CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#62 SEPTEMBER 1995

EMBEDDED PC

SPECIAL BONUS SECTION: EMBEDDED PC

EMBEDDED APPLICATIONS

Developing an Engine Control System

Embedded Debugging Techniques

Exploring Virtual-86 Mode

DC Motor Control with I^2C

$3.95 U.S.
$4.95 Canada

guest EDITOR'S INK

Mark That Horse

No question—most embedded processors are specifically designed for and implemented in embedded applications. It works better that way. The size is right. The power's right. The architecture's right.

But, occasionally, there's an exception. Take the Z80. As a desktop processor, it gained its popularity. Its processing power, low cost, and familiar instruction set made it a prime candidate for embedded systems. Although supplanted by the more powerful 2180, the Z80 remains an active participant in the embedded world, its popularity pushing it into many embedded applications still active today.

Knowing the monetary power of the embedded market, newer "desktop" processors anticipate the needs of an embedded system as well. ARM processors quite capably drive the Acorn Archimedes computer as well as the Newton and 3D0. Here too, the embedded prowess of the processor exceeds that of its desktop counterpart.

With the launch of Embedded PC, Circuit Cellar is placing its bet on the embedded PC. Sure the size, bus, and power consumption of the desktop PC has had to be rearranged for the embedded world, but it's out of the gate. PC/104 has exponentially accelerated it through the first stretch. How long will it be until embedded PC revenue exceeds that of the desktop PC?

Embedded PC is a 32-page quarterly insert devoted to bringing you the latest on the embedded PC race. Feature articles will cover topics such as off-the-shelf ISA-bus motherboards, expansion boards, networking, PCI, other buses, and embedded PC software. With our columnists Rick Lehrbaum and Russ Reiss, we gain a front-row seat on the PC/l 04 standard and how to implement the embedded PC in real-life applications.

This month, David Prutchi shows us how to hot swap with an active extender card for the ISA bus while Stefano Chiti-Batelli develops an ISA bus simulator. Rick starts off with how PCMCIA makes a complementary match with PC/104 while Russ overviews the embedded PC.

INK's September theme of embedded applications overlaps with **Embedded PC.** Ed Lansinger gives us an engine control system based on Motorola's MC68HC16, Stuart Ball shows us how to debug when we're short of I/O, and Gordon Dick covers both mechanical and electrical aspects of building an XYZ router table.

And, of course, there are our *INK* columnists. While Ed answers the all-time question: Why emulate a 8086 at all?, Jeff begins a two-part series on DC motor control using I^{2}C. Tom introduces us to the TSL230 intelligent optosensor, and John finishes his series on Dallas Semiconductor's newest processor with power-management capabilities.

Notably, with this issue, John crosses the finish line and takes his entrepreneurial enterprises on home. We salute John as he moves on to other endeavors.

Janice Marinelli
Managing Editor of **Embedded PC**

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Janice Marinelli

EDITORIAL ASSISTANT
Beth Andrix

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski
Rick Lehrbaum
Russ Reiss

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 (203) 675-2751. 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 0. Box 696, 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 190459613.

Cover photography by Barbara Swenson
PRINTED IN THE UNITED STATES

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST & MID-ATLANTIC Barbara Best (908) 741-7744 Fax: (908) 741-6823	**SOUTHEAST** Christa Collins (305) 966-3939 Fax: (305) 985-8457	**WEST COAST** Barbara Jones & Shelley Rainey (714) 540-3554 Fax: (714) 540-7103
	MIDWEST Nanette Traetow (708) 357-0010 Fax: (708) 357-0452	

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

12 **Developing an Engine Control System**
Part 1: System Architecture and Fuel Delivery
by Ed Lansinger

20 **Embedded Debugging Tricks**
by Stuart Ball

24 **Designing an Industrial-grade XYZ Router Table**
by Gordon Dick

32 ❑ **Firmware Furnace**
Journey to the Protected Land: Entering Virtual-86 Mode
Ed Nisley

40 ❑ **From the Bench**
Creating the SMART-MD
DC Motor Control for the I²C Bus
Jeff Bachiochi

84 ❑ **Silicon Update**
IC de Light
Tom Can trell

88 • J **Embedded Techniques**
Power Management with the DS87C530
Part 2: The Software
John Dybowski

EMBEDDED PC
See pages 47-82 for Our NEW Bonus Section

INSIDE ISSUE 62

2 **Editor's INK**
Janice Marinelli
Mark That Horse

6 **Reader's INK**
Letters to the Editor

8 **New Product News**
edited by Harv Weiner

ConnecTime 99
Excerpts from the Circuit Cellar BBS conducted by Ken Davidson

Steve's Own INK 11
Welcome to Gambleticut

Advertiser's Index 81

READER'S INK

MAKING A DIFFERENCE

Congratulations to Gregg Norris and Eric Wilson for their excellent award-winning design of the Eye Mouse (*INK* 59).

The subject of adaptive devices for the severely disabled is near and dear to my heart. A few weeks ago, I lost a good friend to ALS (Lou Gehrigs disease). By the grace of God, her own courage, and a multidiscipline team of caring people, she was able to communicate until the last days of her life. It was a constant scramble to reinvent things on the fly. I'd like to pass on what we learned in hopes that it will help others.

The HeadMaster is a computer input device that uses head movements for mouse control. It works by ultrasonics with the sensors mounted on a telephone-style headset. Its keyboard is emulated on the screen through software, and the key and mouse input is by a sip and puff switch. Contact Prentke Romich at (800) 262-1984.

For people who cannot use a switch, Magic Cursor software emulates clicks and menu pull downs through head movement. Characters are typed by simply suspending the cursor over the letter for a preset amount of time. This company also offers a "head" mouse, which consists of a reflector that mounts to any part of the body that can be controlled. Contact Madenta at (800) 661-8406.

Our biggest success was the head-mounted laser presentation device and a set of alphabet cards. This inexpensive device never crashes, locks up, or has power problems.

The patient had only slight head, neck, and knee movement, so she alerted aids using a piezo buzzer connected to an air-cushion switch.

I'm continuing with volunteer work and would be thrilled to exchange ideas with anyone through the BBS.

Jay Davis
Derby, KS

EYE MOUSE SENSORS

There have been a number of enthusiastic enquiries about "The Eye Mouse" (*INK* **59**). Specifically, several readers have had a hard time locating the A-7 sensors used in the project. So, here's where to call:

Lead-Lok, Inc.
(208) 263-5071
Fax: (208) 263-9654

Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial offices at (203) 875-2199.
Fax: All faxes may be sent to (203) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (203) 871-1988 with your modem (300–14.4k bps, 8N1).
Internet: Electronic mail may also be sent to our editors and regular authors via the Internet. To determine a particular person's Internet address, use their name as it appears in the masthead or by-line, insert a period between their first and last names, and append "@circellar.com" to the end. For example, to send Internet E-mail to Jeff Bachiochi, address it to jeff.bachiochi@circellar.com. For more information, send E-mail to info@circellar.com.

Syndetix Embedded Controller™

Embedded Controller with Flash Memory

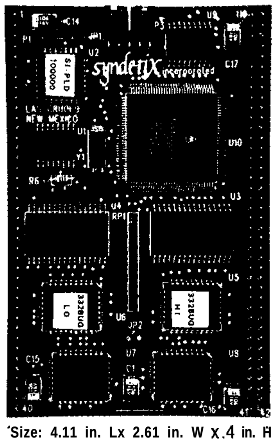
The Syndetix Embedded Controller™ (S.E.C.™) combines on-board CPU32Bug and Flash Memory to reduce development time and improve flexibility when software modifications are required. This mezzanine style embedded controller is suitable for data acquisition, process control, and other real-time applications.

16.76 ***MHz** MC68332 or MC86331 CPU*
***128KB** Zero-wait State EPROM*
***256KB** Zero-wait State Static RAM*
***256KB** Zero-wait State Flash memory*
Built-in RS-232 interface
*Motorola **CPU32Bug** with Flash load command*
*Industrial Temperature Range (**-40°** to **+85°**)*

Price: $365

Syndetix Incorporated
2820 North Telshor Blvd.
Las Cruces, NM 88011

Telephone (505) 522-8762
FAX (505) 521-1619
E-mail secQsyndetix.com



'Size: 4.11 in. Lx 2.61 in. W x.4 in. H

CPU32Bug is a trademark of Motorola, Inc.

#103

NEW PRODUCT NEWS

Edited by Harv Weiner

SUPER SMALL EPROM EMULATOR

Mitech announces an EPROM emulator that is easily embedded into a small space. The **Mitech Emulator** can simulate all of the popular 27xxx series EPROMs used with 8-bit microcontrollers and features an access time of under 100 ns. With the purchase of additional emulators, code can be developed for processors with larger data paths.

The emulator has a low profile (about 2" x 1" x 1") and plugs directly into an EPROM socket. This small size lets the user embed the emulator deep into most systems, even if the system is enclosed. The emulator, with provided DOS-based software, can be uploaded through any of the four standard COM ports in an IBM PC-compatible computer. The uploader can also be run from within Windows. The software supports both binary and Intel hex file formats.

The unit draws power directly from the EPROM socket or through an umbilical link. Communications, backup power, and a remote reset switch are through the provided umbilical cable. After uploading is complete and power is applied to the system, the umbilical link can be detached. With the umbilical attached and a battery installed, the system can be turned off and the emulator unplugged without losing its memory contents.

The emulator sells for $199.95, which includes shipping and handling.

Mitech Electronics Corp.
411 Washington St.
Otsego, MI 49078
(616) 694-9471 • Fax: (616) 692-2651 **#500**

8-BIT MICROCONTROLLER AND DEVELOPMENT KIT

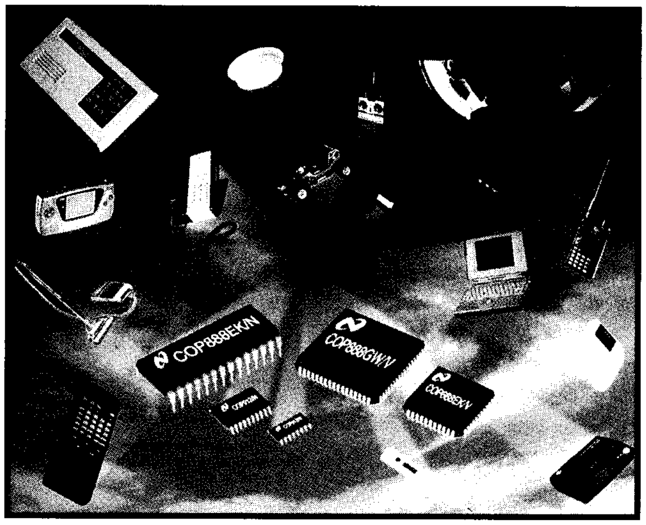
National Semiconductor has announced three new controllers that add analog functionality (**COP888EK**), hardware multiply and divide **(COPSSSGW),** and low cost (COP912) to its family of COP8 8-bit embedded microcontrollers. A low-cost simulator, the COP8 **EPU,** lets programmers debug code and hardware designs.

The COP888EK integrates an analog function block into the COP8 architecture. This additional circuitry enables the COP888EK to be used as an A/D converter with 16 bits of resolution for applications that include battery chargers, security systems, remote data-monitoring systems, and control systems. The chip contains 8 KB of ROM and 256 KB of RAM.

The COP888GW includes built-in hardware multiply-and-divide functions and is capable of performing a 16 x 8 multiply in 1 us or a 24 × 16 divide in 2 µs. Applications for the device include fuzzy-logic controllers and stepper-motor drives. The chip comes in a 68-pin PLCC package and includes 16 bytes of ROM and 512 bytes of RAM.

The COP912 is a truly low-cost device providing 768 bits of ROM, 64 bytes of RAM, a 16-bit timer, three interrupts, and power-saver mode. It supports Microwire/Plus serial I/O.

The COP8 Evaluation and Programming Unit (EPU), controlled via an RS-232 link from any standard PC running MS-DOS, provides simulated access to all of the features of the COP8 microcontrollers either interactively or through applications. This interface enables designers to use the COP8 EPU for step-by-step test and debug of their hardware and software designs against both assembly-level code and system-level software.



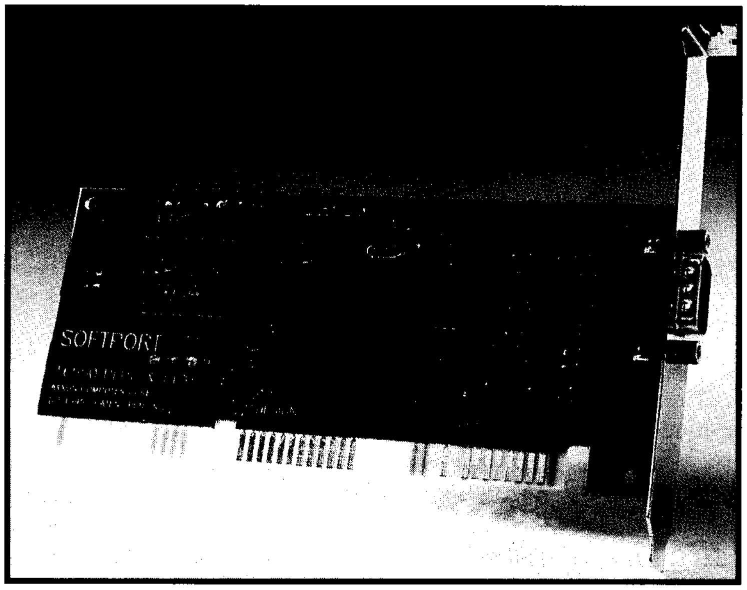
The COP8 EPU package comes complete with all necessary components for debugging designs and/or programming COP8 devices. The COP8 board with DIP programming socket, a DE-9 RS-232 communications cable, power supply, 40-pin DIP target interface cable, two 40-pin DIP COP8780CD OTP samples, host software diskette, National's COP8 assembler/linker package, and EPU-COP8 User's Manual are included.

The COP888EK sells for $4, the COP888GW for $6.75, the COP912 for $0.65 (10,000 quantities), and the COP8 EPU for $135.

National Semiconductor Corp.
2900 Semiconductor Dr. • Santa Clara, CA
95052-8090 • (408) 721-5000 **#501**

NEW PRODUCT NEWS

PLUG-AND-PLAY SERIAL PORT



A plug-and-play serial port for IBM and compatible computers has been introduced by Axxon Computer. **Softport** provides a single, 16-bit, high-speed 16550 serial port that is compatible with Windows 95. Softport is completely free of jumpers for configuration in both plug-and-play and Legacy environments (non-plug and play). Softport functions in either S-bit or 16-bit ISA or EISA slots.

Included software can change the serial port address and select from interrupts 3-7, 9-12, or 15. With plug-and-play environments, Softport automatically configures without conflict when selecting I/O address and IRQ for the onboard 16550 UART. The high-speed serial port is ideal for 9600+ bps external modems, pointing devices, or fast data transfer.

Softport sells for $69 and comes with a 5-year warranty.

