 L3.323 R .250
N3 T10 S3500
N4 G40
N5 G00 XO.012 YO. LO.002
N6 G01 X-O.006 YO. 20.107
```

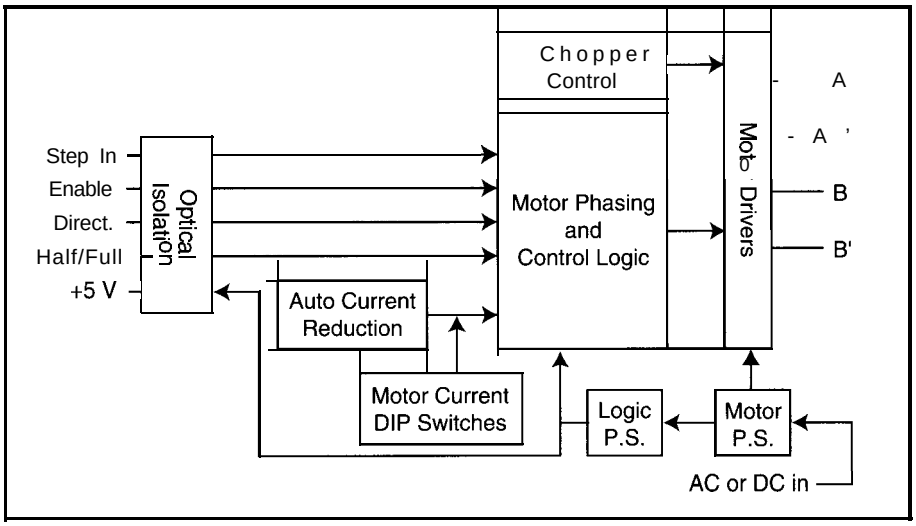


Figure 2: A block diagram of the ***29ALPT*** shows the optical isolation of the inputs.

The indexer is a high-level controller taking commands from a user-written program. In relation to the translator, the indexer issues step pulses and direction information that the translator electronics use to control the stepper motors.

In thisapplication, the low-voltage power supply is two independent power sources. The translator's onboard power circuitry feeds the translator electronics as well as the stepper-motor windings. A separate switching power supply powers the indexer module.

Limit switches normally indicate the extreme limits or boundaries of axial motion for a motion-control system. The selected indexer allows a user-written program to sense a pair of mechanical or optical switches. Indexer LPT automatically senses and acts on the correct limit switch.

The translator of choice for this project is the General Controls Dragon Driver **Stepper Motor** Driver. For the indexer hardware, I selected the Octagon Systems 40 10 PC Microcontroller Development System. Indexer LPT is the magical software that transforms the Octagon 4010 into an indexer.

The 29ALPT is a 40-V, 3-A per phase, dual-axis driver which connects to any standard PC printer port. Working with the indexer hardware and software, the 29ALPT eliminates the need for two indexers and two translators for dual-axis motion-control systems.

An advantage of the 29ALPT in this application is that it is designed specifically to work with the Indexer LPT software package. Figure 2 is a block diagram of the Dragon Driver 29ALPT. The actual Dragon Driver 29ALPT is shown in Photo 1.

THE OCTAGON 4010

The Octagon 4010 PC microcontroller is an ISA '386CX-based single-board PC. This little 25-MHz board is chock-full of

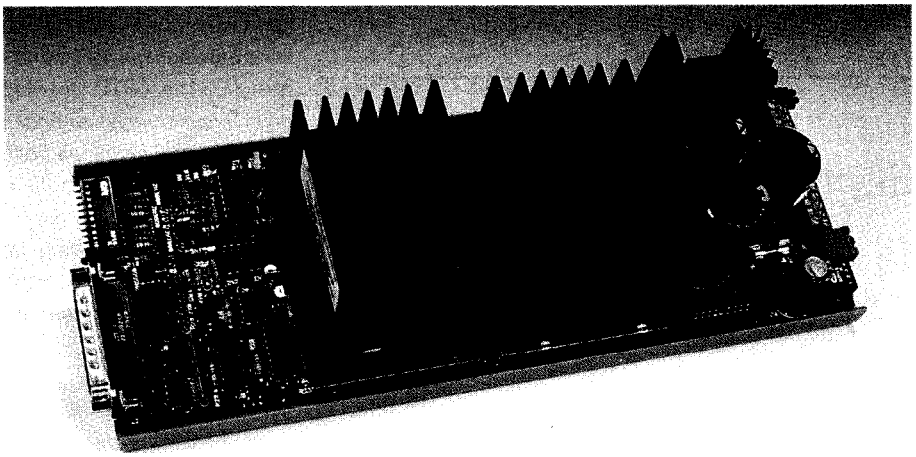


Photo *1:* The **Dragon Driver *29ALPT*** is everything you need for stepper-motor control. This compact electronic package is rugged and easy to use.

APC

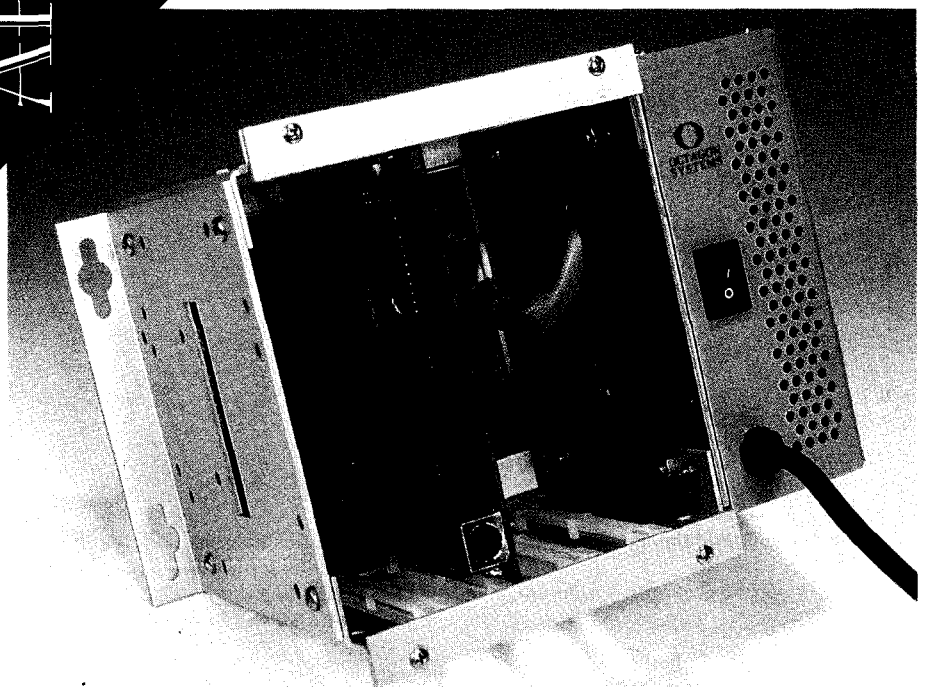


*Photo 2: **Not** only does the 40 **10** look good, it is solid enough **to** be air dropped!*

goodies. I chose the 4010 system for this application for several reasons.

Like the Dragon Driver, the 4010 is rugged. It withstands extreme temperatures and abusive environments. It comes ready-to-roll right out of the box.

The 4010 is housed in a four-slot card cage with an integral power supply. All four slots connect via an ISA passive backplane. In this configuration, the Octagon 4010 is a perfect fit for this motion-control application because I can "hang" or "box" it anywhere.

The two solid-state disks onboard the 4010 resemble drives. SSDO houses the Datalight ROM-DOS V. 6.0—which acts as a sort of "instant" DOS in ROM along with the BIOS. SSD1, which can be up to 1 MB of EPROM or 5 12 KB of flash, is dedicated to application storage. The standard is 2 MB of DRAM. The ROM-DOS feature and abundance of DRAM and flash let me run my motion-control indexer software with a minimum of hassle. Photo 2 shows the 4010 card.

This motion-control application doesn't require all the resources the 4010 offers. Basically, I'll use the DRAM, DOS in ROM, the solid-state disk, COM1, and LPT1 These hardware resources are all that's needed to implement the indexer function via printer port LPT1 . Indexer LPT doesn't need any special motion-control adapters.

THE SOFTWARE

Indexer LPT uses standard printer-port electronics to emulate a multiaxis stepper-motor driver. Used with the Dragon Driver 29ALPT and the 4010, Indexer LPT produces a powerful, compact, and inexpensive motion-control system.

Loading as a device driver, Indexer LPT uses plain-English ASCII text instead of G code to effect motor control. A TSR included with the Indexer LPT package provides additional device-driver routines.

The Indexer LPT device driver emulates a data file and is addressed with a file handle to a DOS installable device named "motor." This clever technique allows bidirectional communication via the device driver with the user program.

Since motor accepts ASCII characters, high-level languages such as C, BASIC, and Pascal can submit plain-English files or commands to motor. This scheme allows command-laden batch files to be transferred to motor via the DOS copy command.

Ill-level outputs for step, direction, reduced current, and all windings off are available on specified pins of the 4010 parallel port LPT1 . The 4010 has a single PC-compatible printer port located at address 378 hex, which correlates to axes c and din Table 1.

The Indexer LPT designers realize that each stepper-motor application is unique. To compensate for various amounts of friction, stiction, and inertia, Indexer LPT lets users define acceleration and velocity by placing values in the acceleration and speed registers within the Indexer LPT software module.

You can therefore define ramp-up acceleration, constant velocity, and ramp-down deceleration. The values for each phase of motion are stored in the Indexer LPT low-speed, acceleration, and high-speed registers.

Most stepper-motor applications drive mechanics that move an object along an axis. The user has to keep up with the position of that object relative to a predetermined home coordinate.