Axxon Computer Corp.
3979 Tecumseh Rd. East • Windsor, ON • Canada N8W 1J5
(519) 974-0163 • Fax: (519) 974-0165 **#502**

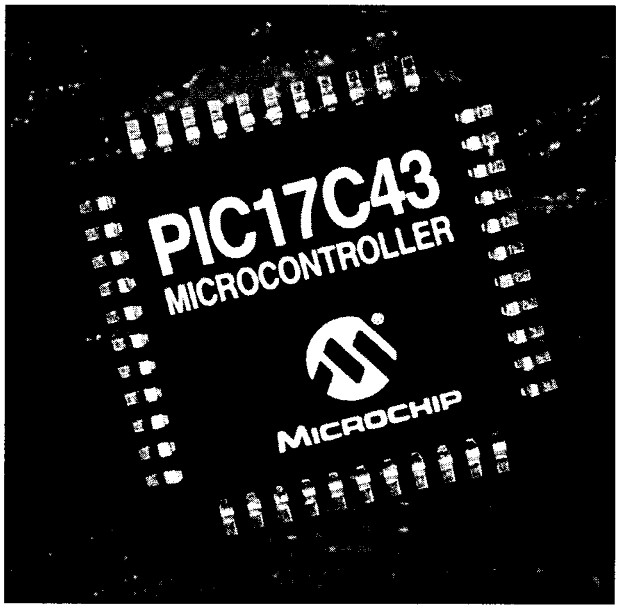
8-BIT OTP MICROCONTROLLER

Microchip Technology has introduced a high-performance, one-time-programmable (OTP), 8-bit RISC microcontroller that provides an instruction execution speed of 160 ns at 25 MHz. The **PIC17C43** offers unique execution of its two 8 x 8 unsigned hardware multiply instructions in a single instruction cycle. The fast execution throughput offers a cost-effective alternative to more expensive 16-bit microcontrollers and dedicated digital signal processors for certain math-intensive applications.

The 40-pin PIC17C43 offers 4K x 16 OTP on-chip EPROM program memory and 454 bytes of user RAM for longer and more complex software algorithms. These features make the PIC17C43 ideal for demanding real-time embedded control applications where high 8-bit performance is critical. Applications include industrial process control, manufacturing equipment, motor control, robotics, appliances, security systems, and data loggers.

The PIC 17C43 features include two PWM outputs of 97.7 kHz at S-bit resolution and 24.4 kHz at 10-bit resolution (for the 25-MHz device) to enable more precise control of process functions. Two fast-capture inputs with resolution of up to 160 ns, a fast full-featured serial interface (USART), and a watchdog timer with on-chip RC oscillator are also included.

The PIC17C43-16/P (1 6-MHz version in a plastic DIP package) is priced at $9.40 in quantity.

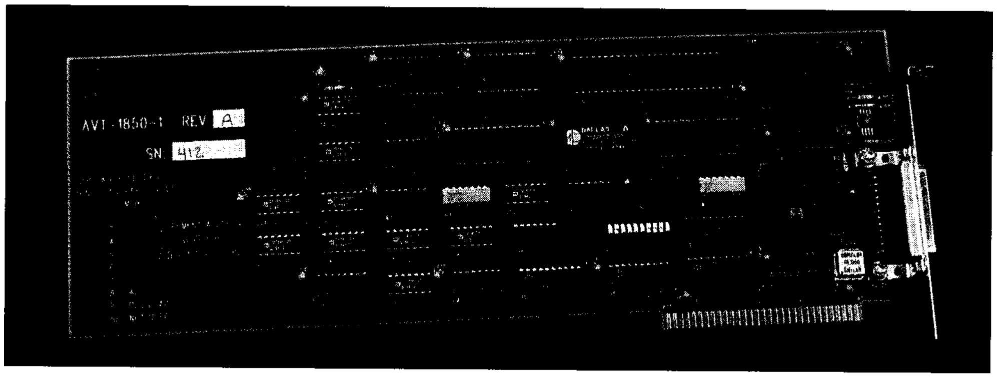


Microchip Technology, Inc.
2355 West Chandler Blvd. • Chandler, AZ 85224-6199
(602) 786-7200 • Fax: (602) 899-9210

#503

NEW PRODUCT NEWS

IN-VEHICLE MULTIPLEX BUS DEVELOPMENT



Advanced Vehicle Technologies introduces a development platform for automotive-based networks. The **AVT-1850-l** is a J1850-compliant system for IBM PC compatibles. SAE J1850 is a two-part standard for in-vehicle multiplex buses. These buses let sensors (temperature, pressure, speed, torque), actuators (locks, windows, etc.], and controllers communicate. External test equipment can also access the bus for diagnostics and maintenance.

The AVT- 1850-l reduces the risk in developing J1850-compatible nodes. Designed to be used with a PC, the board and software give a developer immediate access to the target hardware and interfaces. The board has all network-interface functions controlled by a Harris HIP 7030A0 onboard microcontroller. The microcontroller off-loads from the host processor all network-related activity, thus freeing the host.

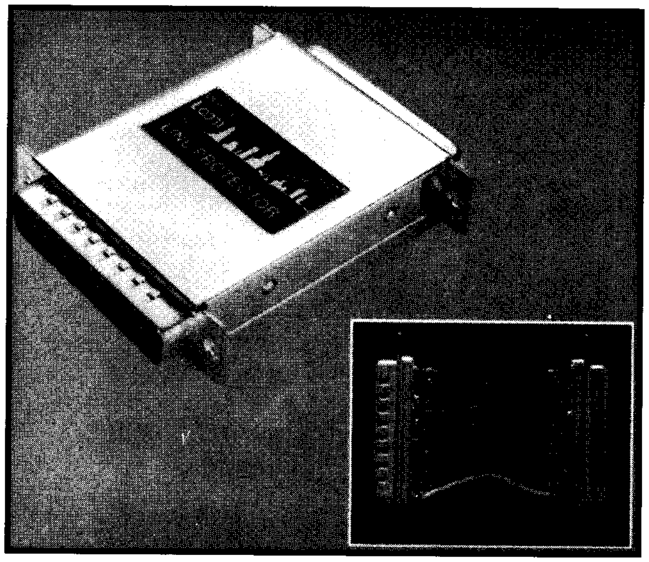
Onboard dual-port memory is mapped into the host computer's memory space, so the user can control and view the operation of the onboard microcontroller. All interfaces with the HIP 7030A0 are accessible to the designer via a rear-panel connector. These interfaces include S-bit parallel, 5-bit parallel, serial peripheral interface (SPI), as well as controller interrupt and status lines. Combined with internal timer and I/O functions, these interfaces offer a wide variety of functions and integration with nearly any sensor, actuator, and so on.

The software is an integrated environment that implements and debugs node and network software on one platform. Called the **Enhanced On-Line Software** (EOS), it consists of program development, program debugging, and network monitoring. EOS requires DOS 5.0 or higher. A Windows version will be available by year end.

The AVT- 1850- 1 Development Package sells for $1500 and includes telephone technical support and software upgrades for one year.

Advanced Vehicle Technologies, Inc.
1509 Manor View Rd. • Davidsonville, MD 21035
(410) 798-4038 • **Fax: (410) 798-4308** **#504**

DATA-LINE SURGE SUPPRESSOR



Data communication lines are susceptible to electrical disturbances and surges. In particular, the RS-232 serial port is vulnerable to damage resulting in burnt or damaged cards or motherboards. L-corn is offering an in-line device to act as a permanent safeguard.

Model DLPSS features silicon avalanche zener diodes on eight data lines (pins 2-8 and 20) to protect against transient voltages. Avalanche diodes react faster than MOVs and are voltage sensitive to provide a more accurate 27-V threshold level for sensitive microprocessing circuits. The power-handling capability is 600 W on each line. The device features an all-metal case with male and female connectors for in-line operation.

Model DLPS8 sells for $28.95 in single quantities.

L-corn
1755 Osgood St. • North Andover, MA 01845-1092
(508) 682-6936 • Fax: (508) 689-9484 **#505**

FEATURES

12 Developing an Engine Control System

20 Em bedded Debugging Tricks

24 Designing an Industrial-grade XYZ Router Table

FEATURE ARTICLE

Ed Lansinger

Developing an Engine Control System

Part 1: System Architecture and Fuel Delivery

Even though virtually all of today's automobiles have multiple computers under the hood, designing those computers is no trivial task. In this first article, Ed gives us a hand with the fuel-delivery system.

College students blasting around campus in a race car might get suspended at most schools. But, at Rensselaer Polytechnic Institute, such frivolity can earn you course credit, especially if that race car features a 68HC16-based engine control system.

This article is the first in a three-part series describing the system I developed as a senior project. After explaining system requirements, I'll diagram the system architecture and delve into its fuel-delivery subsystem.

The two subsequent articles deal with the ignition subsystem, other interface electronics, the code required to tie everything together, and testing and tuning the system. Ultimately, I'll cover the complete system: hardware, software, sensors, actuators, installation, and testing.

After finishing the series, I hope you will be able to duplicate this system for your own applications.

REQUIREMENTS

The Engine Control Module (ECM) was developed for Formula SAE, an intercollegiate competition sponsored by the Society of Automotive Engineers and the Big Three Automakers. Over 70 colleges across North America participate, each school building a single-seat, open-wheel race car from scratch. The cars are raced in an annual competition held in Detroit. Competition is fierce. Everyone is looking for an edge, the rules encouraging the development of advanced technology.

Carburetors had given us poor performance, so we chose a fuel-injection system to get a performance advantage. I developed the ECM to optimize engine-power output, provide better driveability than a carburetor, be easier to tune, and gain a base for future advanced control systems.

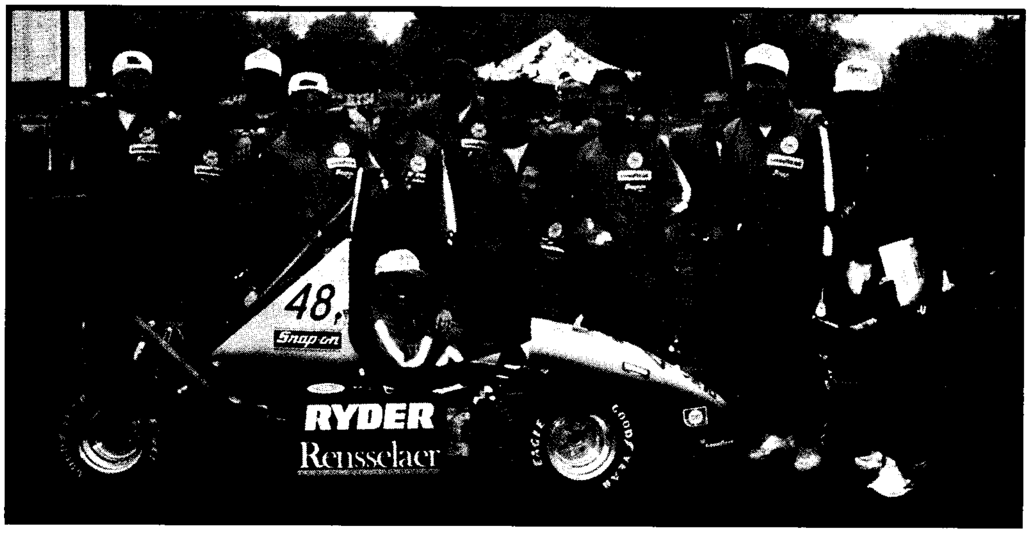


Photo I--Students on the 1994 Rensselaer *Formula* SAE Team used a fuel injection system they developed *from scratch to race in a national competition. The* author is standing *directly* behind fhe *person seated in the car.*

Fundamentally, the ECM needs to do just two things:

- provide the correct amount of fuel to the engine
- fire spark plugs at the right time.

It sounds simple enough, but think of the environment in which the ECM must operate. Our 550-lb. car with a 70-hp engine is made of composites and chrome-moly steel. It can out-accelerate, out-brake, and out-corner nearly all production automobiles. The g-forces, vibration, and shock leave drivers breathless and fatigued after only **15** minutes.

Now, consider that the ECM has to ride shotgun for countless hours of testing and racing. It must wring maximum power out of the engine at the driver's command. When possible, it must protect the engine and the driver from dangerous conditions.

Furthermore, it must remain in control at all times, even with RF ignition noise so severe it can reset a CPU.

ARCHITECTURE

The block diagram of the power-train in Figure 1 includes the ECM. The ECM is a microcontroller with interface electronics and control software. Figure 2 takes a look at the hardware inside the ECM, while Figure 3 depicts the software architecture running on the micro. As this series progresses, I'll cover the items in these figures with their respective subsys-tern. But, by having these figures now, you'll get the big picture up front.

Central to the proper operation of the ECM is the requirement that certain actions happen at the right time relative to the position of the pistons and valves within the engine. I refer frequently to the four strokes: intake, compression, power, and exhaust. It is important that you understand what goes on during each stroke (read the sidebar on the four-stroke engine if you need an update).

Early on, I made several system-design decisions based on the hardware available to me, competition rules, experience, and preferences. Here are the most important.

The 600-cc engine is from a high-performance motorcycle. It has four cylinders, a crank-position sensor, and no distributor. Four teeth, one long and three short, are cast into the engine's flywheel, which is attached to one end of the crankshaft. An inductive sensor generates pulses as the leading and trailing edges of these teeth rotate past. Engine position and speed are determined from this signal.

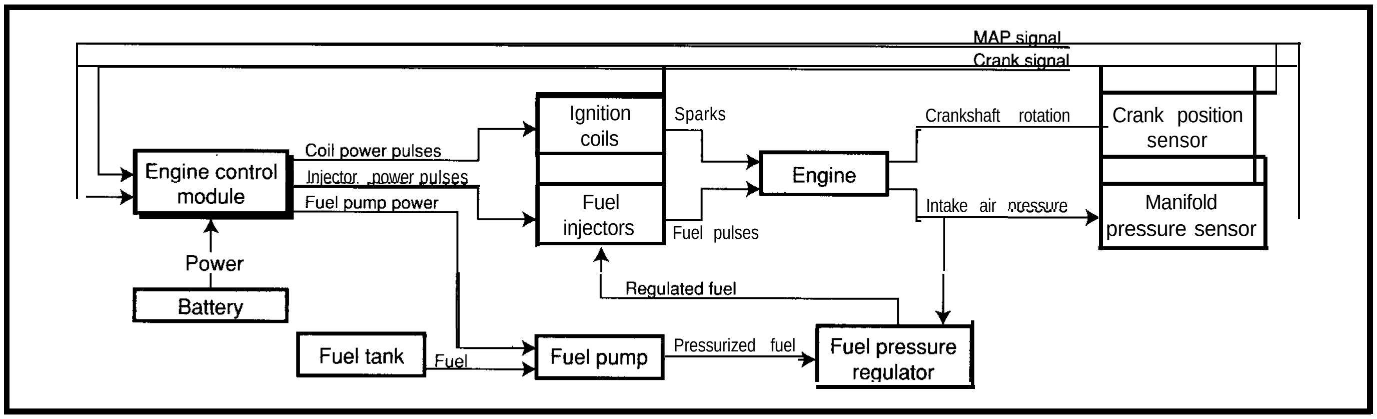


Figure I--The *Engine Control Module relies on inputs from all over the engine for making control decisions.*

Injection is by a technique known as *simultaneous double fire,* in which injectors fire in pairs once per revolution. This technique differs from a sequential setup that individually fires injectors every other revolution according to when the intake valve opens. While the latter technique gives greater control over fuel delivery, it requires an additional sensor to detect the camshaft position. The differences between the two disappear at high RPM, which is the operating region of concern for a race car.

To determine the correct amount of fuel to inject, I decided to use a speed-density system. Such a system uses two inputs to determine fuel flow: RPM and air pressure inside the intake manifold. RPM is calculated from the crank-position signal. Air pressure is measured by a manifold pressure sensor, otherwise known as a MAP sensor. I used an automotive MAP sensor which generates a 0-5-V output proportional to air pressure. A vacuum line runs from the intake manifold to the sensor.

Since the engine had no distributor, I used two ignition coils, each firing two plugs at the same time. This method works because one cylinder fires on the compression stroke while its twin fires on the exhaust stroke. Dual firing does no harm because a spark fires in the exhaust cylinder when there is nothing left to burn. There is little decrease in spark energy available to the compressing cylinder because the hot exhaust gases are ionized and add very little resistance to the spark current path.

Several microcontrollers would work for this application. I chose the Motorola 68HC16 because it is quite fast and was developed in conjunction with an automobile manufacturer specifically for this type of application.

The chip has built-in output-compare timers, so actuators can be controlled by a hardware time reference. It also has input capture timers that timestamp an input edge and cause an interrupt. An 8-channel A/D converter handles all current and near-term-future analog-input requirements.

A Background Debug mode gives greater control and flexibility during debugging. Other features, such as access to lots of memory, queued serial communications, and DSP instructions, are there for future expansion. The ECM uses the Motorola M68HC-16EVB evaluation board, which I found to be an excellent development tool. It comes with a very useful assembler, debugger, and simulator.

The code is in assembly. Because the system deals with physical objects that interact in well-defined ways, I chose an object-oriented approach. Achieving a clean design for the code was aided by using the physical system as a model for the software objects and the messages they send.

The impact of the overhead to support objects in assembler was insignificant, especially in light of the 68HC16's speed and address space. Actual processing time is only a small fraction of available CPU time, even at the highest engine speeds.

Two important objects in Figure 3 require a bit of explanation before moving on. The **Distributor** is driven by the crank-position signal and schedules the operation of the Coil and **Injector** objects. It does not represent a physical device, but its operation is similar to the mechanical distributors present on some types of engines.

AlarmClock is used by any object wishing to be sent a wake-up call at a certain time in the future from a few dozen microseconds to hours. These two objects are the autonomous nervous system keeping the ECM alive.

FUEL DELIVERY HARDWARE

Fuel is delivered by the fuel pump through a pressure regulator to the injectors, which are controlled by the ECM. A fuel injector is an on/off solenoid valve that gates pressurized fuel from the fuel rail into the intake manifold.

When the injector is opened, it sprays a mist of fuel at a constant flow rate. Each revolution, an injector is pulsed on for a short period of time. The duration of this pulse is termed *injector pulse width.* The width of the pulse determines the total amount of fuel injected per revolution and must be carefully matched to current operating conditions.

All this subsytem's mechanical elements up to the injectors are responsible for delivering a supply of clean fuel at sufficient pressure. I chose a high-pressure fuel system where the fuel is pressurized by an electric fuel pump before being sent to the injectors. A variety of OEM and aftermarket fuel-injection components are available for high-pressure systems.

The electric fuel pump runs on 12 V and draws about 8 A. It is mounted outside the fuel tank with an in-line fuel filter between it and the tank. This filter is absolutely essential. The smallest amount of grit can seize an electric fuel pump, making it unusable.

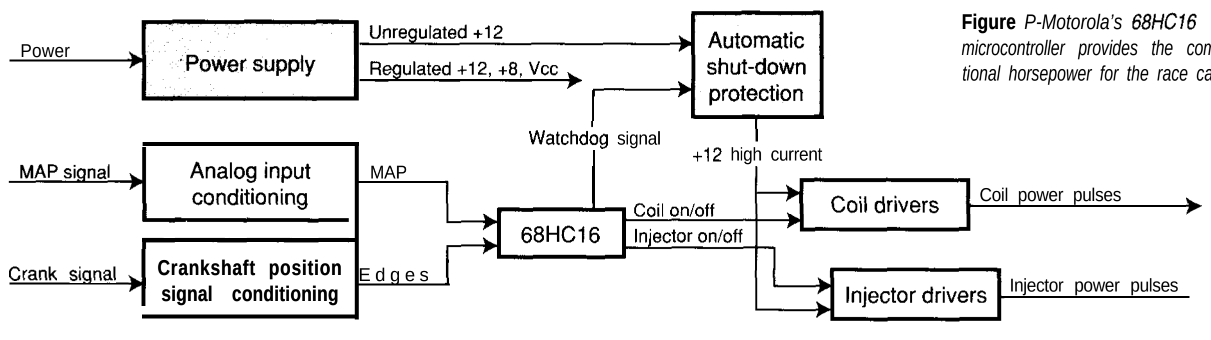


Figure *P-Motorola's 68HC16 microcontroller provides the computational horsepower for the race car.*

I used a fuel-pressure regulator to maintain pressure at 45 psi. The regulator routes fuel against a spring-loaded diaphragm which has a relief hole. When pressure exceeds 45 psi, this hole opens and routes excess fuel back to the fuel tank, reducing pressure.

In addition, the regulator is manifold referenced. There is a vacuum line from the regulator to the intake manifold. The air pressure from the manifold biases the spring force on the diaphragm, making it easier to open when the manifold pressure drops. This keeps the fuel pressure at a constant 45 psi above the manifold pressure. The constant pressure across the injector makes the system easier to tune.

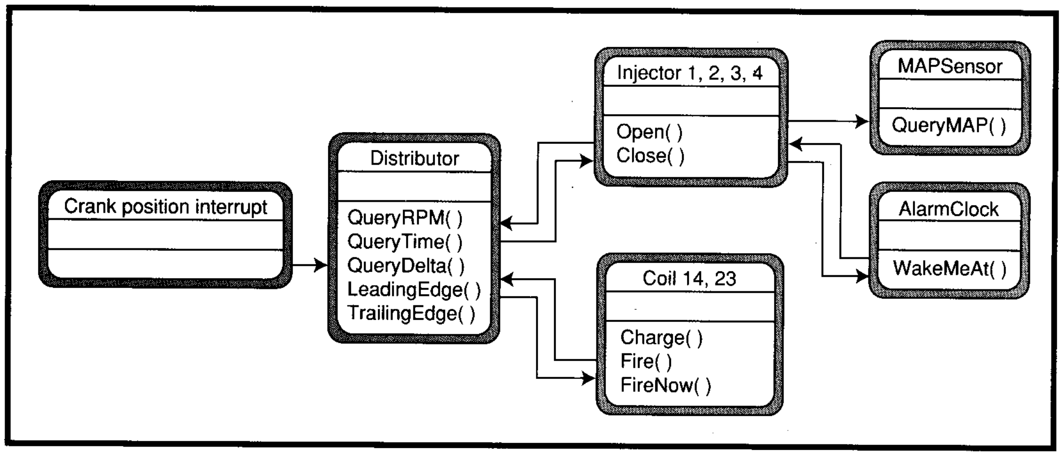


Figure 3—*The control software uses an object-oriented design modeled on the interaction between mechanical system components.*

Fuel injectors come in a variety of flow rates, so make sure you have the right one. The flow rate represents the mass of fuel the injector can deliver in a given time at a given pressure. This flow rate must match the mass flow rate of air through a cylinder at the highest engine speed.

For our 4-cylinder, 600-cc engine, each cylinder processes 150 cc of air every two revolutions. At wide-open throttle, the cylinder fills completely with air at 1 atmosphere pressure. Air at this pressure has a density of about 1.3 kg/m^3. When the engine spins at its 12,000-RPM maximum speed, it is therefore pumping 20 g of air per second per cylinder.

The proper air-to-fuel mass ratio for maximum power in a gasoline engine is 12.6:1. Rounding that down to 10:1 provides a little extra flow to account for future engine improvements. The engine therefore requires 2.0 g/s of fuel.

Injectors should not be open continuously because they can heat up and operate erratically. So, assume

HCS II owners:

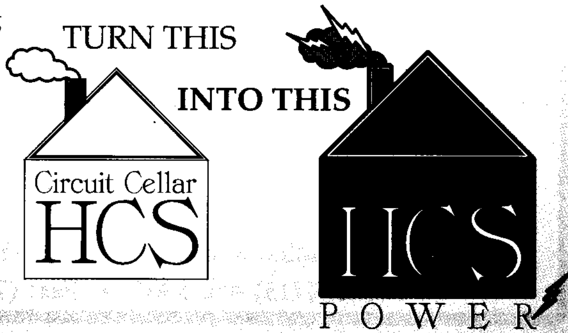
Here's that one little word you've been longing to hear...

ANNOUNCING THE HCS II VERSION 3

UPGRADE

Now you can upgrade your HCS II to Version 3.0 and add these exciting features:

➤ New HCS/PC Interface
 • Query and override any system input, output, or parameter from the PC
➤ Revamped HOST Program
 • Use a mouse to resize, move, open, and close all HOST windows
 • Send messages from XPRESS to a HOST window for debugging
 • Set or clear system inputs, outputs, or X-10 modules and try out speech strings from HOST
➤ Modem Support
 • Call your HCS from a remote location and check its status, load a new XPRESS program, or retrieve logged data
➤ Caller ID
 • Access Caller ID data from XPRESS to announce or log who's calling
➤ Support for more digital I/O expansion boards
➤ Read and write eight netbits at a time with Netbyte
➤ Send messages to network modules directly from XPRESS
➤ Detect loss of AC power from within XPRESS
➤ Plus lots more for just $60



Circuit Cellar, Inc., 4 Park St., Vernon, CT 06066
(203) 875-2751 • Fax: (203) 872-2204

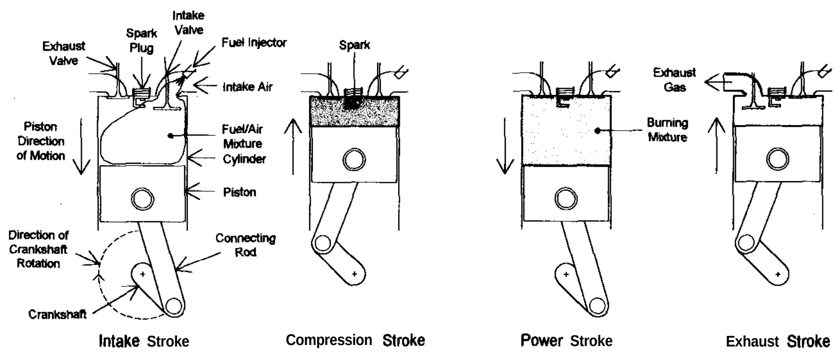
The Four-Stroke Engine

On the intake stroke, fuel and air are drawn into the cylinder. During the compression stroke, this mixture is compressed. Shortly before the piston reaches the top of the cylinder, the spark plug ignites the mixture.

The pressure caused by the burning mixture forces the piston downward on the power stroke. This turns the crankshaft and delivers power to the wheels. During the other strokes, momentum or another cylinder keeps the crankshaft turning.

The spent exhaust gas is pushed out of the cylinder on the exhaust stroke, and the cycle repeats.

The sequence can be conveniently remembered as "suck, squish, pop, ptooey."



NEW 8051 Family Development System

OEI552 *with Fuzzy Logic Support*

DX126 Control Board

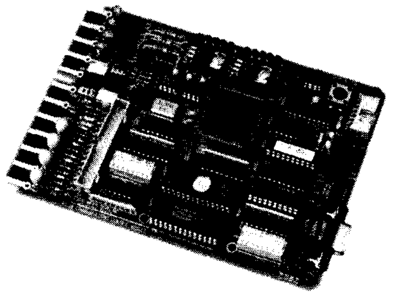
- Powerful 80C552 Miicrocontroller
- Memory up to 64K ROM/ 32K RAM
- AI/DI-DO/PWM Signal Conditioning
- Expansion I/O Header
- Configurable CODE/XDATA Overlays

Software Features

- OEI552 Monitor/Debugger
- FL1552 Fuzzy Logic Inferencing Engine
- DGU552 Diagnostic Package
- Fuzzy Logic Control Template
- Numerous Implementation Examples

Also Available

- Fuzzy Logic Development Tools
- Software Development Tools
- Hdw Accessories & Add-on Modules
- Sfw Drivers & Control Packages



This is Not a Toy!

The **OEI552 development package** is a member of OEI's powerful TRIACS™ family of industrial I/O control modules. OEI offers a full line of industrial control products and services utilizing both conventional and expert system technology.
Call today for more information.

OEI Ortech Engineering Incorporated

17000 El Camino Real, Suite #208 . Houston, TX 77058

Phone: (713) 480-8904 . Fax: (713) 480-8906

#107

that the injector operates at a maximum 80% duty cycle. Therefore, the fuel flow rate must be about 2.5 g/s. I chose commonly available automotive injectors rated at 19 lb./h @ 45 psi, which is 2.4 g/s. A larger injector could be used, but one that is too large has trouble metering small quantities of fuel and the engine cannot idle well.

The injector I chose is a saturated-circuit type, designed to be switched by a transistor directly to 12 V. The coil resistance is 12 Ω, so the drive current through the transistor is only about 1 A.

There are power Darlington drivers in a DIP package that can be used for driving this type of load. However, these may not have the power-dissipation capacity to handle all four injectors being on simultaneously.

For this reason, I chose discrete TIP120 power Darlingtons. They easily handle the current, are not damaged by the inductive-voltage kick when the injector closes, and require only minimal heatsinking (a 1" square area of copper on the board does just fine].

Figure 4 shows how simple the drive circuit is. Turning an injector on and off is a simple matter of turning on and off the 68HC16 output it's connected to.

ALGORITHMS AND SOFTWARE

Figure 5 shows the data members and messages for the Injector class. An Injector object receives the Injector.OpenO message from the Distributor object. The Injector object looks at current operating conditions and determines a correct pulse width. Having turned on the physical injector and calculated the pulse width, the Injector object tells the AlarmClock object to send it an Injector.CloseO message after the required time period elapses.

Note that the injectors are not directly controlled by a hardware timer since there are not enough output compares for individual injector actuation. Of the five available, two are used for the ignition coils and one by AlarmClock, both of which have more stringent timing requirements.

One solution pairs the injectors by driving two off the same transistor.

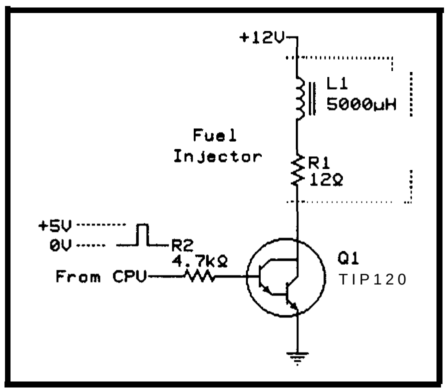


Figure *4-The injector* drive circuit makes *turning the* injector on *and off as simple as flipping a* processor *output bit.*

This solution, however, precludes adjusting individual injector pulse widths to account for differences in air flow between cylinders. It turns out that the off messages generally do get sent on time. Even if they are slightly off, they err on the late side, providing slightly more fuel than necessary, which is safer than feeding too little.

There are a few important things to keep in mind when creating a fueling algorithm. First of all, running an engine for too long with insufficient fuel can melt internal engine parts. “Too long” is dependent on the engine and conditions, but can be as little as several seconds. If an error in fueling is made, it must be the sort of error that doesn’t last very long or result in too much fuel.

Still, too much fuel causes power reduction and carbon fouling of the spark plugs. In the worst case, the injectors open and never shut off, filling the cylinder. Not only can this cause fuel to leak out of the engine, creating a fire hazard, but if the engine is spun, it tries to compress a cylinder full of liquid fuel, which breaks internal parts. If you are lucky enough not to break anything, you must still change the oil immediately since it will likely be full of fuel and useless as a lubricant.

My system was protected by a separate automatic shut-down circuit (to be described in *INK* 64), which prevents fuel overflowing through software glitches.

In this speed-density system, fuel required by the engine is a function of manifold pressure and RPM. Ideally, the engine is run on a dynamometer and all combinations of pressure and RPM are tested for each combination of pulse width that results in maximum power. If reasonable limits on quantization are made, say 32 steps between full vacuum and atmospheric pressure and O-12,000 RPM in increments of 500 RPM, one ends up with 32 x 24 = 768 points to test.

Such testing can take a lot of time. I knew from previous tests that at wide-open throttle (WOT), the injector pulse width is fairly constant, regardless of RPM. I therefore chose to base the injector pulse width solely on pressure. I recorded pulse width at WOT and used the ideal gas law to estimate pulse width at lower pressures.