Indexer LPT enables the user to set a relative home position. Indexer LPT auto matically tracks and tallies positive and negative motor steps. It tracks position 2,147,483,647 steps in either direction.

Card Base Addr. (hex)	278		378		3BC	
Axis	a	b	c	d	e	f
Reference Group	x	y	x	y	x	Y
Function						
Step	2	6	2	6	2	6
Direction	3	8	4	7	3	7
Reduced Current	4	9	5	8	4	8
All Windings Off	5			9	5	9
Low (-) Limit Switch	1	16	1	16	1	16
High (+) Limit Switch	14	17	14	17	14	17
Auxiliary Input	13	12	13	12	13	12
Signal Ground	18-25		18-25		18-25	
	Pin Numbers					

Table 1: The Octagon 40 I O's base address is 378 hex. Our motion-control ***system*** uses axes c and d.

PUTTING IT ALL TOGETHER

First, you need to establish communications between the 4010 and a host PC. The 4010 Development System comes equipped with a communications package called PC Smartlink. SmartLink turns the host PC into a terminal allowing the user bidirectional access to the 40 10 using Xmodem protocol.

No problems here.. ..I connected the 4010 serial cable to my host PC via a null modem and started PC SmartLink. Following the SmartLink ban-

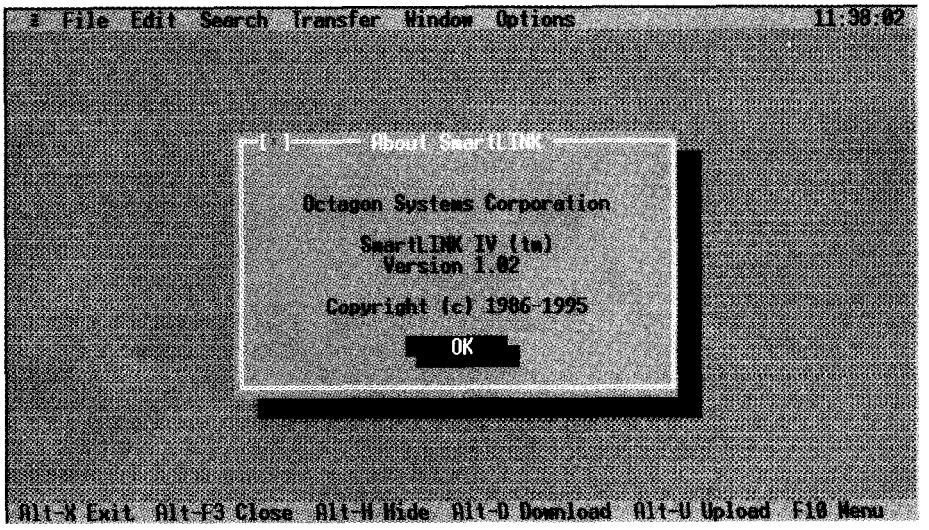


Photo 3: A welcome ***(yet*** expected) ***sight!*** The Octagon 4010 hardware and ***the SmartLink IV software*** establish a ***communications*** session.

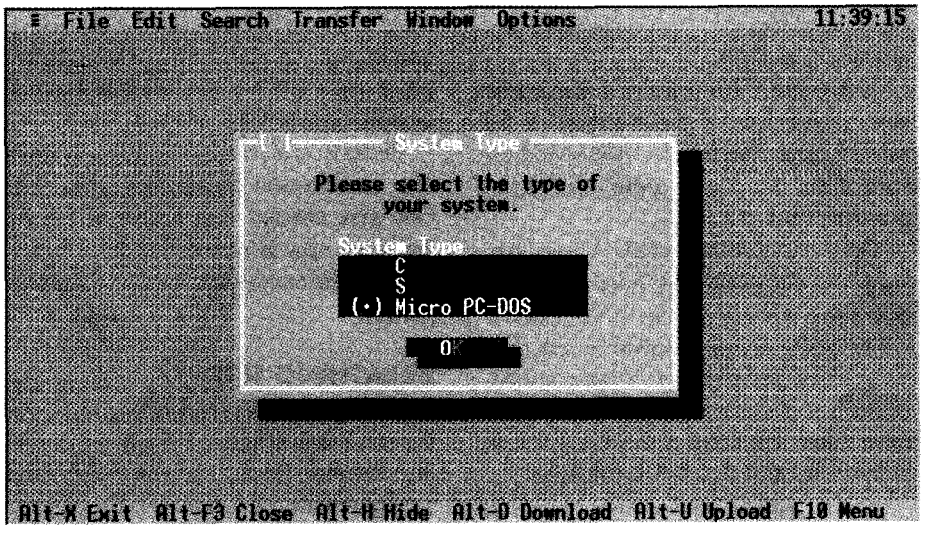


Photo 4: ***Although*** Jndexer ***LPT*** requires Micro PC-DOS, note ***the*** ability to use two versions of good ***old*** BASJC too. ***The*** 40 ***10*** is designed to dance to any programmer's ***tune.***

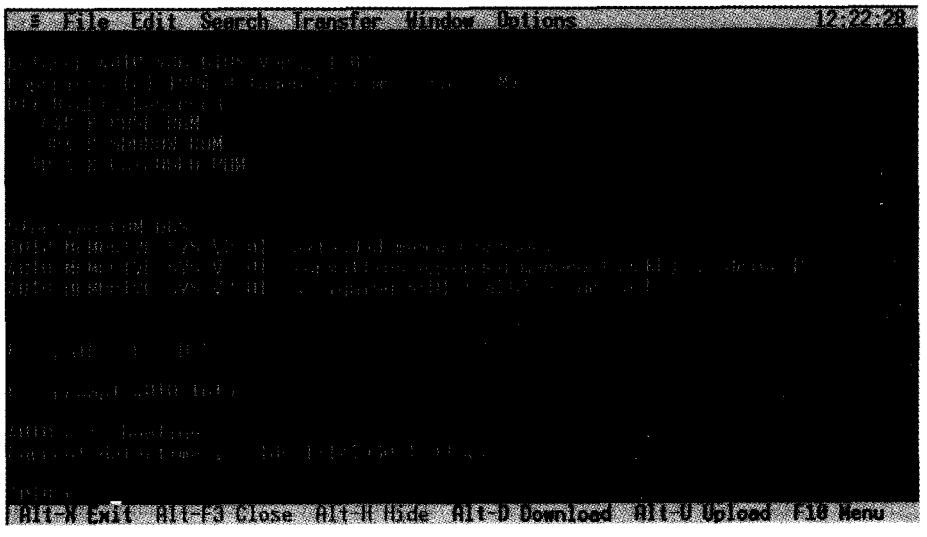


Photo 5: A ***C:\>*** prompt, 640 KB of base memory, and ***1*** MB of extended RAM. ***All*** this within the same space consumed by a desktop's power ***supply!***

APC

ner screen, I specified the system type as DOS (Photos 3 and 4) and BOOM! The 4010 booted immediately. Photo 5 shows its banner.

I next loaded the Indexer LPT device driver and TSR onto the 4010 boot drive. To do this, I must build a new CONFIG.SYS file which includes the Indexer LPT device driver, I X D EV . SY S. Since I need access to all of the other solid-state disks, I include the MEMDRIVE.SYS driver in the new CONFIG.SYS, too.

As you see in Photo 5, I am currently booting from 40 10 drive C, which is SSDO. Drive C is write-protected to prevent "accidents" that may occur during file transfers between the host PC and the 4010. Thus, the target boot device will ultimately be SSD1, drive E. Drive D holds the Indexer LPTTSR, IXTSR. EXE.

Now I'll store the new CONFIG.SYS, ROM in DOS COMMAND.COM, MEMDRIVE.SYS, and IXDEV.SYS into their own directory on the host PC. You can copy the COMMAND.COM and MEMDRIVE.SYS files from the driver disk included with the 4010 Development System. Or, you can upload the files from the 4010 drives.

If **you** upload them, be careful. Trust me, CON FI G. SY S on your host PC can *accidentally* become CONFIG.SYS on the 4010 and vice versa. Don't ask me how I know that. I really don't want to talk about it.

Now that the files reside in a directory of their own, I can transfer them to the 4010 E drive using SmartLink transfer utilities.

At the 40 10 end, this transfer is done by executing the 4010 Disksave procedure.

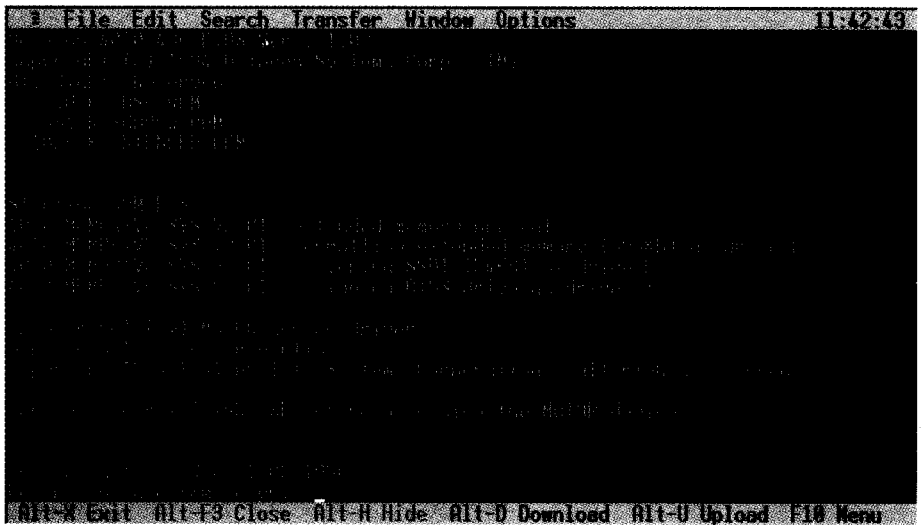


Photo 6: With our new system generation, virtual **E** drive becomes our boot drive. Note that the BIOS drive is now designated **as** virtual **F.**

With the SmartLink Disksend program, Disksave places files from a specified host directory into flash.