A good initial guess at WOT pulse width can be made from the estimated air flow through the engine. Each cylinder draws in 0.2 g of air per intake stroke at WOT. As mentioned, maximum power occurs with an air-to-fuel mass (not volume) ratio of 12.6: 1. So, for each intake stroke, we need to inject 0.016 g of fuel.

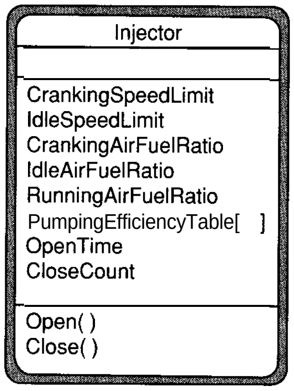


Figure *5-The* ***software*** *model for a mechanical fuel injector includes lots of data,* ***but*** *on/y two message functions.*

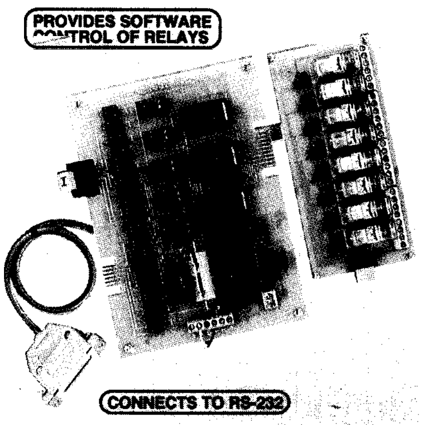
Note that one intake stroke occurs every two revolutions, and the injectors are programmed to fire once per revolution. We need two separate injections of 0.0080 g of fuel. The injectors have a flow rate of 2.4 g/s, so the flow time for 0.0080 g is 3.3 ms.

Since the injectors need 0.6 ms to open, a total pulse width of 3.9 ms is needed. As a quick check on injector size, note that one revolution at 12,000 RPM takes 5 ms. Thus, our injectors operate at a safe 3.9/5 = 78% duty cycle while providing the right amount of fuel at maximum RPM.

Part-throttle pulse width can be extrapolated knowing the manifold

RELAY
INTERFACE

(PROVIDES SOFTWARE CONTROL OF RELAYS)

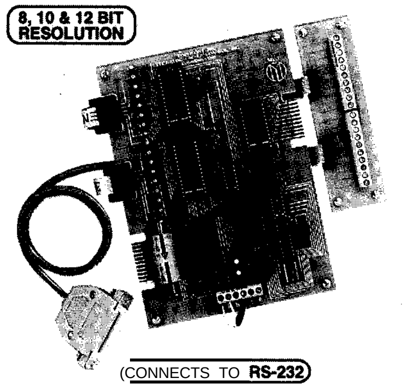


(CONNECTS TO RS-232)

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

ANALOG TO DIGITAL

(8, 10 & 12 BIT RESOLUTION)



(CONNECTS TO **RS-232**)

ADC-16 A/D CONVERTER’ (16 **channel/8 bit)..$ 99.95**
ADC-8G A/D CONVERTER’ (6 **channel/10 bit).$124.90**
Input voltage, amperage, pressure, energy usage, joysticks and a wide variety of other types of analog signals. **RS-422/RS-485** available (lengths to 4,000’). Call for info on other AA configurations and 12 bit **converters** (terminal block and cable sold separately).
ADC-8E TEMPERATURE INTERFACE’ (6 **ch)..$ 139.95**
Includes term. block **&** 6 temp. **sensors** (-40’ to 148’ F).
STA-6 DIGITAL INTERFACE’(6 **channel).........$ 99.95**
Input **on/off** status of relays, switches, HVAC equipment, security devices, smoke detectors, and other devices.
STA-8D TOUCH TONE INTERFACE’................ $ 134.96
Allows callers to select control functions from any phone.
PS-4 PORT SELECTOR (4 channels **RS-422)....$ 79.95**
Converts an RS-232 port into 4 selectable RS-422 ports.
CO-465 (RS-232 to **RS-422/RS-485 converter)......$ 44.95**

***EXPANDABLE...expand** your interface to control and monitor up to 512 relays, up to 578 digital inputs, up to 128 analog inputs or up to 128 temperature inputs using the PS-4. EX-18, ST-32 &AD-18 expansion cards.

FULL TECHNICAL **SUPPORT...provided** over the telephone by our staff. Technical reference&disk including test software&programming examples in Basic, C and assembly are provided with each order.

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

CONNECTS TO **RS-232,** RS-422 or **RS-485...use** with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud).

Use our 800 number to order FREE INFORMATION PACKET. Technical information (814) 484.4470.

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

International & Domestic FAX (814) 484-9658
Use for information, technical support & orders.

ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 804
Columbus, Ohio 43215.5438

#108

pressure. The pressure in the manifold is atmospheric (100 kPa) at WOT. As the throttle blade closes, it restricts the amount of air flowing into the intake manifold, reducing the manifold pressure.

Remember from high-school physics the ideal gas equation $PV = nRT$. From this equation, you can derive a formula that shows that for a given volume and temperature, the mass of air in the volume is directly proportional to the measured pressure.

As a reasonable approximation then, if the manifold pressure drops to half of what it is at WOT (i.e., 50 kPa), the mass of air entering the cylinders is also cut in half. The required fuel mass and injector pulse width is similarly cut in half. So, knowing the WOT injector pulse width, the pulse width for any operating condition is simply:

$$\text{WOT pulse width} \times \frac{\text{manifold pressure}}{100\ \text{kPa}}$$

At idle, when there is no need to produce maximum power, I can save fuel by using a different pulse width based on a 14.7: 1 air-to-fuel ratio, the ratio for best fuel economy. I also need a richer mixture during cranking to make starting easier. In the code, I therefore chose different base pulse widths depending on RPM.

The computed pulse width is multiplied by a factor of 0–2 depending on RPM. This factor is looked up in a pumping-efficiency table. The airflow through most engines varies somewhat at different engine speeds due to resonances in the manifolds and flow restrictions. This table allows the pulse width to correct for changes in airflow that occur at different RPMs.

`OpenTime` is added as the last step. The code first computes the total time that the output signal to the injector should stay high. It then turns the injector on and tells `AlarmClock` to send a turn-off message at the right time in the future. This command calls the `Injector.Close()` function at the appointed time. Figure 6 specifies the injector-pulse-width computation.

Under certain rare conditions, the possibility exists that the injector might not close before it needs to be opened again. This problem is most likely to happen at high RPMs at WOT. Theoretically, with a properly sized injector, this should never happen, but software latencies or an erroneous pulse-width calibration could cause it.

So, I implemented a counter that increments every time the injector is reopened, even if it is already open.

Inputs:

MAP	current manifold pressure
RPM	current engine speed

Calibrations:

CrankingSpeedLimit	RPM below which the engine is considered to be cranking
IdleSpeedLimit	RPM below which the engine is considered to be idling
CrankingAirFuelRatio	constant that determines air/fuel ratio at cranking speeds
IdleAirFuelRatio	constant that determines air/fuel ratio at idle speeds
RunningAirFuelRatio	constant that determines the air/fuel ratio at running speeds
PumpingEfficiencyTable	table of pumping efficiencies at different RPMs
OpenTime	time it takes for injector to open

outputs:

PulseWidth	length of time the injector must be open over the next engine revolution

Equations:

```
if (RPM <= CrankingSpeedLimit)
  AirFuelRatio = CrankingAirFuelRatio
else if (RPM <= IdleSpeedLimit)
  AirFuelRatio = IdleAirFuelRatio
else
  AirFuelRatio = RunningAirFuelRatio

PulseWidth = AirFuelRatio x MAP x PumpingEfficiency[RPM]+OpenTime
```

Figure 6—*This* algorithm runs once per revolution for each injector to determine how much fuel should be injected.

The counter represents the number of extant `Injector.Close()` messages waiting to be processed for that injector. Each call to `Injector.Close()` decrements the counter, and only when it reaches zero is the injector actually turned off.

When pulses overlap in this fashion, the later pulses are shortchanged by this scheme since the total amount of injected fuel is less by the amount of overlap than the sum of the individually scheduled pulses. This is not a condition to be concerned about since the open times for each pulse become full-flow times since the injector is already open. Also, every other revolution, all the fuel is flushed through the cylinder, so there is no point in trying to inject a "past due" amount of fuel.

Since this condition happens rarely and lasts only briefly, it does not harm the engine.

CONCLUSION

The development of electronic fuel injection paved the way for increased vehicle performance while meeting strict emission and fuel-economy requirements. It made our race car more powerful and easier to drive. It's a far cry from the earliest attempts at mixing fuel and air, one of my favorites being the use of a wick just like a kerosene lamp!

Next month, I'll show you how to generate carefully timed 35-kV sparks without frying any electronics or sending the CPU out to lunch with RF noise. ❑

Ed Lansinger is a computer and systems engineer who worked on the Cadillac Northstar powertrain control software, cofounded an industrial software company, and does consulting. He has returned to Rensselaer Polytechnic Institute for graduate studies and is forming a team there to build an electric race car. He may be reached at lansie@rpi.edu.

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

Embedded Debugging Tricks

Stuart Ball

When designing an embedded controller, the ability to get feedback from the system during debugging is usually limited. Drawing from years of experience, Stuart reveals his tricks for getting rid of pesky bugs.

A problem we've all encountered in working with embedded microprocessors is debugging the pesky things.

Our equipment sometimes must be debugged at a customer site without affecting customer operations. This situation often precludes the use of an emulator where the processor stops when a breakpoint is detected.

In some cases, no emulator is available for the processor. Consequently, I cannot always depend on an emulator.

It is also critical that the debug data be generated from the normal operating code. I cannot use a method requiring a handshake with a host PC or terminal. The system can't behave one way when a debugging tool is connected and another when it is not.

Over my fifteen years as an electrical engineer working with embedded microcontrollers, microprocessors, and DSPs, I've developed a standard debugging technique I'd like to share with you.

MY TECHNIQUE

The basic debugging system I have developed makes use of a standard 10-pin connector (see Figure 1). This connector contains a ground, write strobe (• STB), and eight data lines (DO-D7). I usually implement the connector as an inline 10-pin header. The microprocessor being debugged contains in firmware instructions that write specific data to the test connector so it can be captured and analyzed. Figure 2 shows the timing of the write operation.

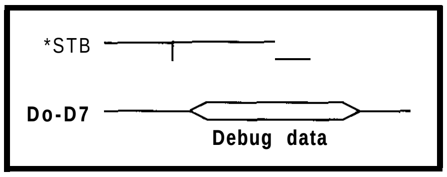


Figure 2—*With the test-connector signal timing, the only requirement is that the data be stable on the rising edge of *STB.*

For example, Table 1 offers a partial list of values (in hex) that I used on a recent character-recognition project, implemented with a DSP. Each time the DSP enters the A/D converter's sampling routine, it outputs a value of 02 to the test port. A detection of a character edge produces 03.

Using a logic analyzer (in state mode) connected to the standard test connector, I can trace the history of program operation by time tagging the data. Of course, with the proper interface, the data could just as easily be transmitted to and logged on a PC.

On this DSP system (see Figure 3), the write strobe for the standard test connector is just another address decode line from the PLD already on the board for the other DSP peripherals. The data lines on the standard test connector are the DSP data lines. On a Z80 or x86 family processor, the write strobe can be a spare memory or I/O decode. On a Motorola processor without a separate I/O space, the strobe must be a memory decode.

1	2	3	4	5	6	7	8	9	10
Gnd	*STB	DO	D1	D2	D3	D4	D5	D6	D7

Figure l--By *including a standard test connector in every new design, it's possible to develop a common set of test fools that can be used on all designs.*

This technique works fine if you have decoded read/write strobes for peripherals connected to the CPU. However, I don't always have the luxury of a spare I/O line, and sometimes I don't have any I/O decoding at all. Nevertheless, I still like to use the standard connector for debugging.

Figure 4 shows a generic debugging system. The microprocessor in the system under test outputs test values to a debug output circuit. They are translated to the standard output connector format and can then be captured by a logic analyzer or transmitted to a PC.

On systems, such as this DSP system, the debug output circuit merely connects the data lines and decoded strobe to the test connector. On others, the circuit is more complex, requiring approaches like the ones below.

WRITE TO ROM

Most micros use an external ROM or EPROM. Normally, there is nothing else residing in the EPROM address space. If a write is executed to the EPROM space, the data won't go anywhere, but it can still be captured with the simple address decoding logic of Figure 5. This circuit generates the ● STB signal with a 74LSO0. The ● CE signal to the EPROM (pin 20 on a 27256) is gated with the *WR signal. The test connector data lines connect to those of the microprocessor.

This method works with any micro that has an external EPROM and can write to the EPROM address space. Some micros require more logic to decode the strobe.

READ EPROM ADDRESS

On 805 1 systems with external EPROM, code and data spaces are often separate, so you can't write to the EPROM. However, they can read data from the EPROM. The circuit in Figure 6 is based on the fact that most 8051 systems (at least most of mine) don't use the entire 64-KB space. Any access to the upper 32 KB (Al5 = 1) generates the ● STB signal. In this case, the debug data comes from the microprocessor address lines. Listing 1 includes the code which implements this on an 805 1.

The value read from the EPROM space is unused and is discarded. If you need more than 32 KB of EPROM, use a wider NAND gate [such as a 74LS30) and decode the address lines to generate ● STB when A8–A15 are all ones. This technique uses the upper 256 bytes (FFOO-FFFF) for the debug space, leaving the remainder for code.

Of course, this method works for most micros with an external EPROM.

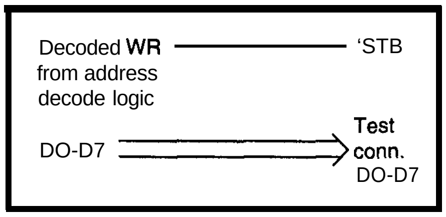


Figure 3—The debugger uses generic connections from a microprocessor to the standard test connector.

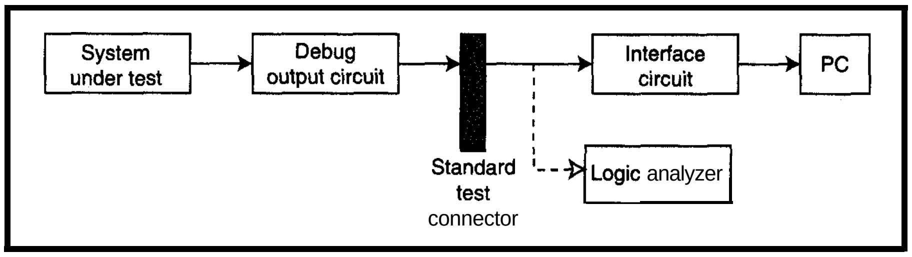


Figure 4-The form the debug output circuit takes depends on the system under test, but everything from the test connector on can be identical no matter what system is being checked.

Value	Meaning
01	Powerup reset processing complete
02	ADC sampling interrupt entry
03	Character edge detected
04	Blanking on
1x	Character recognized; x = char code (O-F)
2x	Host command received; x = command code (O-7)
3x	Recognition occurred when blanking; x = char code (O-F)

Table I--Data values sent to the debug output represent various program states to allow tracing.

SYNCHRONOUS SERIAL OUTPUT

Some 8051 systems use internal EPROM or ROM. On these systems, generating debug data with the 8051 serial output connected to the COM port of a PC is common. The circuit in Figure 7, however, uses the 8051 serial output in synchronous mode.

This method provides two advantages over the asynchronous mode:

- you can use a standard test connector
- the maximum rate of debug output data increases

When programmed for synchronous mode, the 8051 outputs data at 1 Mbps with a 12-MHz input clock. Still, this method limits the maximum output data rate. The output code for this method is:

```
MOV SBUF,#CONSTANT
```

This debug method has the advantage that no CPU registers are required to output the data. The output instruction can be put anywhere in the code without having to verify that a needed register hasn't changed.

However, it is possible to output data to the serial port faster than it can be transmitted, which results in garbage values. When I use this method, I don't put the debug circuit on the 805 1 board. Instead, I use a separate breadboard and connect it to the 8051 via an IC clip.

USING DEBUG DATA

Generating debug data is useless if you can't do anything with it.

Figure 8 shows a FIFO connected to a PC parallel printer port. The FIFO is written with data from the standard

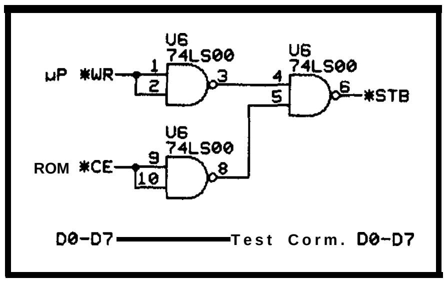


Figure 5—One alternative for generating *STB gates the ROM chip select signal (*CE) with the microprocessor *WR signal.

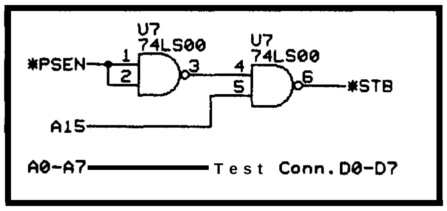


Figure 6—Accesses to ROM above 8000h are decoded and gated with the microprocessor read signal (*PSEN) to generate *STB. Debug data uses microprocessor address fines.

test connector. When the FIFO has data, the *ACK signal to the PC is driven low. When the PC detects debug data, it reverses the direction of the port (you must use a bidirectional port), then drives *STB low to read it.

The specific code for this depends on your PC. However, you can use any other input and output control lines for ● ACK and *STB. If your cable from the FIFO board to the PC is more than a foot long, it's wise to buffer the data lines with a bus driver IC (such as a 74LS244). This circuit needs a reset for the FIFO, which can be connected either to the *INIT signal on the printer port or to the circuit under test.

Another way to send data to a PC is to connect an 8031 to the FIFO, and transmit the data serially to the PC via the 803 1 serial output. Although I've never used this technique, it lets you collect data with a standard communication program such as Procomm.

While the FIFO permits debug data to be sent faster than the serial data rate supports, the average debug data rate cannot exceed the serial data

Listing I--When used *with the* circuit in Figure *6, the 8051 assembly code necessary to generate a read-from-ROM strobe adds little overhead.*

```
MOV  DPTR,#8000H
MOV  A,#CONSTANT       ; CONSTANT is the debug value
MOVC A,@A+DPTR         ; Read the EPROM, generate strobe
```

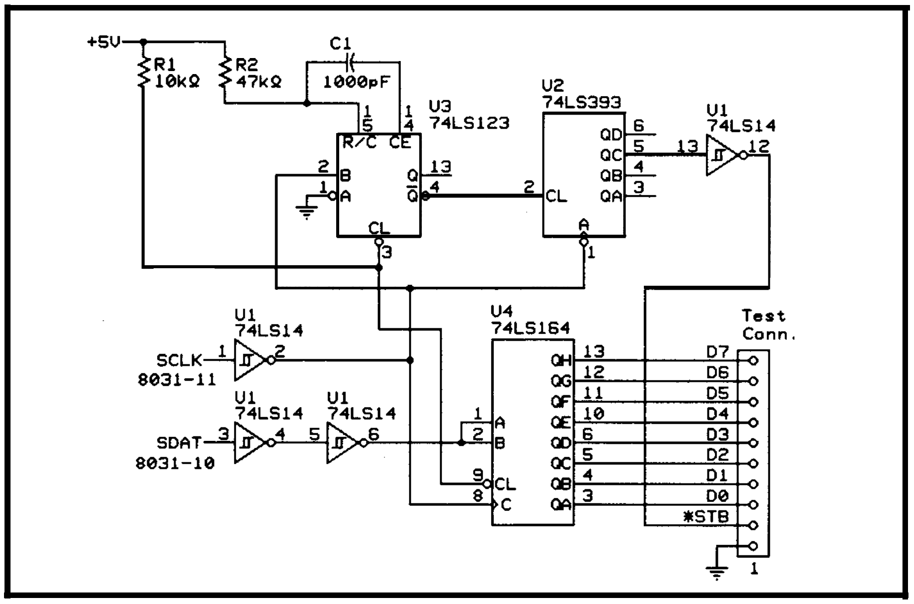
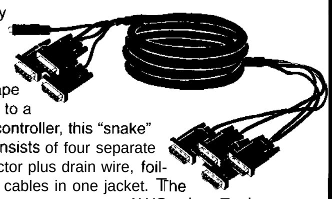


Figure *7-The 8051's serial port in synchronous mode can be used to generate the debug data and ● STB. The shift register captures serial data from the 8051, and the counter generates *STB when 8 bits are received.*

ALL ELECTRONICS CORP.

QUALITY PARTS • DISCOUNT PRICES • FAST SERVICE • HUGE SELECTION

INCREDIBLE SAVINGS!
30' CABLE



Originally made to connect Digital Audio Tape machine to a remote controller, this "snake" cable consists of four separate 9 conductor plus drain wire, foil-shielded cables in one jacket. The conductors are stranded 24 AWG wire. Each end of each smaller cable is terminated with DB-9P connectors. The cable is well-made and quite flexible for its size. Snake cable nominal O D 'is 0.52" Interior cable O.D. is 0.15". The outer jacket can be slit and removed if only the 9 conductor cable is required DB-9 connectors Include thumbscrew hold-downs If you are using multiconductor shielded cable this is a great deal.

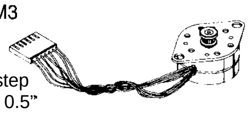
$5.50 each

CAT# CBL-3

2 for $10.00

SMALL STEPPING MOTOR

Airpax # NB82115-M3
12 Vdc, 65 ohm stepper motor with 6 leads. 15 degree step angle. 1" diameter x 0.5" high. Mounting flange with holes on 1.25" centers. 0.08" (2mm) dia. X 0.2" long shaft. 0.27" dia. pulley is press-it onto shaft. Great for robotics and experimentation.



$1.50 each

CAT# SMT-13

10 for $12.50

6 INCH IEC POWER CORD

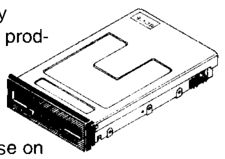
That's right! 6 inch...Standard 3 prong grounded male one end and 3 prong IEC receptacle on other end. CSA listed. Large quantity available.

2 for $1.00 CAT # LCAC-36

10 for $4.50 • 100 for $35.00

720K 3.5" INTERNAL FLOPPY DRIVE

Sony # MP-F11W-2Z
Brand new 720 KB floppy drives. Built for an OEM product, these are, in every way, compatible with PC/XT/AT computers except that the faceplate is slightly larger than those on exact replacements. The faceplate size on these is 4.04" X 1.23". (Normal size is 4" X 1"). The faceplate is removable and can be trimmed if desired. In some cases it may work without trimming. Also, we have found that, when used in 5.25" bays with the adapter kit below, the adapter can easily be modified to accommodate the unit. Because of this slight inconvenience, we are selling these drives at an incredibly low price.



$14.00 each

CAT# MPF-11SF

10 for $120.00

INSTALLATION KIT for MPF-11SF allows for installation into 5.25" drive bays. Simply cut and remove three plastic gussets to Install.
CAT# FDD-3.5 $3.50 each

CALL, WRITE or FAX for a FREE 64 Page CATALOG
Outside the U.S.A. send $2.00 postage.

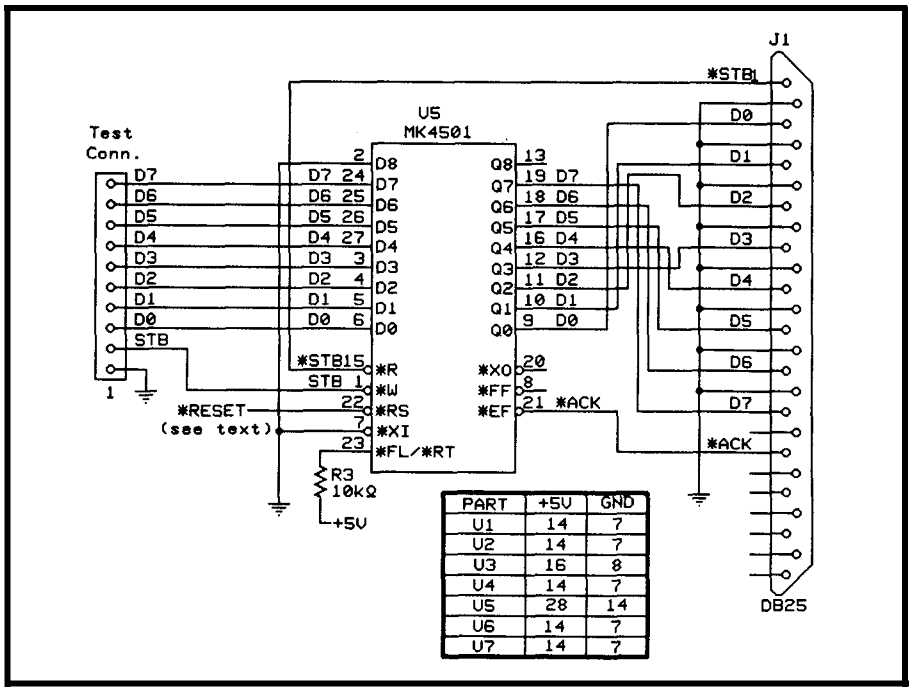
MAIL ORDERS TO:
ALL ELECTRONICS CORPORATION
P.O. Box 567
Van Nuys, CA 91408
FAX (818)781-2653
E-Mail - allcorp@allcorp.com

ORDER TOLL FREE
1-800-826-5432
CHARGE ORDERS to Visa, Mastercard or Discover

TERMS: *NO MINIMUM ORDER*. Shipping and handling for the 48 continental U.S.A. $5.00 per order. All others including AK, HI, PR or Canada must pay full shipping. All orders delivered in CALIFORNIA must include local state sales tax. Quantities Limited. NO COD. Prices subject to change without notice.

Visit Our World Wide Web Site...
http://www.allcorp.com/allcorp/

#109



PART	+5V	GND
U1	14	7
U2	14	7
U3	16	8
U4	14	7
U5	28	14
U6	14	7
U7	14	7

Figure *8-A* ***FIFO*** *can be used as a buffer when connecting the circuit under test to a PC's bidirectional parallel printer port.*

rate or the FIFO overflows. You could do the same thing with a discrete UART and some control logic, but the 803 1 approach is probably simpler.

Of course, all of the logic shown here as 74LSxx could be implemented with HC, AC/ACT, or as equations in a PLD, the method I prefer.

MISSION ACCOMPLISHED

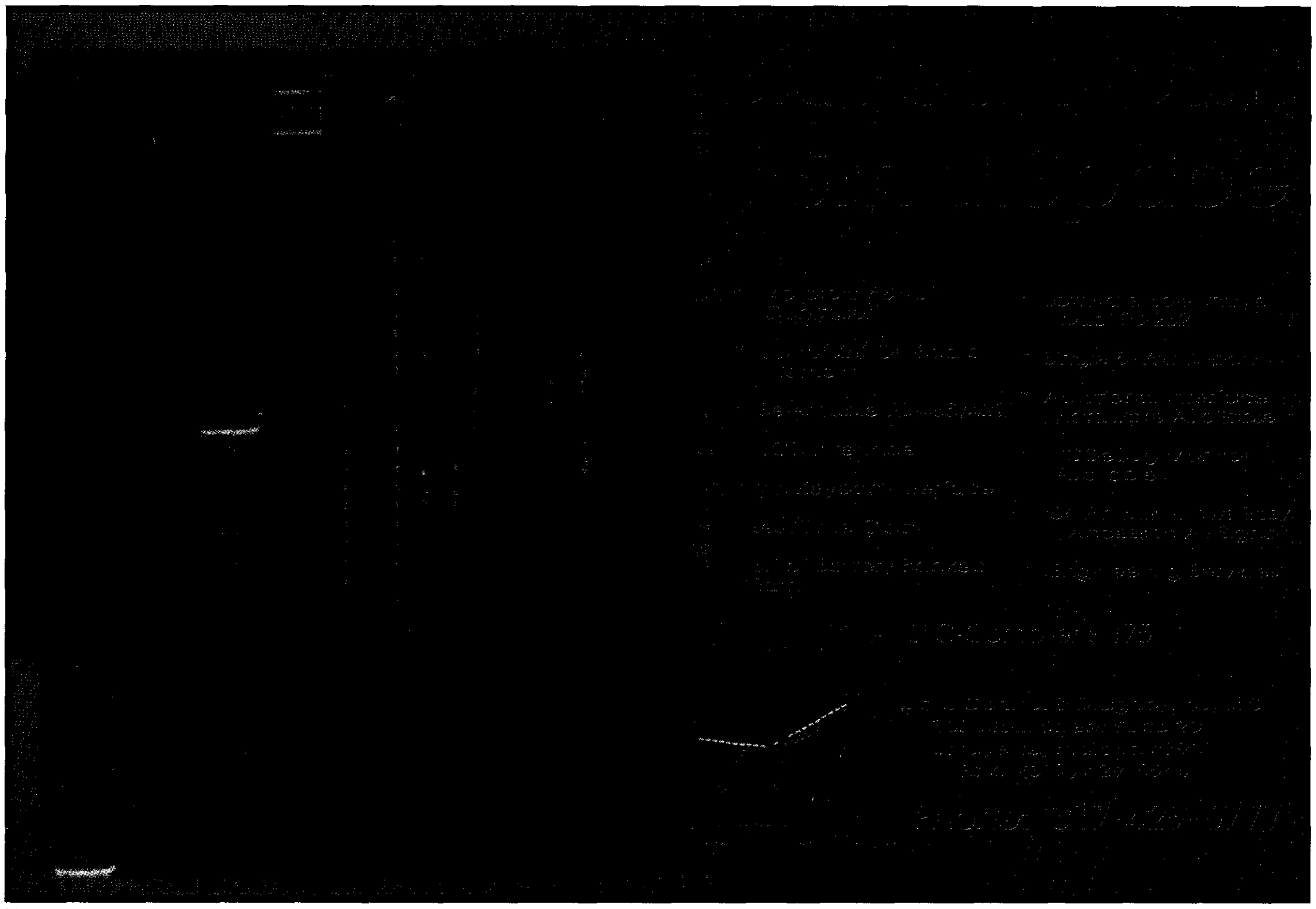
This standard debug connection works with almost any microprocessor or DSP system. It gives a window into what the processor is doing with minimal external connections. It can even be used by factor-test technicians in isolating faults.

It has saved me enormous time in integrating and debugging circuits. I hope it does the same for you. ❑

Stuart Ball has spent the last 15 years working on systems as diverse as Global Positioning System and single-chip interface translators. He is currently employed as a principal engineer at BancTec Technologies, a manufacturer of document processing equipment for the banking industry. He may be reached at (405) 354-5042.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful



#110

Designing an Industrial-grade XYZ Router Table

FEATURE ARTICLE

Gordon Dick

Gordon walks us through a start-to-finish project for a computer-controlled 3D router table. He deals with both the mechanics and electronics necessary for controling router movement.

I was originally introduced to intelligent motion control at work. Its advantages made me crave it for my own woodworking projects. I decided to develop a system of my own that was as affordable and capable as the expensive commercial units.