In this case, the directory containing the new system files is downloaded to the 4010 E drive. The transfer utilities and on-card 4010 hardware erase and reprogram the flash.

Once the transfer process is complete, the 4010 setup utility specifies E as the new boot drive and restarts the 4010. Photo 6 is the newly created system image. The Indexer LPT device driver is now loaded. Photo 7 is a directory listing of the new boot drive. Note that I also give you a glimpse ofthenewCONFIG.SYS inthisshot.

So far, I generated a new operating environment and successfully loaded the Indexer LPT device driver. The next step is to load the Indexer LPT TSR.

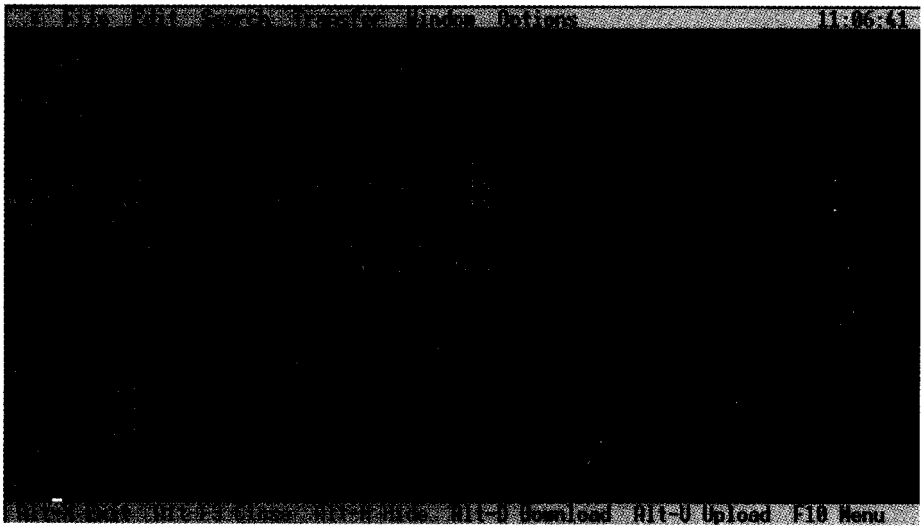


Photo 7: Everything necessary to run Indexer LPT is here. Including memdrive device driver directives in the 4010 *CONFIG.SYS allows us* **to** *access all* of the 4010 goodies.

I chose to load the TSR to drive D. You could load the TSR on the new boot drive and include an AUTOEXEC.BAT to kick it off. If the motion-control system is unattended, use the latter method.

Photo 8 is a shot of the 4010 D drive after a successful file transfer. The . CMD files were created later on the 4010. All that's left now is connecting the Dragon Driver to the motors and the 4010 parallel port.

BREATHING FIRE

The General Controls Dragon Driver accepts many different stepper-motor wiring schemes. Four-, six-, and eight-lead configurations are supported. The connection depends on your application and motor.

The Dragon Driver documentation gives detailed instructions on attaching almost any two- or four-phase stepper motor. My kitcamewith motorsandeverything needed to hook up both motors and interface to the 4010 parallel port.

You need an Octagon VTC-5/IBM converter cable to interface the 4010 parallel port to the Dragon Driver 25-pin logic-input connector, but there's nothing to it! I connected the stepper motors according to the stepper lead-wire diagrams included with the Dragon Driver board. The motors I used were eight-lead types connected as eight-lead parallel.

Next, I connected the 26-VAC, 6-A transformer. Piece of cake. The line cord and secondary cable were equipped with ¼" quick-connect terminals. The only way to mess this up is to do it in the dark!

Photo 8: The Indexer LPT executable resides on virtual drive **D** along with the *. CMD* application filer. The DOS *TYPE* command reveals the contents of *TST1 . CMD*.

After finding my motor's current rating in the specifications chart, I set the 10-position DIP switch according to tables found in the manual. Half-step mode is recommended, and that's how I set it up.

Just for grins, I also configured for reduced motor current at rest. I didn't connect any limit switches, but you may need to if your application requires them. The last step involves interfacing the 40 10, now an indexer, to the Dragon Driver via the VTC-5 and printercable combination.

Everything powered up just fine.. .(Yea!) Time to turn some shafts. If I could **show** you the shafts turning via the printed page, I would not be on Earth. So, I captured a short series of step pulses with a logic analyzer. The uppermost trace is step. The trace directly below it is direction. This pulsetrainwasgenerated by the TST1. CMD file listed in Photo 8. You see the step pulse series in Figure 3.

The program sets a home position for the c axis and then rapidly moves the stepper 3500 steps positive, 1000 steps negative, 800 steps positive, and returns home. I kicked this little program off with a DOS copy command to motor (copy TST1. CMD motor).

NO PAIN-MUCH GAIN

The real beauty of this motion-control system is its modularity. If your application specifications call for a motor the 29ALPT doesn't support, use the Dragon Driver quick-selection guide and find the driver that supports that particular motor. It's that easy. Indexer LPT can be integrated with a variety **of** step-motor drivers including some servo amplifiers from numerous manufacturers. This is motion control's version of plug and play....

As for software modularity, at rest, Indexer LPT is virtually invisible. The user stores homegrown and commercial motioncontrol programs on the same hardware and schedules them at will.

Embedded PCs combined with off-the-shelf stepper equipment like the Dragon Driver take the pain out of designing and implementing even the most demanding motioncontrol application.

Think of it this way. In a matter of minutes, I tied two pieces of equipment together, loaded some software, and drove a couple of steppers! No heavy programming and no extended hardware build cycle. APC.EPC

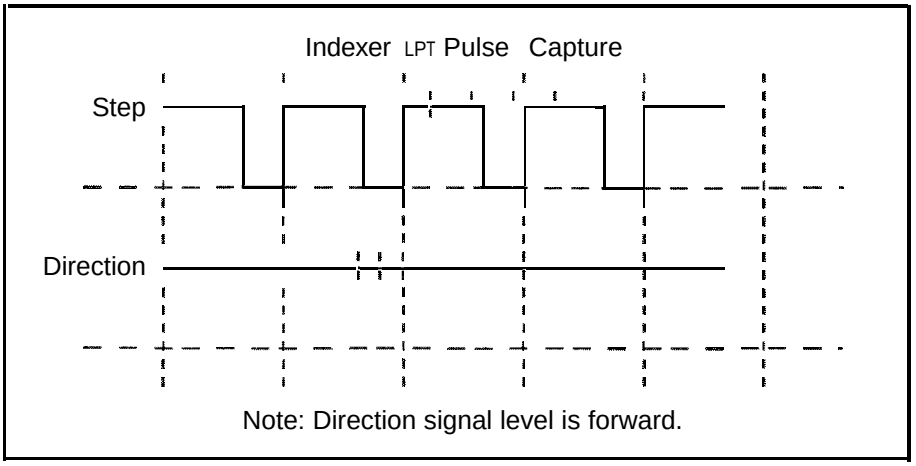


Figure 3: The top trace represents the pulses that fire the DRAGON. The direction ***logic level*** is directly below. This timing diagram is the ***result of*** executing *TST1 . CMD*.

Fred Eady has *over 19* years experience *as a* systems engineer. He has worked with computers and communications systems large and small, simple and complex. His forte is embedded systems design and communications. Fred may be reached at *edtp@ddi.digital.* net.

REFERENCE

[1] **Wayne** Hohler, "Wahoo's CNC Machining **Page,**" http://www.neca.com/~wahoo/cnchist.html

SOURCES

Indexer **LPT**
Ability Systems
1422 Arnold Ave.
Roslyn, PA 19001
(215) 657-4338
Fax: **(2 15) 657-78 15**

BOBCAD-CAM
1001 S. Myrtle Ave., Ste. 1
Clearwater, FL 34616
(813) 441-9793
Fax: (8 13) 44 I-8947

ROM-DO5 V. 6.0
Datalight
188 10 59th Ave. NE
Arlington, WA 98223
(360) 435-8086
Fax: (360) 435-0253

Dragon Driver Stepper Motor Driver
General Controls
2350 Brickvale Dr.
Elk Grove Village, IL 60007
(708) 595-2 152
Fax: (708) 595-2271

4010 PC Microcontroller Development System, VTC-5/IBM converter cable
Octagon Systems
6510 W. 91st Ave.
Westminster, CO 80030
(303) 430-I 500
Fax: (303) 426-8 126

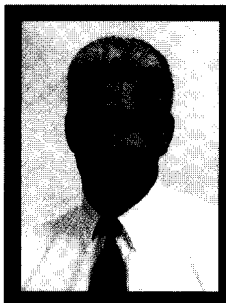
IRS

428 Very Useful
429 Moderately Useful
430 Not Useful

The Little LAN That Could

SILICON UPDATE

Tom Cantrell



Is Tom's secret decoder ring a sign he's moving into the spy business? Not likely once he's let the cat out of the bag (or the time out of the can) about Dallas Semiconductor's line of touch products.

Given the hoopla—not to mention IPOs and quick bucks-surrounding the Internet, I imagine it won't be long before someone starts touting an embedded version.

Instead of arcane wiring schemes and complicated commands, distributed system control will be E-mail based. Punching a switch will transmit something like "Yo, CPU! Can I get some service?" Perhaps it will respond with a cheery "Hold yer horses, bub." Debugging a dysfunctional system may become something akin to running an online group-therapy session.