Aside from the satisfaction derived from building things, my ultimate goal was entrepreneurial. I was approaching this project as a part-time business venture and was concerned about keeping costs under control.

With these guidelines in mind, I began my largest project ever.

MECHANICAL ENGINEERING AND CONSTRUCTION

From brochures, prices, and specs, **two** trends were evident:

- rotational-to-translational conversion uses a rack-and-pinion gear
- motion stages use bearings on a hardened steel rail.

However, for my application, these options were too expensive or not practical since I didn't have a lathe or milling machine.

I started with a pulley-cable approach. Even though the design looked good on paper, I abandoned it since it was going to be quite difficult to build. I gave the toothed belt and pulley scheme serious consideration, but I had occasion to see an example of an XY table using this approach. It required significant speed reduction, which would have made construction more difficult.

At this point, a lead-screw system looked attractive. The price of a commercial lead screw and nut was prohibitive. However, I discovered that general-purpose thread rod is available cheaply in **12'** lengths. Even though these rods aren't hardened, I decided to use them and replace them as needed. Since the threads on thread-rod are rolled on, significant backlash occurs in a combination of thread-rod lead screw and nut. To eliminate the backlash, I used a double nut scheme as shown in Figure **1.**

Having the motors and their spec sheet, I knew the armature moment of inertia. I picked a lead-screw diameter to suit this moment of inertia. Since the lead screw and nut mechanical advantage is so large, the movement of the translational mass reflected back down the motor shaft as a moment of inertia is extremely small. Hence, it has very little effect on the performance of the motor. However, since the lead screw attaches directly to the

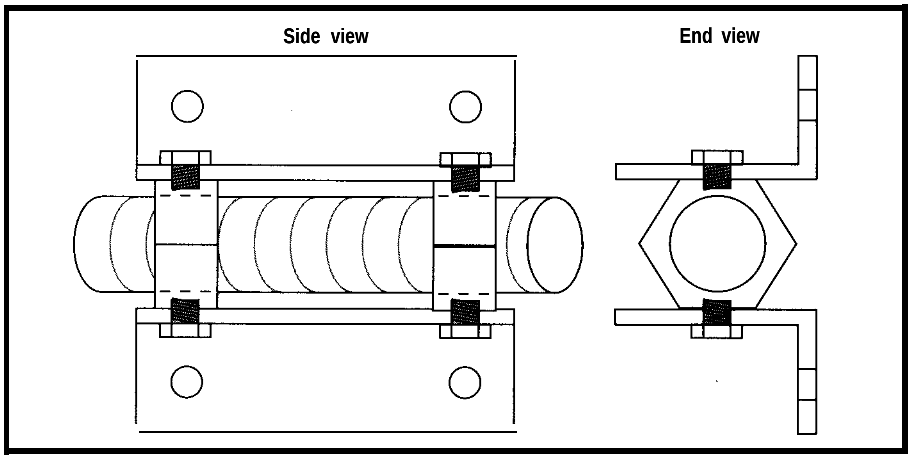


Figure 1—*The* **key to** *table motion is fixed nuts that move along each axis of motion as the lead screw onto which they* **are threaded rotates.**

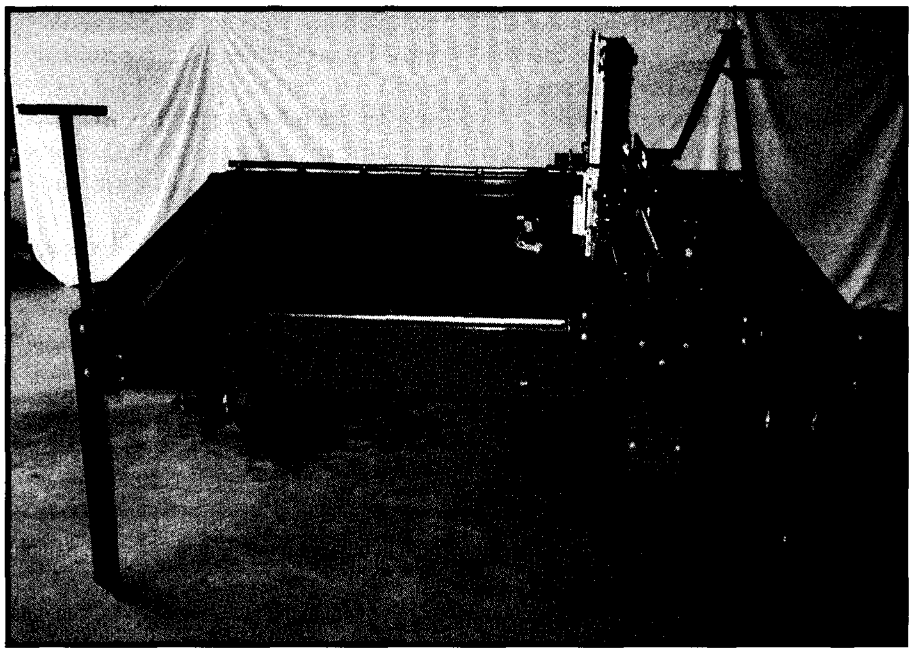


Photo I--Work *proceeded from mechanical construction to painting to final assembly. The x-axis motor cannot be seen here since if is under the work fable.*

motor shaft, its moment of inertia adds directly to the armature. Based on this rationale, I selected a 0.75" NC lead screw (10 threads per inch).

Figure 2 illustrates how I designed the motion stages. Although this design requires some machining, it's simple and repetitive since the stub shafts are identical for x and y motion stages.

I also needed a simple method of holding a work piece in position on the work surface. Commercial machines use techniques (e.g., a vacuum) too difficult to fabricate or too expensive. I used a simple grid of holes containing T nuts in a replaceable work surface. A wooden clamp fastened by a bolt into the T nut restrains the work piece and keeps metal away from the cutting area. To provide more versatility, the table adjusts up and down with jacking bolts.

Since the machine may have to be moved at some time, I wanted the mechanical design to be portable. This machine is made with subassemblies; each fits through a standard door opening and most can be carried by one person.

I spent considerable time producing detailed drawings for all parts and assemblies. This was time well spent! Machine fabrication proceeded glitch-free. Photo 1 shows the results.

THE ELECTRONICS

Since at work I had become familiar with the features and instruction of the Galil DMC600 intelligent motion controller series, it made sense to use it in this project. A variety of companies make intelligent motion controllers, so prices and features are competitive. Some controllers install in a PC expansion slot and some are standalone, communicating with a PC over a serial link.

An intelligent motion controller solves the hardest part of system design. Design focus shifts from worrying about getting the system to work to getting the system stable. The intelligent motion controller provides adjustment coefficients in the control algorithm to stabilize a wide range of mechanical systems.

The electronic portion of an intelligent motion control system demanded that I make choices between stepper motors and servo motors, stand-alone and expansion-slot, and so on.

Traditionally, stepper motors interface with a computer for applications requiring computer position control. However, at the time, I couldn't find an intelligent motion-control card for a stepper motor which provided coordinated motion (i.e., the x- and y-axis speeds are altered so that a move from one point to another is executed via a straight line between the two points).

In contrast, all intelligent motion-control cards for DC servos provided this feature. Since my application involved cutting with a router and not just simple positioning, coordinated motion was essential. (In a design incorporating new components, you also need to choose between brushless and brush DC servos.)

The choice to use stand-alone or an expansion slot was easy. Stand-alone models are intended for applications where one computer controls several motion-control tasks, communicating serially with each controller. Expansion-slot models tie a PC to a single activity since the intelligent motion controller gets its power from the PC. For this reason, expansion-slot models are cheaper. Since my application needed only a single motion-

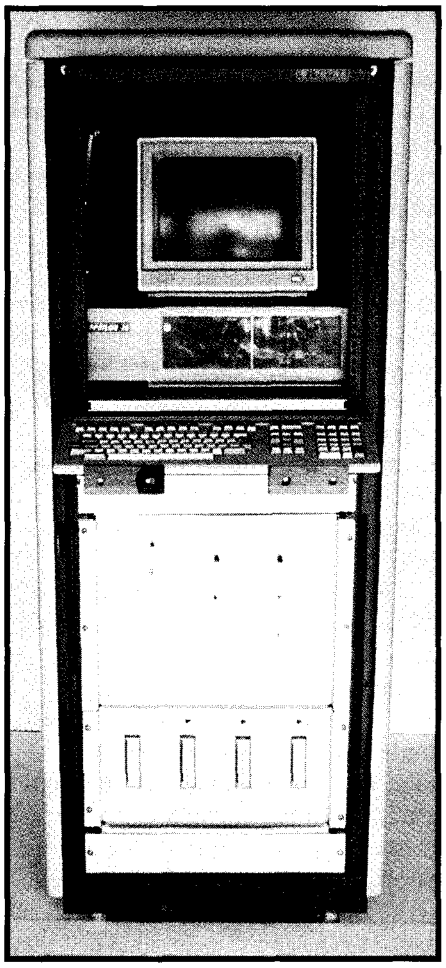


Photo 2—*The original tape-drive cabinet becomes the* **xyz fable control center. The PC keyboard is on** *a fray which s/ides info fhe cabinet when if's not in use.*

control activity, the expansion-slot model was sufficient.

Figure 3 shows the major components of an intelligent motion-control system. As you can see, a servo amplifier controls the DC servo motors. You can choose PWM, linear, transvoltage, and transconductance. PWM servo amps are better for small size, low cost, and high efficiency. Transconductance servo amps, on the other hand, make the system easier to stabilize and are therefore more popular. But, the choice can't really be made until the dynamics of the mechanical system are known.

Since I wanted to build my machine at the lowest possible cost, I decided to salvage DC servo motors, servo amplifiers, and power supplies from a mainframe tape drive. The unit, an attractive cabinet on casters, contained two 50 oz.-in./A motors and servo amps. In addition to a raw ±28- and ±55-V supply for the servo amps, a ±15- and +5-V supply was also present.

Of course, you could build the servo amps from scratch. Since audio power amplifiers and servo amplifiers are very similar, you could use an audio power amp design for your servo amp, but it would have to be a direct-coupled audio design.

Using the surplus equipment meant the *x* and *y* servos were already defined. Generally, DC servos are sized according to how rapidly the load will be accelerated or decelerated. If servo size is already determined, you are restricted to whatever the servos can deliver. Fortunately, the servos I salvaged matched my application reasonably well.

Since DC servos do not have the inherent digital positioning of stepper motors, some means of monitoring position must be included. Many rotational and translational position transducers can be used, however rotational incremental optical encoders are most popular because of their low cost, good resolution, and uncomplicated computer interface. Various styles of encoders match different applications and therefore have different prices. I don't have room to discuss the various styles, but some features need to be discussed.

Many encoders send their signals back to the motion controller differentially. Although this involves one more wire, this feature provides superior performance in a potentially noisy environment. Also, many encoders provide a one-pulse-per-revolution marker, which is used by some controllers as a homing sequence. If your application requires the machine to accurately position itself at a known starting position, then the marker pulse is important.

You must also choose the encoder for its resolution. This is a double-edged choice since an encoder with more lines provides finer positioning resolution, but in a closed-loop control system, there's less stability. In fact, doubling encoder resolution is equivalent to doubling the control system's loop gain.

If encoder resolution influences stability, you'd think there'd be a rule of thumb for deciding which resolution to use. However, my experience indicates there's no sure way.

With a control-system modeling package, I constructed a model. Information from data sheets and

Let Your Development Fly!

Choose Hitex professional tools for your embedded microprocessor design and get your project off the ground ahead of schedule. Hitex builds all types of microprocessor development tools, from sophisticated in-circuit emulators to remote debuggers, monitors and simulators. Complete solutions are available for:

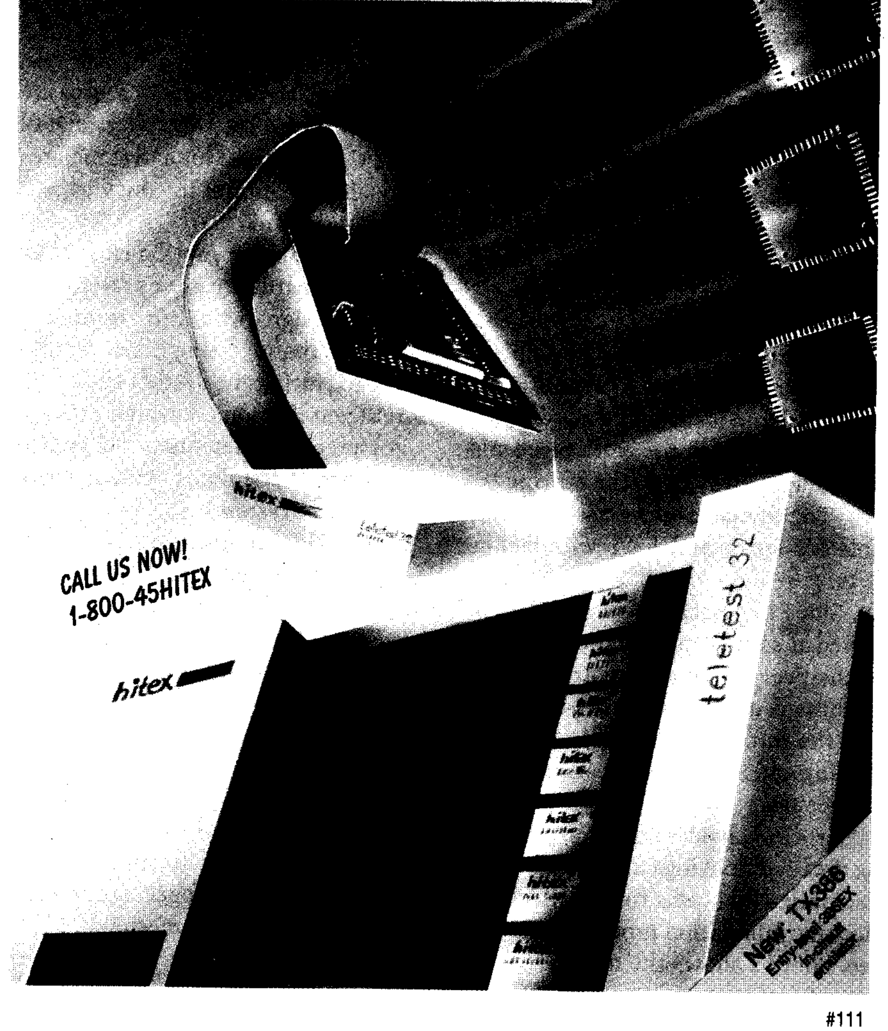
✔ the complete 8051 family
✔ 80C86/88, 80C186/188EA/EB/EC/XL, 80286, V20/V30/V40/V50
✔ 80386DX/SX/CX/EX
✔ 80C165/166/167
✔ 68HC11 family
✔ 6800x, 6830x, 6833x, 68340

Call Hitex for your free demo package and learn how smooth and efficient development with professional tools really can be!

HITOOLS Inc.
2055 Gateway Place
Suite 400
San Jose, CA 95110
☎ (408) 451-3986
☎ 1-800-45HITEX
Fax: (408) 441-9486

hitex

© Copyright 1994 Hitex. Hitex is a registered trademark of Hitex · All other trademarks are acknowledged to their respective owners.