Of course, there's no need to perform mundane tasks associated with a front-panel display. Just spawn the appropriate Java applets and maybe an agent or two to do your bidding.

What the heck, why not just give every switch, relay, display, and chip its own web page? Of course, making room for all the little telephone poles inside your box is worrisome, as will be your phone bill.

The funny thing is, I'm only half joking. In fact, communication on the embedded front continues to rocket forward. There's plenty of data to move and plenty of ways to move it.

Indeed, with new buses and networks announced practically daily, it looks like embedded communications is where the action is. Why don't we let Dallas Semiconductor get their two cents ('er, wires) in.

LAN-IN-A-CAN?

Back in the good old days when 40 pins was as big as a chip got, I and my Silicon Valley colleagues would oft bemoan the fact that we couldn't cram in a hundred pins' worth of features. This chat generally devolved into a mux versus no-mux flame fest that would invariably end with some wise guy's sarcastic suggestion, "Why don't we multiplex power and ground?"

Well, Dallas couldn't quite manage that trick, but they did the next best thing. As shown in Figure 1, the appropriately named *MicroLAN* packs power and data in just two wires.

Before exploring this rather intriguing concept further, let's look at the

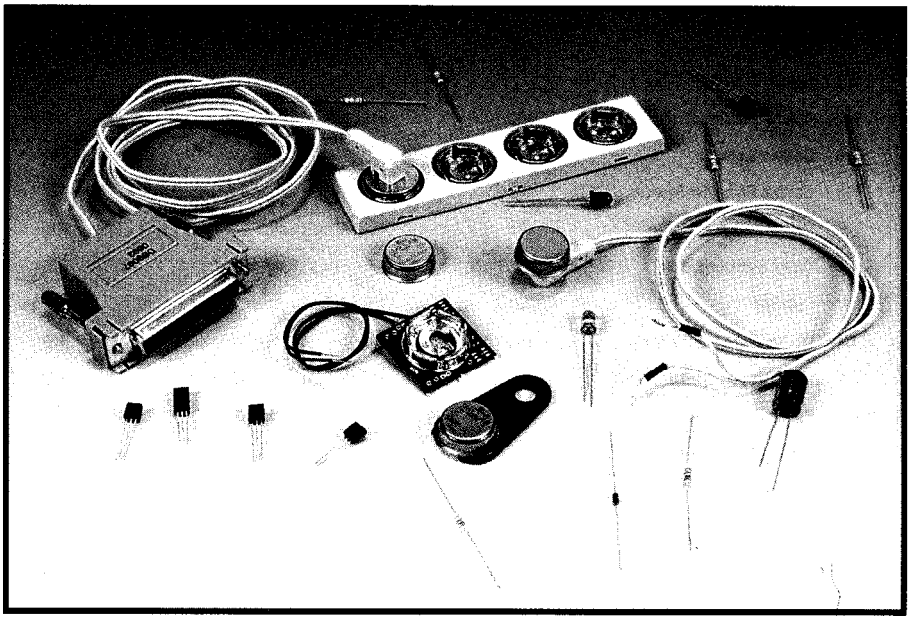


Photo 1—*The DS9091K MicroLAN Evaluation Kit ($49)* includes a *variety of touch and solder-mount devices and accessories.*

Photo 2—*Talk about keeping your data close at hand! The* **decoder** *ring actually has a touch memory device* **mounted on if. Touching** *the* **ring** *to* **any reader instantly identifies** *the* **wearer.**

selection of interesting MicroLAN plug-ins that Dallas offers.

For reference, a couple of the most novel devices have been covered in Jeff's columns, including "Temperature Sensing Eludes Analog Interfacing" (INK 42) and "Time In A Can" (*INK* 58). In fact, you can trace the evolution of the MicroLAN from the original three-wire version shown in the first article through the one-wire scheme later on.

To update the situation, Table 1 shows that the MicroLAN product line consists of touch memories (in the familiar metal can), solder-mount products (similar functions in IC form), software (development tools and evaluation kits), and accessories (connectors and interfaces).

Touch memories of various sorts, including NVRAM (i.e., internal battery), EEPROM, and Add-Only (EPROM), are widely used in access control, security, activity-logging, and inventory-control applications. The main claim to fame is the robustness of the "can" technology compared to finicky bar codes and expensive and fragile smart cards.

The touch memories are supported with an impressive infrastructure of connectors, carriers, interfaces, and so forth from Dallas and a variety of third parties (see Photo 1). For instance, the familiar employee badge can be had with an embossed-in touch memory, or you can even have your own decoder ring (Photo 2).

The addition of time and temperature functions (see Jeff's articles) and solder-mount products definitely broaden the reach of the product line beyond traditional ID, inventory, and security applications. MicroLAN now seems poised to compete in the "LAN-In-A-Box" arena currently dominated by I^2C, SPI, and Microwire, among others.

An easy way to check out MicroLAN is to take advantage of one of the many evaluation kits Dallas offers, such as the $49 Minimalist Temperature Control board (see Figure 2 and Photo 3).

It includes a DS1820 digital temperature sensor and single- and dual-bit I/O expanders (the DS2407 and 2407P, respectively). The latter drives red and green LEDs in a heating and cooling simulation, while the single-bit unit connects a piezo film sensor (see *INK 22*, "Kynar To The Rescue") as a touch switch input.

The DS1920 ($4.86 in 1000s) and solder-mount equivalent '1820 ($3.59 in 1000s) are MicroLAN-compatible upgrades of the DS 1620 discussed fully in Jeff's earlier article.

To rehash, as shown in Figure 3, they combine a temperature sensor with a 9-bit ADC and cover a wide temperature range (-55" to 100°C for the '1920, -55" to 125°C for the '1820) with good accuracy (to 0.5°C).

The DS2407 ($2.03 in 1000s) is an interesting part that combines one or two digital I/O lines with 1 Kb of OTP EPROM. The paraphrase "Give me a bit, and I'll move the world" describes the interface possibilities, especially given the chip's hefty 13-V, 50-mA I/O capability. If you don't need the EPROM, the DS2405 offers a simple l-bit TTL-like I/O function alone.

The MicroLAN evaluation kits rely on the DS9097, which is a small, passive PC-COM-port-to-MicroLAN adapter (see Figure 5). Most PCs (at least desktop models) provide enough current via their RS-232 driver to power the range of experiments encompassed by the EV kits.

The PC connection is reinforced by the fact that the demo (see Photo 4) and development software all run on PCs, with the newest stuff on Windows. The TMEX (Touch Memory

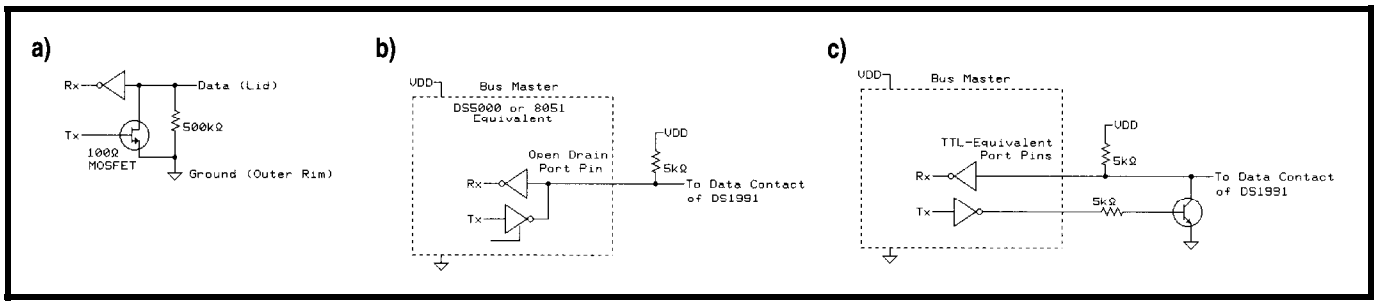


Figure 1—*MicroLAN requires* **a** *single* **bidirectional** *open* **collector** *I/O* **line. Alternatively, unidirectional input and output lines paired with a transistor achieve** *the* **same** *effect.* **The** *pull-up resistor provides power to the bus.*

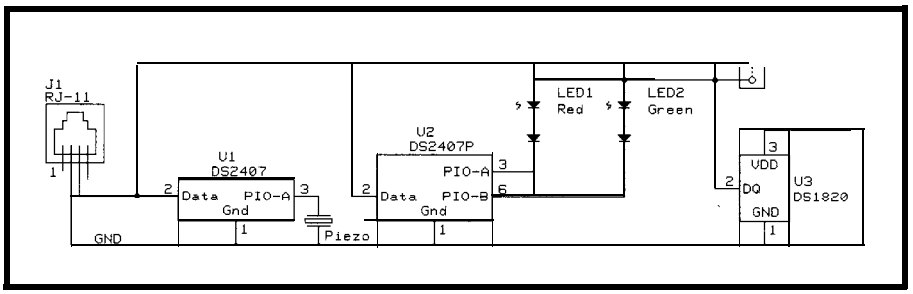


Figure 2—*The "Minimalist* **Temperature Controller" evaluation** *kit certainly lives up to ifs name.*

Executive) toolset ($495) provides language-independent API, DLL, and installable interrupt support for DOS and Windows, utilities, and examples.

Indeed, most of the high-level functionality that turns the formerly mild-mannered interface into a LAN is the responsibility of the PC hardware and software. It's certainly possible to use something other than a PC as a network master, and Dallas gives lots of examples and clues to get you started. Nevertheless, those using a PC, whether desktop or embedded, as the host get a head start.

Touch Mennories	
DS1920	Touch Thermometer
DS1971	256-bit EEPROM Touch Memory
DS1981U	UniqueWare Touch Memory
DS1982U	UniqueWare Touch Memory
DS1982	1-Kb Add-Only Touch Memory
DS1985	16-Kb Add-Only Touch Memory
DS1986	64-Kb Add-Only Touch Memory
DS1990A	Touch Serial Number
D51991	Touch MultiKey
DS1992	1-Kb Touch Memory
DS1993	4-Kb Touch Memory
DSI994	4-Kb Plus Time Touch Memory
DS1995	16-Kb Touch Memory
DSI996	64-Kb Touch Memory
Solder-mount Products	
DS1820	One-wire Digital Thermometer
DS2223	EconoRAM
DS2224	EconoRAM
DS2401	Silicon Serial Number
DS2404S-CO1	Dual-Port Memory Plus Time
DS2405	Addressable Switch
DS2407	Dual-Addressable Switch Plus 1-Kb Memory
DS2430A	256-bit One-wire EEPRCM
DS2501-UNW	UniqueWare Add-Only Memory
DS2502-UNW	UniqueWare Add-Only Memory
DS2502	1-Kb Add-Only Memory
DS2505	16-Kb Add-Only Memory
DS2506	64-Kb Add-Only Memory
Software	
DS0621-SDK	TMEX Professional Software Developer's Kit
DSO63Ox	TMEX Performance Modules
DS9091K	One-wire MicroLAN Evaluation Kit
DS9092K	Touch Memory Starter Kit
DS9103K	Touch Memory Access Control Demo Kit
Accessories	
DS1401	Front Panel-Button Holder
DS1402	Button Cable
DS9092	Touch Memory Probe
DS9092R	Touch Port
DS9093x	Touch Memory Mount Products
DS9094	MicroCan Clip
DS9096P	Touch Memory Adhesive Pads
DS9097	Touch COM Pot-I Adapter
DS9097E	Touch COM Pot-I Adapter
DS9098	MicroCan Retainer
DS9100	Touch and Hold Probe Stampings
DS9101	Multipurpose Clip

Table 1—*MicroLAN add-ons include time, temperature, bit I/O, and a wide variety of memories including NVRAM, EPROM, and OTP EPROM.*

ONE WIRE SHORT OF A FULL LAN

Taking a closer look at the MicroLAN protocol reveals that clever behind-the-chips engineering and software stretch bang per bit to the max.

A MicroLAN network consists of a single master (i.e., the PC for the demo kits) and multiple slaves [the previously mentioned touch-memory and solder-mount IC products). The single data line serves as both the half-duplex conduit for bidirectional data transfer and as a power source for the slaves.

The bus (i.e., combined power and data line) is open collector with TTL levels (i.e., less than 0.8 V low and more than 2.2 V high). To provide power to the slaves, the bus is pulled up to 2.8-6 V at the master.

Getting the most out of the few milliamps likely to be available relies on a smorgasboard of power-saving techniques. Most basic is that all the memory technologies require no stand-by power (i.e., EPROM or EEPROM) or that the battery {NVRAM} is built in. In addition, power to run the Micro-LAN interface logic is derived from the data line itself, with a small internal capacitor bridging the gaps.

Nevertheless, don't expect miracles your RS-232 port can't deliver. For instance, the thermometers (DS1820 and 1920) draw more current during the actual temperature-conversion cycle while the EPROMs need a 12-V programming voltage. One hint: program the EPROMs a bit at a time to minimize power consumption.

Despite the limitations (for instance, the EV kit can't drive a network of more than one temperature chip), you can lash together a surprisingly powerful setup quickly and easily.

If you need more or different power (i.e., EPROM programming), Dallas offers a version (9097E) of the COM port adaptor that accepts a wall-mount DC input. Of course, any kind of serious industrial strength Micro-LAN setup is going to call for much more extensive interfaces with auxiliary power, isolation, and noise-immunity enhancements. Fortunately, the documentation covers a number of likely alternatives in some detail.

Addressing is a chronic dilemma for small LANs. If you're just talking about a chip or two on a PCB, dedicating a chip-select line is likely the easiest way to go.

Otherwise, especially if you're running wire, addressing has to be handled over the data link. Of course, then, each node has to know its own address, typically imply-

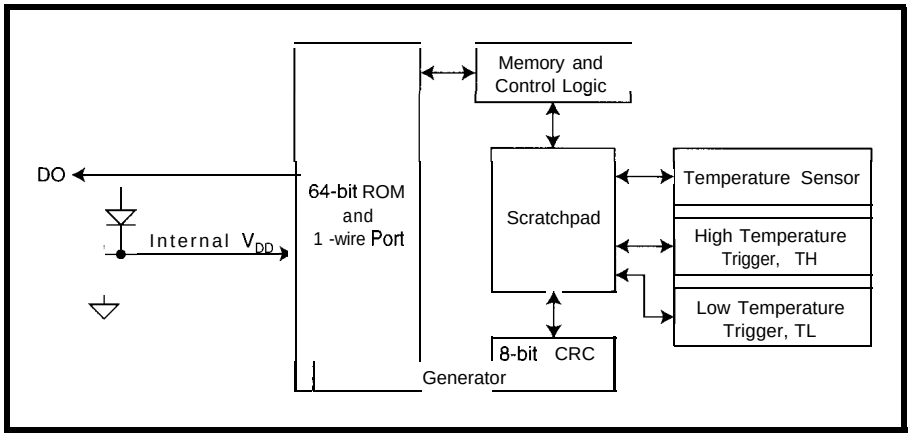


Figure 3—*The DS1820 (solder mount) and DS1920 (touch) temperature sensors include MicroLAN interface, lasered ID, CRC generator, and EEPROM high- and low-temperature alarm registers.*

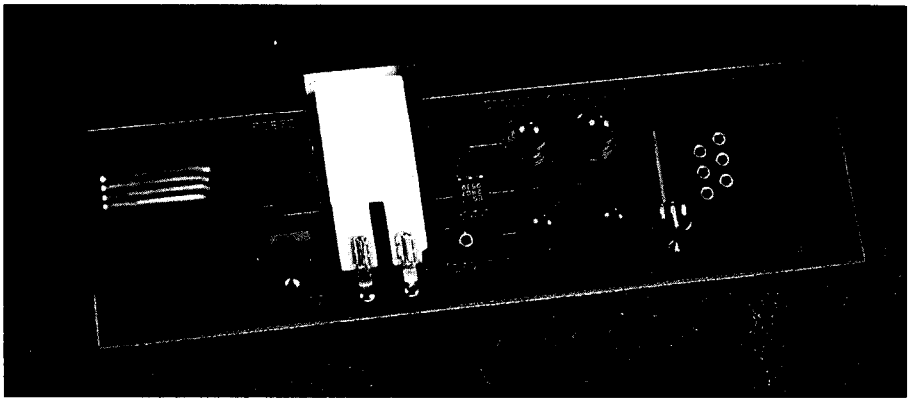


Photo 3—*The* **extra (unpopulated)** *RJ-11 connector on the Minimalist* **Temperature Control board** *allows daisy-chaining, but the DS9097 COM port interface can only power one DS1820.*

ing a unique **ROM** image or-horrors-a DIP switch.

Alternatively, intricate self-addressing schemes are the only option.

By contrast, addressing is natural for MicroLAN since, by definition, every compatible add-on includes a unique (laser-programmed at the factory) 48-bit address, supplemented by an 8-bit device type code and 8-bit CRC as shown in Figure 5. A brief pondering assures you that Dallas can sell a heck of a lot of chips and not run out of addresses anytime soon.

Indeed, the DS2401 Silicon Serial Number ($1.08 in 1000s) is nothing more than the eight bytes of lasered ID and a MicroLAN interface. Besides the obvious security and inventory applications, the DS2401 makes a good input switch. A bunch of them can be strung on a wire and, by arranging so the chip's ground line makes or breaks contact, the status of each switch can be determined by the presence or absence of its unique ID.

I'M **OK, UART OK**

There are a lot of embellishments, but a high-level transaction on the MicroLAN goes something like this. The master resets the network to get everyone's attention and then selects a slave for access by its address. The master then issues additional chip-specific commands and performs any required data transfers to or from the slave, which remains selected until the next reset.

Key to the one-wire LAN living up to its moniker is the fact that it relies on self-timing (i.e., each device has a timebase), rather than the usual serial clock line. The master controls the transfer of each bit of information by generating slots as shown in Figure 6.

For instance, to write a 0, the master drives the data line low and leaves it there for the duration of the 60-µs slot. To write a **1,** the master drives the line low, but then high within **15** µs. This shift guarantees that the MicroLAN slaves see the correct value within their specified (15–60-µs) sampling window.

A read-data slot is also initiated by the master driving the data line low and then immediately (1µs is recommended) releasing it **(to** float high via the pull-up at the master). At this point, the selected slave either leaves the data line floating **(1)** or drives it low (0) for sampling by the master.

There's also a reset and presence-detect sequence in which the master drives the data line low for 480 µs and

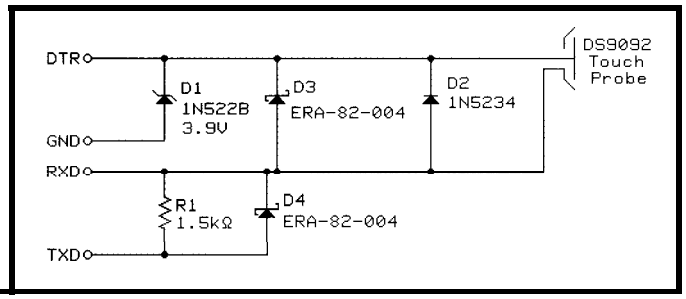


Figure *4-A simple COM port* **adapter,** *the 059097 has some functional limitations, but the price is right.*

then lets it float high for another 480 µs. During the latter half, the master looks for a presence pulse from any newly connected slaves. It's on this humble foundation, little more than read and write a bit, that the higher layers of the MicroLAN protocol rely.