#111

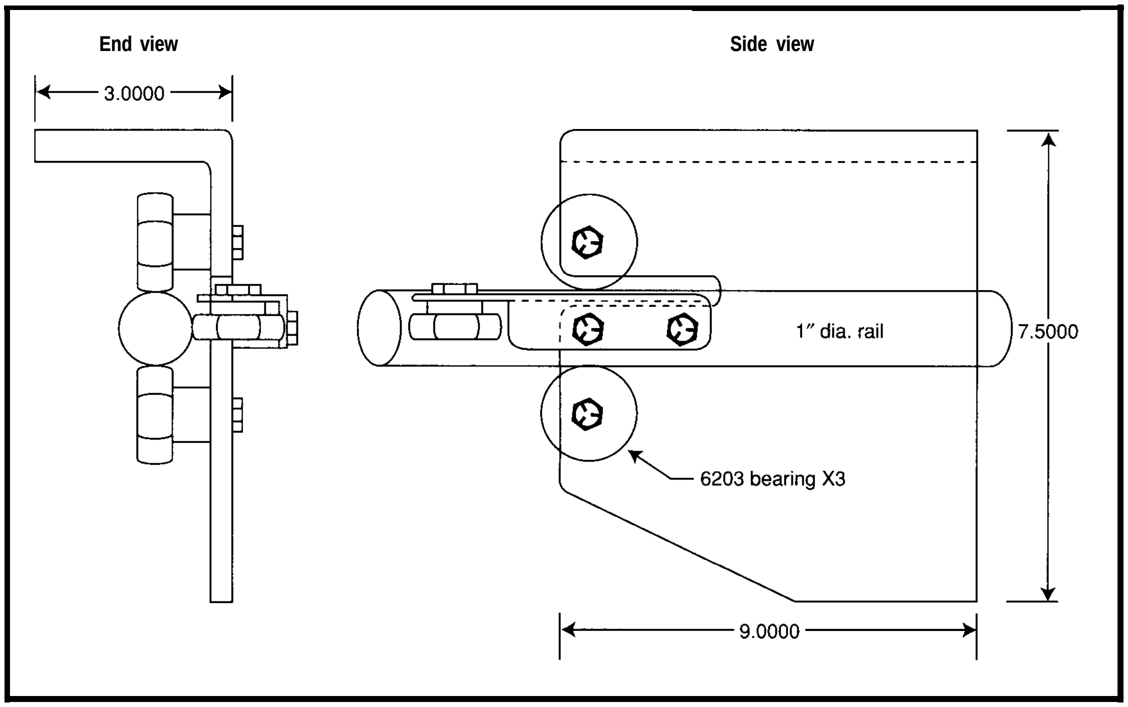


Figure 2—*Off-the-shelf bearings and shafting* **combine** *to* **make** *a simple motion stage. On/y one* **half of** *fhe stage is* **shown** *in the* **side view since** *if is symmetrical about the right-hand side.*

measurements on the system determined the mass in motion. I then constructed transfer functions for parts of the system. These individual transfer functions are submitted to the modeling package, which can then do simulated "loop gain" frequency-response plots and simulated step-response plots.

The modeled data, however, conflicted with measured data. The model predicted trends correctly but could not obtain quantitative results. While discrepancies could stem from many factors and certainly don't mean the system can't be modeled, it does show is that even with analysis tools, constructing a model which accurately mimics a real system is difficult.

So, don't choose an encoder resolution that exceeds your requirements. Instead, choose the lowest resolution that meets your needs. Anticipate stability problems. Hopefully, your controller's algorithm offers some fine-tuning to stabilize the system. But, to make your system stable, be prepared to build a compensator, change to a lower-resolution encoder, or both.

Many intelligent motion controllers employ an encoder resolution-multiplication technique. In the DMC600 series, encoder resolution is **multiplied** by a factor of four and the position units are referred to as *quadrature counts.* This means a 250-PPR encoder behaves as though it were a 1000-PPR encoder!

In choosing encode resolution, you also need to consider the total number of pulses produced when the machine moves from end to end. The number of pulses is determined by the lead-screw pitch, travel length, and encoder resolution. The intelligent motion controller establishes the maximum value of encoder pulses before rollover occurs. (For the DMC600 series, the number is 8,388,607 quadrature counts.)

Limit switches are activated when a motion stage gets close to reaching a mechanical "end of travel" limit. This feature is essential if the motors have

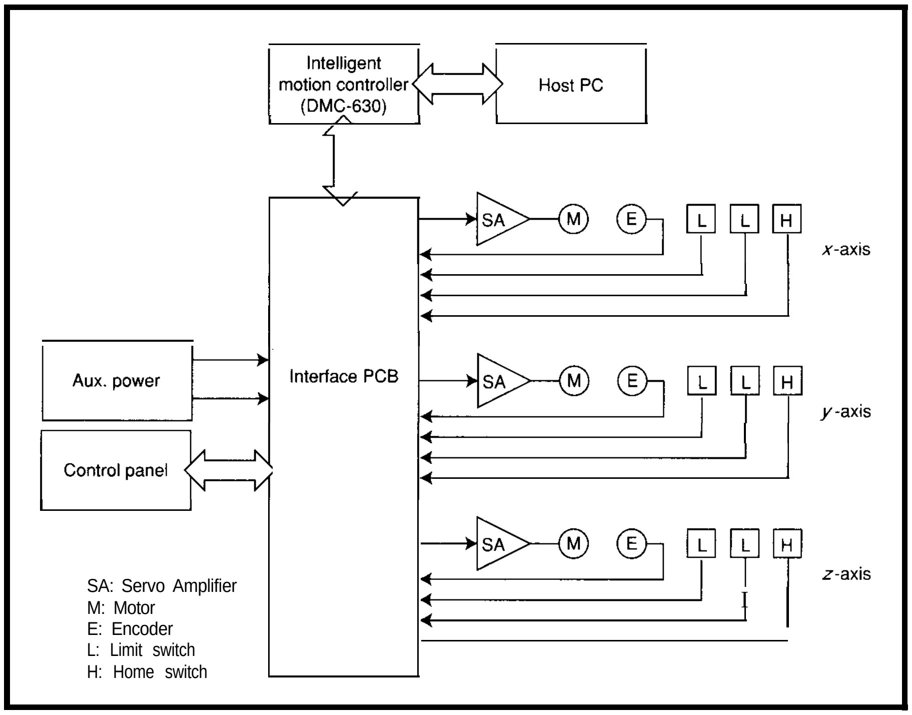


Figure 3—*Most* **intelligent motion control** *systems* **require** *the* **elements shown** *here. As you can see, the* **interface** *PCB plays a central role as a collector and distributor of signals.*

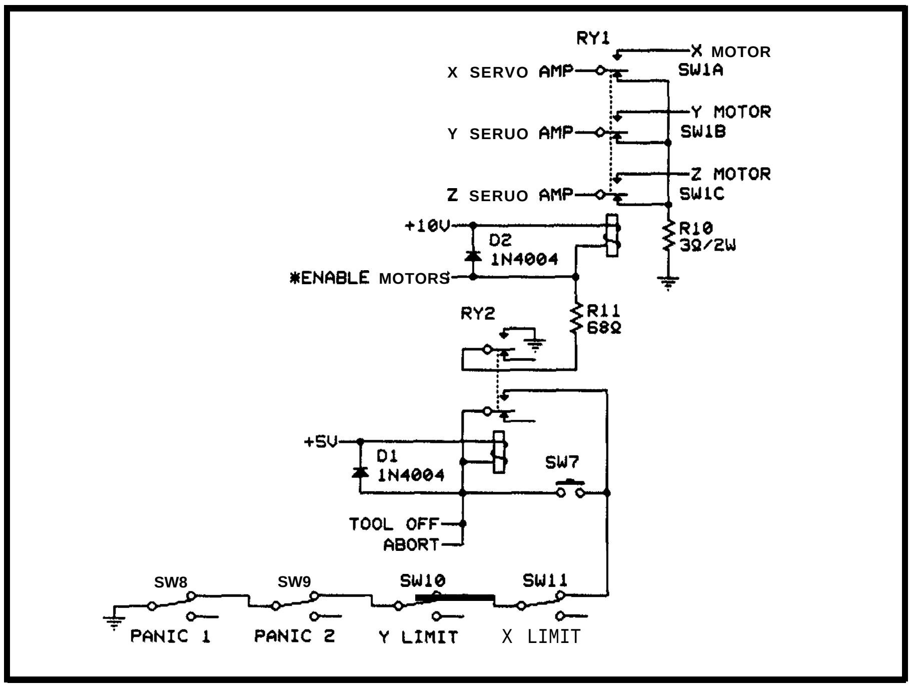


Figure *4-Safety features are important. Any tripped limit* or *panic* *switch stops the machine immediately.*

sufficient torque to do mechanical damage. Most controllers have a provision for acting on limit switch signals.

Figure 4 shows emergency shut-down and panic switches. It's essential that the machine can be shut down quickly in the event that something unexpected begins to happen. Rather than have users type a Stop or Halt command, they can stop the machine by pressing a large red panic switch located near the machine operating position.

Most controllers have logic-level inputs dedicated to an emergency stop function, but I chose to have the panic switch deenergize a relay whose contacts break the connection between the servo amps and the motor. A relay in the tape-drive power supply filled this purpose. You can see in Figure 4 that the limit switches and the panic switch are in a series. When any switch opens, the motor drive is lost.

The interface PCB is a circuit board whose main function is to distribute signals from the controller and gather signals for the controller. In my system, it does some simple logic and relay driving as well. RY2 in Figures 4 and 5 is configured to latch when the start switch is pressed as long as *all* the limit and stop switches are closed.

A second set of contacts on RY2 energize the coil of relay RY 1 (Figure 4), which connects the x and y servo amp outputs to their respective motors. As long as RY2 stays latched, the output of Ula (Figure 5) stays high and the abort input is not asserted.

Abort is a software panic stop. Ula, d, and c provide some simple logic to turn the tool off under panic conditions. Q2 operates a solid-state relay (not shown), which provides AC power to the tool. Q3 and Q4 energize auxiliary relays, which also switch AC power. One switches a dust collector on and off, and the other is a spare.

The last major design problem involved connecting cables to moving equipment. Commercial units typically deal with this problem by using retractile cables on small machines and a hinged cable tray on larger machines. I decided not to use retractile cables because I personally dislike them and am concerned that constant flexing might cause problems later. The hinged cable tray arrangement was too expensive to buy and too time consuming to build.

Some experimenting with a rope for a cable led to the approach shown in Figure 6. Here, the y-axis motion stage at one travel limit and the loop of cable for y-axis is almost touches the floor. When the stage is at the other travel limit (dotted line), the cable loop opens up and the lower part of the loop rises from the floor. The cable is never strained or flexed significantly. A similar approach was employed to install the z-axis cable.

WIRING THE SYSTEM

It is important to resist the temptation to decrease cost and work by hard wiring rather than installing connectors. If an item may require repair or maintenance, install a connector.

The control cabinet (see Photo 2) has sufficient cable that it can move 10 away from the XY table. This distance keeps the PC away from the dust as much as possible. The cabling between the control cabinet and the machine has connectors at each end. There are three multiconductor cables (one for each axis), each with a different connector so cables can't be interchanged.

The bottom half of the cabinet contains the interface PCB, servo amps, power supply, switches, and other electronics. Of course, these items don't occupy all of the available space, but it was convenient to leave the salvaged items mounted and wired as they were originally. It also contains salvaged meters which monitor the various voltages.

COMMISSIONING

When a lot of time, effort, and money has gone into building and wiring a machine such as this, powering it up for the first time has to proceed carefully. I double checked that the voltages on the wires leaving the cabinet corresponded to the correct voltages and pins of the encoders. I was also concerned about feedback. I didn't want the system to oscillate due to instability or to run away on powerup because the sense of the feedback is wrong.

Some software available from Galil helped here. The software lets the system power up with the feedback loop open (i.e., the digital control algorithm is not yet running). This way, you can verify that the controller is reading the encoder correctly. The loop is then closed. However, if the system tends to run away, the loop is immediately opened, and you are directed to take corrective action. Either the motor leads need to be interchanged or the encoder channels need to be swapped to get the sense of feedback correct.

Once the loop can be closed without runaway, stability problems may still exist. The Galil software contains an automatic tuning routine which varies coefficients in the control algorithm and tests the response of your system. Continue testing until the system response is optimal (i.e., fastest response with least overshoot).

The values obtained by automatic tuning are a convenient starting point for additional tuning with integrator gain. Since introducing an integrator into the loop is destabilizing, apply it carefully. Integrator gain eliminates steady-state positioning errors, so I introduced a moderate amount.

USING THE MACHINE

There is an enormous amount of satisfaction in seeing months of mechanical and electrical work come together and function as intended.

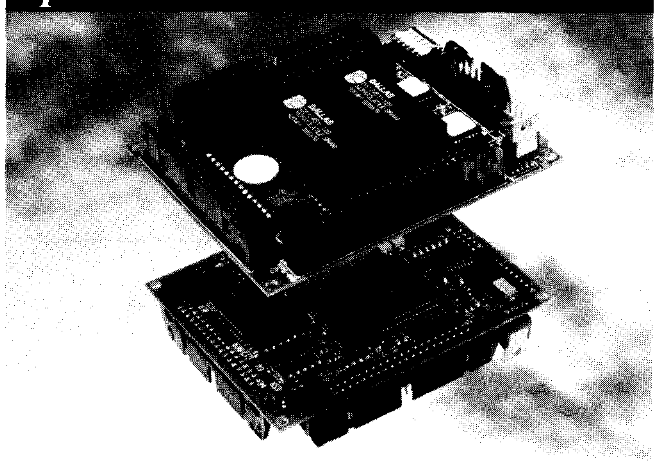
Anyone familiar with assembler programming will find this intelligent controller straightforward. The DMC600 series has 74 instructions to become familiar with, but you can get started with only a handful. However, just as assembler takes a great deal of effort and code to perform real tasks, so does programming an intelligent motion controller. You soon get the "there must be an easier way" yen.

With an additional piece of software available from Galil, the machine becomes part of a full-fledged CAD/CAM system. Making programs which direct the machine to follow a particular path and execute a particular set of instructions becomes virtually a drawing activity:

1. Draw the x and y path of a particular part using any CAD system that can produce a DXF version of the drawing. Typical drawings include macros to set gains and speeds, monitor digital inputs, and activate digital outputs. The behavior of the cutting tool is also macro driven. Directives indicating the tool radius and the side of the part the cutter is to follow are also important.
2. The DXF version of the drawing is submitted to an application which calculates a new path, taking into account tool offset, and produces a set of assembler-like motion commands. Essentially, your high-level code has been compiled into executable assembler code.
3. Do a dry run of the DMC code on the machine to verify that you included everything in the drawing that was needed for the machine to manufacture the part correctly. It's important to make a permanent record of the tool path. On my machine, the tool holder also includes a tube which accepts a felt marker.

Like any program development task, the steps have to be reiterated before everything is exactly right. Some examples of various work samples are shown in Photo 3.

Speed Your Product to Market



With the QED Board™

This ready-to-use computer packs dozens of analog and digital I/O lines, communications, graphics display interface, and resident software on a palm-sized card. It delivers everything you need:

- Programmable in C and Forth
- 8 and 12 bit A/D, 8 bit D/A
- Two RS232/485 serial ports
- Real-time multitasking kernel
- 68HC 11 microcontroller
- 32 digital I/O lines
- High current drivers
- Built-in software libraries

Off-the-Shelf* Solutions *Put You on the Fast Track

The QED Product Design Kit™ and Industrial Control System™ integrate the QED Board with keypad, display and connectors in a versatile enclosure to deliver a complete solution for your product design and control needs,

Mosaic Industries, Inc., 5437 Central Avenue, Suite 1, Newark, CA 94560, (510) 790-1255 phone, (510) 790-0925 fax

#112

Program Microcontrollers! Control the World! . . . or just Bebop to the Boolean Boogie!

Programming Microcontrollers in C
by Ted "an Sickle, Motorola

Microcontrollers are showing up everywhere! But to use them, you have to program them, and that is a lot different from programming a PC. Van Sickle takes you Inside today's hottest microcontrollers and shows you how to use C to program them for such tasks as motor control and digital signal processing. $29.95 and worth every cent!