Developing your own drivers isn't that hard, though the microsecond-scale timing likely dictates the use of assembly language. You see an example for the '5 1 in Listing 1.

The use of a PC COM port calls for a rather inspired hack in which Dallas engineers coerce the 8250 UART into generating the proper slot waveforms. A character transferred from the CPU is shifted out, with 8 bit (a start bit, 6-bit char(!), and stop bit) composing a single slot on the MicroLAN.

The UART runs at 115.2 kbps, which means the bit rate on the MicroLAN is one-eighth that, or 14.4 kbps.

Generating some of the trickier waveforms abuses the 8250 indeed. For instance, to generate the long reset slot, you temporarily set the UART to 8-N-l at 10,473 bps, before switching it back to 6-N-l at 115 kbps. Notice how the UART receives everything it sends. So, extracting received data involves sorting out bits driven by the slaves from the bits you just sent.

Keep in mind that there's a lot of overhead given the address traffic and host software machinations, especially for short messages such as time and temperature readings or bit I/O.

Though MicroLANs can be expanded nearly indefinitely using a switch-based tree topology, more likely setups (e.g., single branch) are ultimately limited by bus-loading concerns. MicroLAN is fine for time, temperature, ID, switches, relays, and such, but more grandiose plans must be tempered by bandwidth constraints.

The cabling, bandwidth, and host software limitations jointly conspire against MicroLAN (as against all serial buses) when it comes to the concept of interrupts, such as those generated by the time (DS 1994) and temperature (DS1920) chips.

Basically, there's nothing any MicroLAN device needing attention can do but yank the data line down repeatedly and hope for service. Presuming the master detects the activity, it must search the MicroLAN for the interrupting device and service it. Having found and serviced the first interrupt, the master must repeat the search-and-service rigmarole as long as the pleas for help continue.

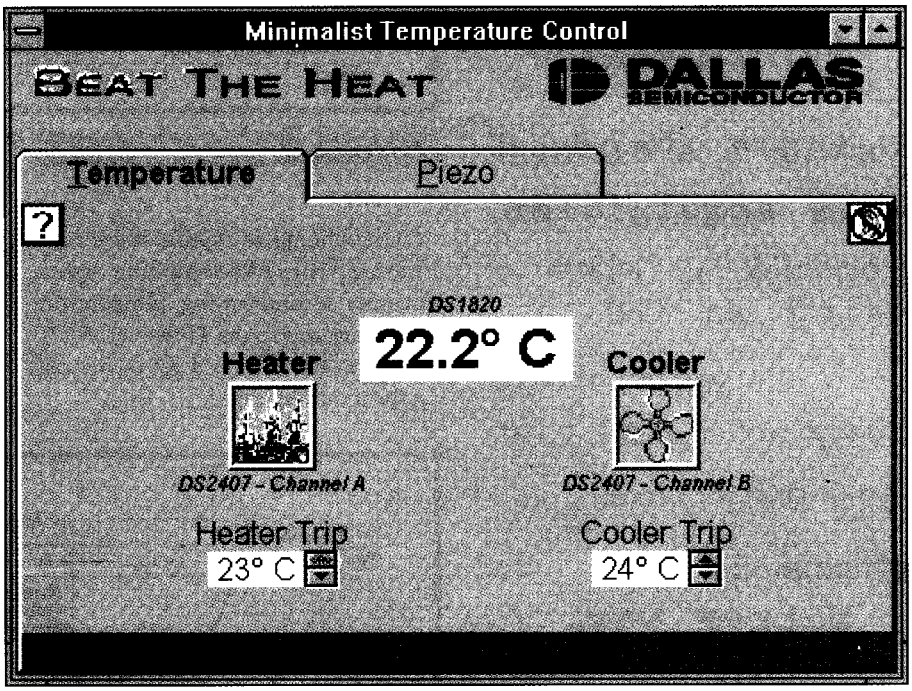


Photo 4—*Running under Windows, the Minimalist Temperature Control demo software simulates a thermostat (red LEDs for heating, green for cooling) with a switch (piezo film) input.*

8-bit CRC Code		48-bit Serial Number				8-bit Family Code (09h)	
MSB	LSB	MSB			LSB	MSB	LSB

Figure 5—*Every MicroLAN device includes a unique (laser-programmed at the factory) 48-bit ID, supplemented with an 8-bit type specifier, all protected by an &bit CRC.*

The scheme works fine for human-time activities like switching, lights, once-per-second timing, temperature sampling, and so on. However, it's clear that things could get rather ugly under too much load.

Consider the interrupt and service latency issues carefully and certainly don't rely on MicroLAN interrupts to attempt any of high-speed stuff you might try with a microprocessor interrupt.

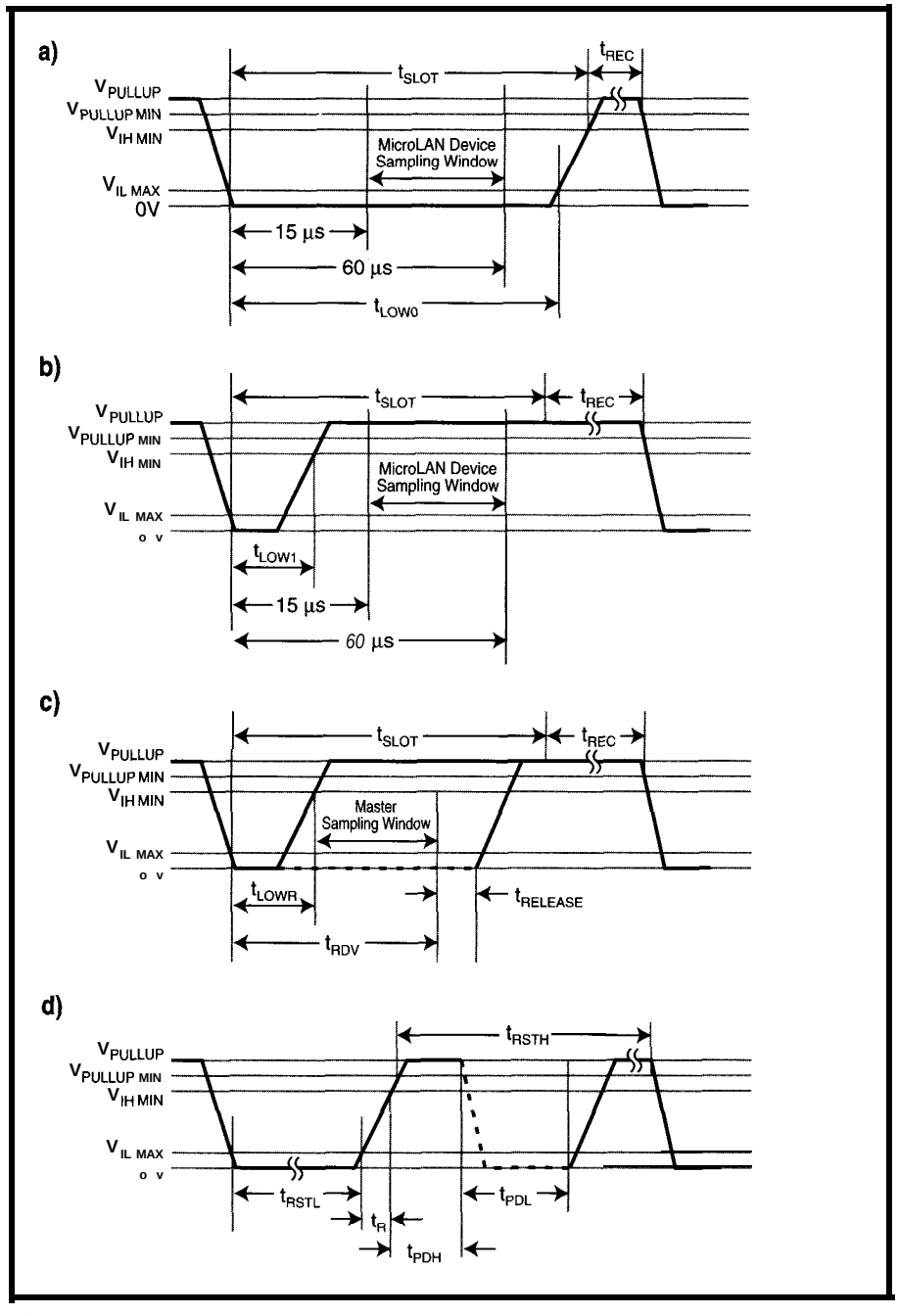


Figure 6—*The master initiates bit-by-bit transfers by driving the data line low. a) To write a 0, data remains low for the duration (60 µs) of the slot. b) To write a 1, the data line is driven low, but quickly released to float high for sampling by the slave. c) A read cycle is similar, except the data is sampled by the master. d) A reset cycle (960 µs) signals slave devices to assert their presence and prepare for action.*

Listing 1—*As this* **example** *for a '51* **CPU** *shows, writing a MicroLAN* **driver** *(i.e.,* **generating the different** *slots) isn't* especially hard, *but* **timing is critical.**

```
; TouchByte sends the byte in the accumulator to
; the touch memory and simultaneously returns one
; byte from the touch memory in the accumulator.
; The NOPS are intended to give optimum performance
; with a 11.0592.MHz crystal. They make the pulses
; as long as possible, consistent with the touch
; memory's timing constraints. With other crystal
; frequencies, adjust the delays to conform to the
; touch memory's timing requirements.

TOUCHBYTE:
        PUSH    B                       Save the B register.
        MOV     B,#8                    Setup for 8 bits.
BITLOOP:
        RRC     A               ; 1.    Set bit in carry.
        CALL    TOUCHBIT        ; 2.    Send bit.
        DJNZ    B,BIT_LOOP      ; 2.    Get next bit.
        RRC     A                       Get final bit in ACC.
        POP     B                       Restore B register.
        RET                             Return to caller.
TOUCHBIT:
        CLR     DATA-BIT        ;       Start the time slot.
        NOP                             Delay to make sure
        NOP                     ; 1.    that the Touch Memory
        NOP                     ; 1.    sees a low for at
        NOP                     ; 1     least 1 µs.
        MOV     DATA_BIT,C      ; 2     Send out the data bit.
        NOP                     ; 1     Delay to give the
        NOP                     ; 1     data returned from
        NOP                     ; 1     the touch memory
        NOP                     ; 1     time to settle
        NOP                     ; 1     before reading
        NOP                     ; 1     the bit.
        MOV     C,DATA_BIT      ; 1     Sample input data bit.
        PUSH    B               ; 2.    Save B register.
        MOV     B,#12H          ; 2.    Delay until the end
        DJNZ    B,$             ; 36    of the time slot.
        POP     B                       Restore B register.
        SETB    DATA-BIT                Terminate time slot.
        RET                             Return to caller.
```

THIS LAN IS YOUR LAN?

If you have an application that calls for the Dallas touch technologies, MicroLAN (i.e., the addition of time, temperature, and I/O) is certainly a winner, as long as you don't "byte" off more than the LAN can chew. The advantage is maximized if your application is PC-based to exploit the Dallas host software, tools, and examples.

Even if you don't need any cans,the Dallas solder-mount products are a competitive alternative to other clocked serial offerings. And needless to say, MicroLAN starts looking real good if you've only got one pin to spare! ❑

Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.

SOURCE

MicroLAN, DS2407, DS1920, DS2407, DS2401
Dallas Semiconductor
4401 South Beltwood Pkwy.
Dallas, TX 75244-3292
(214) 450-0448
Fax: (214) 450-3715
http://www.dalsemi.com/

IRS

431 Very Useful
432 Moderately Useful
433 Not Useful

CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(860) 871-l 988-Four incoming lines
Internet E-mail: sysop@circellar.com

This month, I thought we'd deal with the analog world for a change. In the first thread, we take a very quick look at what would be necessary to measure several characteristics of a sample capacitor.

In the other thread, we discuss the merits of several schemes for protecting an LED in an AC application. Everyone has their favorite.

Capacitors

Msg#: *8676*
From: Wayne Bush To: All Users

I am interested in measuring the nonideal properties of capacitors, in particular dielectric absorption and dissipation factor. I would like to measure D (dissipation factor) to 0.001 with 0.0001 resolution. Can you obtain this specification with a bridge circuit?

I also want to see how D changes with frequency (10 Hz to 1 MHz). The capacitance values I'm measuring are in the 40–100-pF range. How do you measure dielectric absorption? Dielectric absorption is modeled as a resistor (Rda) and capacitor (Cda) in parallel with a nominal capacitance (C). Can Rda and Cda be separated out from the capacitor?

Any information or references would be appreciated. Thanks.

Msg#:15374
From: Pellervo Kaskinen To: Wayne Bush

Yes, you can measure the dissipation factor to those resolutions. But the bridge has to operate at high frequencies. The low-cost (if there is such a beast!) bridges are limited to 120-Hz and/or 1 -kHz operation. I think that even 10 kHz might not provide the answers you are seeking.

What you probably need is a true RF bridge: 1 MHz and higher adjustable frequency. You may find some old, manual ones from used-equipment resellers, but still be prepared for the sticker shock.

Another option is finding a Q-meter. It is nominally intended for inductances and tank circuits, but if you make a good reference coil (thick silver-plated copper wire, air-core coil), you can then connect the capacitor in parallel and see the effect. A little cumbersome for measurements, but gets the job done. Again, there is a danger of severe sticker shock.

Commercial units, such as HP4284A, offer the 10-kHz to 1 -MHz range and a "D" resolution meeting your needs and sell for some $11,500. The Q-Meter was in a 1973 catalog at $1650. I believe it or similar is still sold by Boonton, but I have no idea about the present list price. I think I have seen *used* equipment for sale at double the 1973 price.

Msg#: *8837*
From: Brad Sanders To: Wayne Bush

You charge the cap, discharge it, then measure the remaining charge.

See Richard Marsh's most excellent series on capacitors from *Audio* magazine, circa 1977-1980 (somewhere in there). This is basically "the article that got things started," and it will tell you everything you need to know (including schematics of test gear]. Beyond that, look for *anything* else with Marsh's name on it; he's made a living off his patents in this area, and is a really good guy besides..

Warning: 100 pF is a very low value for performing these tests.

Diode reverse bias

Msg#:10442
From: Joseph Lehman To: All Users

I have a question on how to protect the LED on an opto-coupler from a reverse bias of ≥3 V. In the past, I've put a diode in parallel with the reverse polarity. This limits the reverse bias of the opto to the forward bias of the parallel diode.

In the current problem, I cannot do this since in the reverse case my control chip cannot source the required current. The control chip is designed only to sink the correct current.

Msg#:15375
From: Pellervo Kaskinen To: Joseph Lehman

What required current? The protection diode is not there "requiring" any current. It is there to safely bypass any *unrequired* current. If your chip cannot source anything in that direction, so much better. That allows a smaller and cheaper diode to be used!

If I try to read more between the lines of your message, it starts looking a bit like you are not using an open-collector

output and the LED from there to the positive rail. You have some sort of split power rail or multiple LED arrangement? In many such cases, the LEDs themselves can be connected in antiparallel and protect each other. They just work with opposite polarity drives.

If that is not the case, you still might be able to use the traditional parallel diode, but you need another, series diode from the chip output to the LED/diode combo. You probably also need a current-limiting resistor in series with everything.

Is this what you are battling?

Msg#:10856
From: George Novacek To: Joseph Lehman

So, why don't you put the diode in ***series*** with the optocoupler LED? Surely, you have more voltage headroom than just the 0.55 V the series diode would drop?

Msg#:10754
From: Ken Simmons To: Joseph Lehman

Use a 3-V zener (if such exists) or string together four 1N914s in series and apply that string reverse-biased across your drive diode. This way, you'll have a breakdown at 2.8 V (or so) where the '914 string will conduct.

You may have to experiment with your meter to discover the proper threshold you want to build with a diode chain.

Msg#:11493
From: Joseph Lehman To: Ken Simmons

I was thinking of putting my diode in series, but I don't know what the reverse voltage would be on the individual diodes. If the series has 5 V across it, what is the voltage across the individual diodes?

Msg#:12941
From: Ken Simmons To: Joseph Lehman

All general-purpose silicon diodes will have 0.6-0.7 VDC across them when forward-biased and conducting. Germanium diodes (a la 1N34) will have 0.2–0.3 VDC across them when conducting (which is why they're used as the audio detectors in AM radios).

As long as your reverse potential doesn't exceed the diode's rated reverse breakdown voltage (I believe it's something like 75 VDC for 1N914s), you shouldn't have a problem. In the situation like you described, it's not a factor. (In a series string, multiply the individual breakdown voltage by the number of diodes in the string to get a total breakdown figure. For a string of five 1N914s, you're looking at nearly 400 VDC before you achieve reverse breakdown in the string!)

Msg#:14115
From: Joseph Lehman To: Ken Simmons

In my case, the opto's diode reverse is 3 V, and I was going to use a 1N4001 with a reverse of 50 V. Does this mean the effective reverse voltage for the series would be 150 V? It sounds like this solution will do what I want. Thanks for the help.

Msg#:14586
From: Ken Simmons To: Joseph Lehman

Three 1N4001s in series will indeed give an effective reverse breakdown voltage of 150 V (plus or minus any tolerances of the diodes).

Msg#:14325
From: Dave Tweed To: Ken Simmons

It isn't that simple by any means. The voltage divides itself across the diodes in proportion to their reverse-biased resistance, which varies all over the place. The "best" diode in the string (lowest reverse leakage) will take the highest proportion of the voltage and break down, causing the next best one to fail, and so on. In high-voltage power supplies, equalizing resistors of about 100 kΩ are always connected across the diodes in a series string.

Getting back to the original problem, which if I remember correctly, involved an open-collector driver connected to a LED-based optocoupler.. .. My first question is: Why worry about it? How is an open-collector driver ever going to be able to reverse-bias the LED in the first place?

Assuming that there's some kind of connector in the path that would let a user connect the LED backwards to the driver, why not follow the usual practice of putting a diode in parallel with the LED, but pointing the other way? The LED will never see a reverse bias greater than the forward drop of the diode, and the current-limiting resistor works fine in either direction.

Msg#:15324
From: Ken Simmons To: Dave Tweed

That's quite true. However, the diode-to-diode variances (due to manufacturing, etc.) are usually slight enough to not worry about.

If it's really a problem, you can use high-breakdown rectifiers (e.g., 200-V+ PRV) to guarantee the safety. They're usually only a cent or two more than the low-PRVs (e.g., 1N4004 vs. 1N4001).

Therefore, in a practical sense, it's not a real big issue.

Msg#:16364
From: Dave Tweed To: Ken Simmons

Ah, but that's not the situation we're talking about, is it?

CONNECTIME

We were talking about putting a silicon diode in series with an LED. Do all LEDs have higher reverse-leakage currents than all diodes?

Msg#:16991
From: Ken Simmons To: Dave Tweed

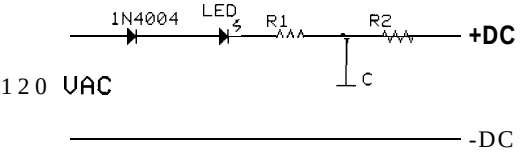
No, we were talking about an inverse-parallel diode connection with that LED. The protection diode/string would be backwards-connected across the driving LED to provide reverse-voltage protection and keep the LED from frying with >3-VDC reverse bias.

As for LED reverse leakage, I'm not familiar with such specs. I do know they're not as forgiving of reverse potentials as regular silicon diodes are.

Msg#:19132
From: George Novacek To: Dave Tweed

I think you guys are splitting hairs and worrying about rather superficial problems. Personally, I prefer the parallel combination of inverse-biased diodes for the majority of circuits.

I can think of only one situation where I needed to use a diode and an LED in series. It was a 120-VAC DC current supply with an on indicator which had to dissipate minimum power and be cheap:



We probably built at least 100,000 of those over 5 years. I don't recall a single one coming back because the "power supply" failed.

Msg#:20867
From: Pellervo Kaskinen To: Joseph Lehman

A series-connected silicon diode alone *does not* protect an LED against reverse bias. The reasons are two-fold:

1. Leakage. Until there is the breakdown, the LED leakage is orders of magnitude lower than the leakage of a typical 1N4001 or similar. On series-connected elements, the voltage concentrates on the higher resistance (lower leakage) element. This is handy for controlling the flow of energy through a switch that is in series of the load!

With your example, let's put in some numbers (out of my sleeve, admittedly) for working it out. Starting with the 5-V source and assuming 10-mA normal forward operation, you are likely to actually have three elements in series: the diode, the LED, and a 300-Q resistor. All this in addition to whatever switching device (IC, transistor, etc.) you might have. Well, 300 Ω is 300 Ω, but what are the diode and the LED? They depend heavily on the temperature, ambient humidity, and so on, but I would put in a number of 100 MΩ for the diode and 10 GΩ or more for the LED. Compared to these, the 300-Ω resistor is insignificant.

So, what voltage will each element see? Roughly 4.95 V over the LED and 0.05 V over the diode. The diode did nothing to prevent the LED from being reverse biased over its limit of 3 V. Actually, the resistance numbers I quoted were for *below* the breakdown limit. When the breakdown starts, the resistance of that element drops. In fact, if the 3 V is the actual breakdown voltage of the LED, then that is the exact voltage there will be. In effect, you see a combination very similar to a zener diode and the current-limiting resistor (in this case, the 100 MΩ of the diode). A 3-V regulator circuit of sorts.

Whether the 50 nA of current through the LED is harmful or not is another issue. And whether the 3 V is the actual avalanche voltage or a maximum safe reverse voltage should also be considered, but I leave it for now outside of this discussion.

Putting most any resistor or the usual antiparallel diode over the LED in addition to the series diode is a way of getting around these issues. Then any series diode leakage goes through that added element rather than through the LED. Even with a 10-MΩ resistor, the voltage would divide as about 4.5 V over the series diode and only 0.5 V over the LED.

2. Speed. After conduction, a diode is not turning off instantly (nor does the LED, but let's forget that for now). If

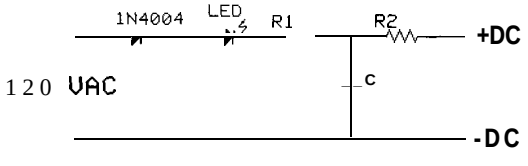
CONNECTIME

we change the polarity of the supply voltage from forward to reverse, there is a time that the diode is unable to block the reverse flow of current. For the 1N4001-series of diodes, we are talking about at least 30 µs at the current levels I have tested and probably still several microseconds at the 10 mA level.

If the LED happens to recover any quicker than the diode, there is little resistance to limit the reverse current. We may be dependent on the 300 Ω, and that is too little, again assuming that the 3 V might be an avalanche point rather than a safe value.

Again, if we have a small, fast diode such as 1N4148 in parallel with the LED, the protection is complete. The 1N-4148 has a recovery time of 4 ns, which is fast enough for the LED.

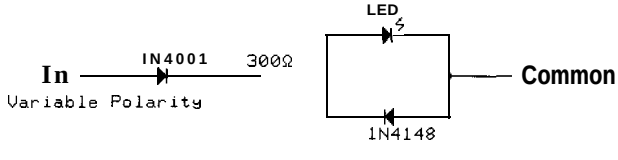
I don't go into the inductances and the capacitances in the circuit (this time anyway]. They, though, are what cause the transients and the need for the LED protection. But here is the proposed circuit:



I use basically this circuit for my optocoupler inputs, and if I keep the 1N4148 close to the input of the optocoupler (to minimize inductance-related problems), there have never been problems. In fact, I normally use a current limiter device such as 1N5306 instead of the 300-Q resistor to cover a wider input voltage range. The 1N5306 provides a constant 2.2-mA current within a 5-60-V range, with my derating-the diode itself is specified up to 100 V.

Msg#:22665
From: Jan Verhoeven To: Pellervo Kaskinen

My favorite optocoupler limiter consists of the following:



In which the current is determined by the resistor. The components are cheap and easy to get, but I never paid much attention to the reverse-bias options (the TL3 17 should take care of that).

What's your opinion on this? Is this safe enough for general-purpose situations?

Msg#:26239
From: Pellervo Kaskinen To: Jan Verhoeven

Your circuit is fine for close location of the components (minimized inductances) and when the input voltage is less than about 35 V. The 1N5306 can handle up to 100 V.

Most of my circuits use optical isolation when there is a considerable distance, ribbon cables, and so on. I am mostly designing products for low-volume applications, where any interruption of the functionality is terribly costly, so I try to avoid any cutting of the corners. But "your mileage may vary," as they say on the car sales lots.

A way of getting the same constant-current behavior before the era of LM3 17 was a discrete depletion-mode FET and a resistor. Too bad, those FETs are difficult to find nowadays, especially with high voltage ratings. I would welcome any suggestion of a 300-V or higher rated depletion-mode FET still available.

We *invite you to call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (860) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity,* and 300, *1200, 2400,* 9600, or *14.4k bps.*

ARTICLE SOFTWARE

Software for the articles in this and past issues of *Circuit Cellar INK* may be downloaded from the Circuit Cellar BBS free of charge. It is also available on the Internet at **ftp://ftp.circellar.com/pub/circellar/**. Web users should point their browser at **http://www.circellar.com/**. For those with just E-mail access, send a message to **info@circellar.com** to find out how to request files through E-mail.

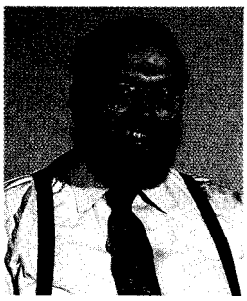
For those unable to download files, the software is also available on one 360-KB IBM PC-format disk for only $12. To order Software on Disk, send check or money order to: Circuit Cellar INK, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Mastercard and call (860) 8752199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

IRS

434 Very Useful **435 Moderately Useful** **436 Not Useful**

PRIORITY INTERRUPT

'Bots Got No Respect



I've always advocated that experience is the greatest teacher. That's one of the reasons Circuit *Cellar INK* has such a hands-on approach to technology. If a picture is worth a thousand words, an engineering experience is worth a thousand opinions.

Over the years, I've presented articles on motor controllers, ultrasonic and infrared ranging systems, remote keypad entry, sophisticated duplex communication schemes, and of course, computers and controllers ranging in sophistication from a single-board PC to a simple chip with BASIC. I've demonstrated remote video transmitters, infrared sensors, and even a video camera made from a DRAM chip. What I've never done, however, is put all these elements together in a single project where experience better balances opinion.

A couple weeks ago, I had an enlightening experience at the International Robotics Competition held at Trinity College in Hartford, CT. As embarrassing as it is to admit this, it was my first real robotics show. While attending it did nothing to eradicate my preconception that-like any technical recreation-it would have its share of geeks and gearheads, the experience decisively confirmed that this was no demonstration of fools and their toys.

This specific contest required navigating an 8' x 8' maze. Within the maze were three rooms and a lighted candle. The object of the contest was to navigate the maze, find the room with the candle, and extinguish the flame in the least amount of time. The typical entrant simply turned on a fan or initiated a pump sprayer when approaching the candle. Those who adhered to the "bigger is better philosophy employed more brute-force techniques like dumping a whole CO, bottle as they came in view of the candle.

As you might have guessed, it was the radical executions that these vehicular contraptions employed in getting to that candle that separated the winners from the losers. Because of the task, all of the entrants had to deal with motor power and efficiency, measuring distances and calculating direction, separating the IR radiation of a candle from its surroundings, and then coordinating it all.

Perhaps because the term "robot contest" sounds like a media-sponsored joke or because, in a snapshot sound bite, things appear just like another computer game, the vast majority of people miss the significance of it all. Even I admit that, until actually viewing a robotics competition, it was hard to grasp its real-world implications.

When I interview an engineer for a job, I'm looking for a combination of aptitude and experience. Young engineers have lots of class work but very little real experience. Civil-engineering schools often use bridge-building contests to teach statics and dynamics in a real, albeit significantly smaller, application.

I'd respect the experience of someone who had designed an artificial-intelligence-commanded, embedded-computer-controlled, motorized platform with sonar-sensing infrared-seeking capability. Such a person would have to understand peak currents, PID loop control, and energy management. Most certainly, this person would also have first-hand experience with the benefits and detriments of distributed versus central process control.

Of course, until more interviewers come to respect the importance and substance of a robot contest, none of them will understand the experience you have to offer.

steve.ciarcia@circellar.com