Controlling the World with Your PC
by Paul Bergman

Use your PC to control stepper and servo motors, monitor fluid levels, switch appliances on and off, monitor alarm systems and much more. All circuits connect directly to the parallel printer port. Complete source code listings in BASIC, C, and Pascal are Included on a 3.5-in floppy. Each circuit is fully described with theory, parts values, and schematic. Only $29.95, including the disk.

Bebop to the Boolean Boogie
An Unconventional Guide to Electronics Fundamentals, Components, and Processes
by Clive "Max" Maxfield

fhe only book we know of where you can simultaneously learn about finite state machines, Reed-Müller logic, musical socks, and the best time of day to eat smoked fish! This reference is huge, with hundreds of diagrams that show **you** the intimate details of integrated circuits, hybrids, multichip modules, and circuit boards. Electronics will never be the same. $35.00, and worth the price for the glossary alone.

To order, call I-800-247-6553
Or send check/money order to HighText (include $4.00 S/H), CA residents Include sales tax.

HighText publications inc
PO Box 1489, Solana Beach, CA 92075

#113

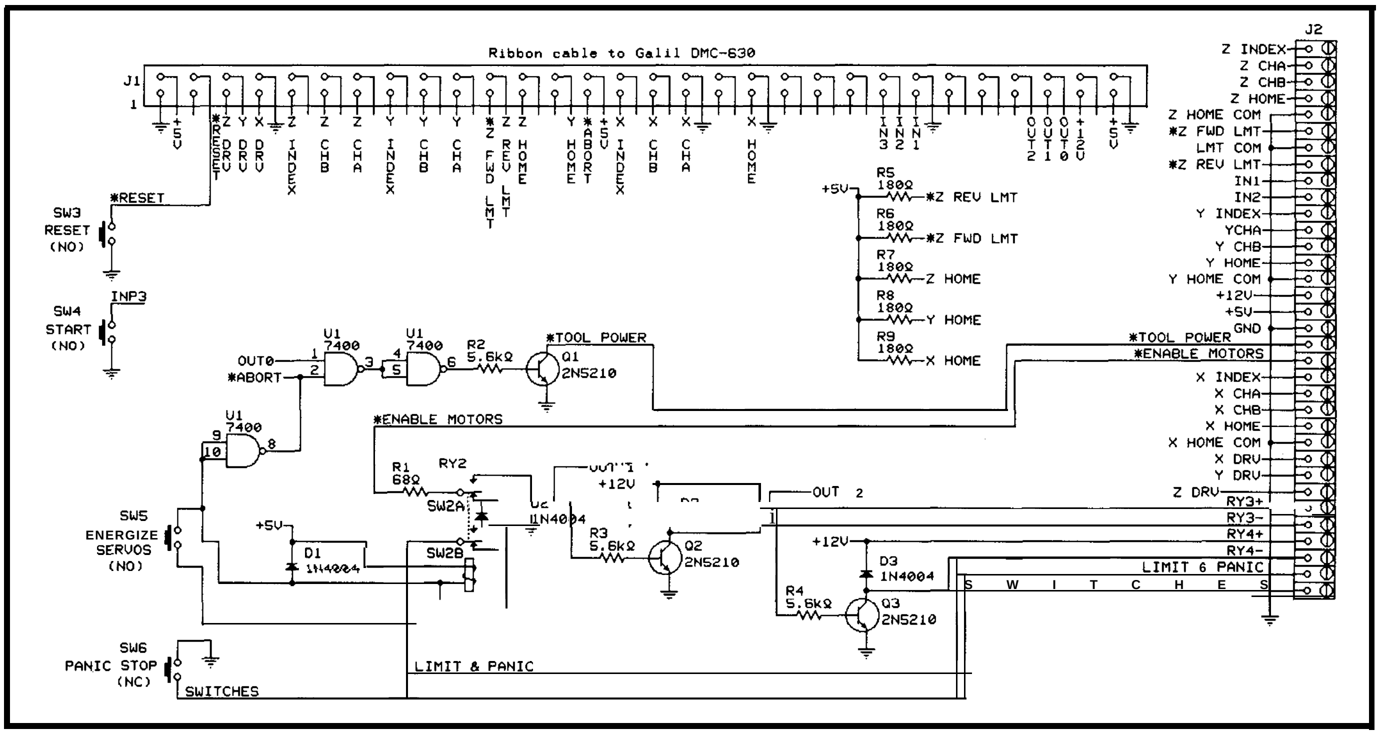


Figure 5—*The interface PCB makes the task of getting signals to and from the DMC630 easier by using a screw-type terminal block.*

DESIGN PROBLEMS AND SOLUTIONS

Despite the best of efforts, the first version of anything always has some problems. Some problems here were corrected with a minor amount of rework, while others will have to wait for version 2.

While using bearings as wheels for the motion stages is functional, it does have one minor drawback. Even though the rails are round, they accumulate a coating of fine dust. As the bearing wheels pass over the dust, it compresses and sticks to the wheel. After 6 hours, the build-up needs to be removed since it is not uniform and the wheels are bumpy. Various approaches could prevent dust buildup on the wheels (e.g., a wiper or small air lines). Enough time is wasted cleaning the wheels, that stopping dust buildup involves significant savings.

The lead screws are subject to whipping, which means they are a combination of too long and too thin. Given that this system was somewhat difficult to make stable, a larger-diameter lead screw would have exacerbated the problem. Lead-screw whip is not a problem at normal cutting speeds (0.5 in./s), but is evident at speeds above 2 in./s.

The worst aspect to this problem is that the stages must move slowly even when the machine is not cutting. By rotating the nut and keeping the lead screw fixed, this problem can be eliminated.

It amazes me that the PC for this machine has given so little trouble since the power supply's cooling fan draws dust into the floppy drives. A clear plastic apron in front of the drive openings slows the dust buildup in the floppy drives, but has not stopped it.

The better solution comes from removing the power supply and fan from the PC enclosure and powering the PC from supplies in the cabinet. Despite dusty conditions, however, the drive's heads have only been cleaned once in two years of operation.

Although there has not been a significant problem, two areas of the motion stage mechanical rigidity need to be improved. Some flexing of the *y*-axis support beam can be observed when the tool is lowered into a work

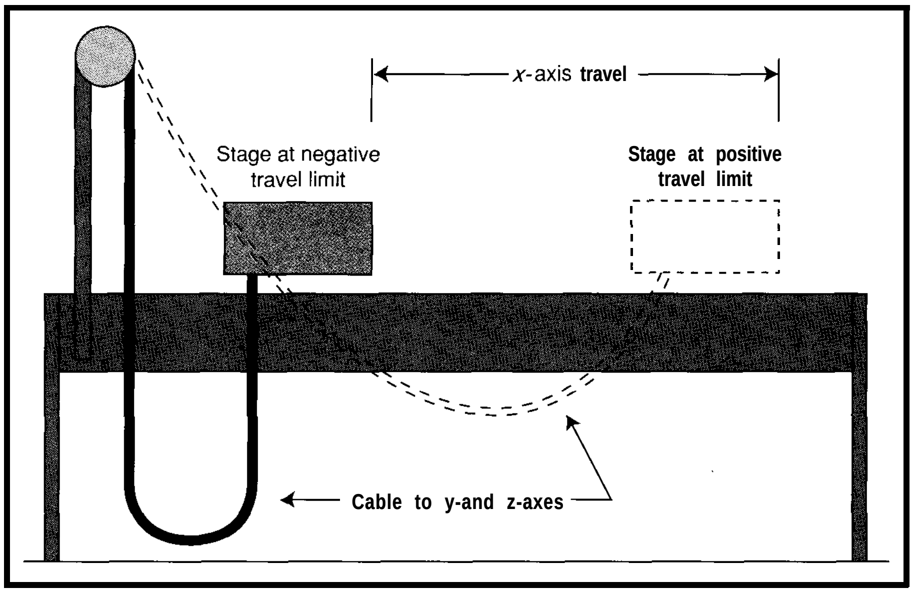

x-axis travel
Stage at negative travel limit
Stage at positive travel limit
Cable to y-and z-axes

Figure 6-By elevating one end of the cable, the stage can move without stressing or damaging *the cable.*

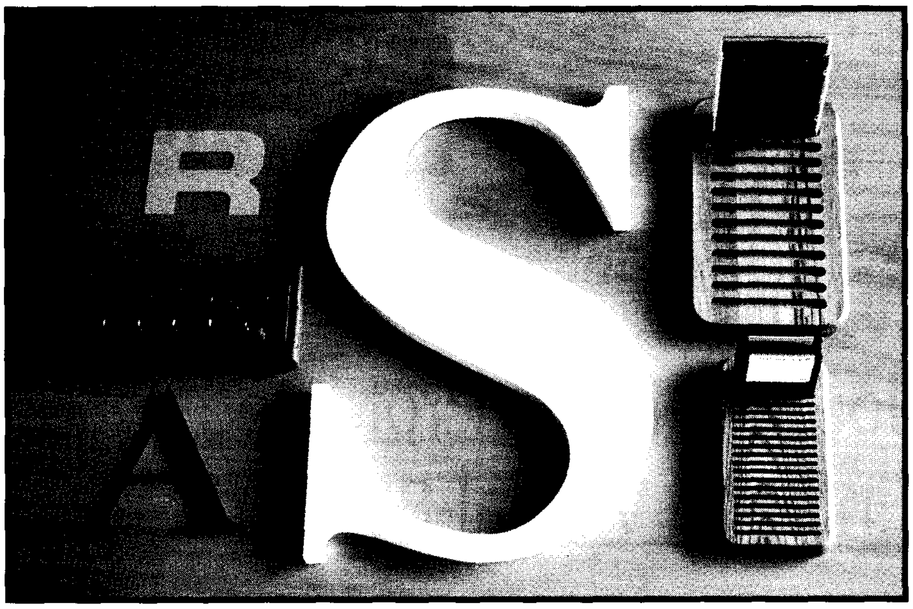


Photo 3-A *wide range of interesting products can be made with this machine such as letters of various sizes and materials, key racks, and holders for CDs and floppy disks.*

piece. The y-axis support beam also racks somewhat in the x and y plane if a x-axis retarding force is applied on one end and not the other due to flexing of the bearing mounting plates on each end of the y-axis support beam.

The racking of the y-axis support beam in the x and y plane could be eliminated by a double-lead screw drive. Instead of one lead screw in the middle, there should be one close to each end, thereby eliminating racking completely. ❑

Gordon Dick is a P. Eng. who teaches electronics at Northern Alberta Institute of Technology in Edmonton, Alberta. He also operates a small business selling the services of a computer-controlled XYZ table. He may be reached at (403) 963-0290.

SOURCES

Galil Motion Control, Inc.
575 Maude Ct.
Sunnyvale, CA 94086
(408) 746-2300
Fax: (408) 746-2315

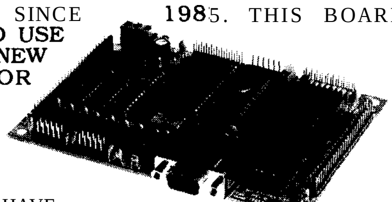
Servo Systems Co.
115 Main Rd.
P.O. Box 97
Montiville, NJ 07045-0097
(201) 335-1007
Fax: (201) 335-1661

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

SINGLE BOARD COMPUTER

THAT'S RIGHT! $129.95 FOR A FULL FEATURED SINGLE BOARD COMPUTER FROM THE COMPANY THAT'S BEEN BUILDING SBC'S SINCE 1985. THIS BOARD COMES READY TO USE FEATURING THE NEW 80535 PROCESSOR WHICH IS 8051 CODE COMPATIBLE. ADD A KEYPAD AND AN LCD DISPLAY AND YOU HAVE A STAND ALONE CONTROLLER WITH ANALOG AND DIGITAL I/O. OTHER FEATURES INCLUDE:



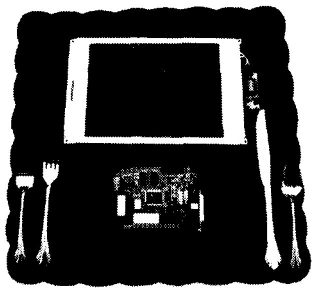
- UP TO 24 PROGRAMMABLE DIGITAL I/O LINES
- 8 CHANNELS OF FAST 8/ 10 BIT A/D
- OPTIONAL 4 CHANNEL, 8 BIT D/A
- UP TO 4.16 BIT TIMER/COUNTERS WITH PWM
- UP TO 3 RS232/485 SERIAL PORTS
- BACKLIT CAPABLE LCD INTERFACE
- OPTIONAL 16 KEY KEYPAD & INTERFACE
- 192K OF MEMORY SPACE, 64K INCLUDED
- **8051 ASSEMBLER & MONITOR INCL., BASIC OPT..**

1985-1995 10 YEAR ANNIVERSARY

EMAC, inc.
618-529-4525 Fax 457-0110 BBS 529-5708
P.O. BOX 2642. CARBONDALE. IL 62902

#114

Flat Panels Served Here



House **Specialty - Monochrome LCD Kit $249**

Le Earth LCD Restaurante Menu

Appetizers [Controllers for PC ISA Bus]

Eorth LCD/M Monochrome LCD Controller	$149
EarthVision/ISA Color LCD Controller	$299

Entrees (Include LCD, ISA Controller, Cable & Bocklite Inverter)

Monochrome 9.4" Kit	$249
Color Single Scan 8.2"	$450
Color Dual Scan 9.4"	$749
Color 9.4 TFT	$995

Ala Carte (LCD Only)

Mono 9.4" LCD	$95
Color 9.4" Single Scan	$125
Color 9.4 Dual Scan	$399

EARTH Computer Technologies *"The Flat Panel Solutions Company"*

P.O. Box 7089 - **Laguna Niguel - California - 92607**
Phone: (714) 448-9368 - Fax: (714) 448-9316 - BBS: (714) 448-8093

#115

DEPARTMENTS

32 Firmware Furnace

40 From the Bench

84 Silicon Update

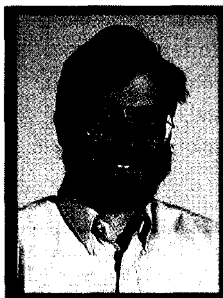
88 Embedded Techniques

99 ConnecTime

FIRMWARE FURNACE

Ed Nisley

Journey to the Protected Land: Entering Virtual-86 Mode



According to Ed, any successful PC operating system must run DOS programs dating back to the original PC. Ed traces through the complex architecture to show you how '86 compatibility takes place.

Looking at the design of the '386 without a historical perspective, it would be hard to imagine what possible use there is for a special mode whose purpose is to restrict the functionality of the processor and introduce obviously inefficient execution. You might ask: "Why would anyone run anything in this silly mode!"

Microprocessors: A Programmer's View *by* Dewar and Smosna

Some folks regard the entire Intel 80x86 line as "silly mode" run rampant. Even ignoring market-share envy, their accusation has a grain of truth. The '386 architecture is gruesomely complex in comparison to early RISC designs. It's less so when compared to more recent RISC mutants, however, which should give pause for thought.

The '386 CPU's Virtual-86 mode is a silicon-assisted PC Compatibility Barnacle, not just a CISC feature. You could write an 8086 CPU emulator in pure software, but acceptable performance requires hardware assistance. The key to V86 mode is the realization that most instructions run on genuine hardware with no emulation overhead at all. Only a few operations require custom intervention.

Why emulate an 8086 at all?

Well, because any successful PC operating system must run dusty-deck

DOS programs dating back to the Original PC. Full DOS compatibility requires a vast infrastructure of code emulating all the quirks built up over the years. Perhaps someone will write a book detailing the evolution of DOS boxes from the advent of the 80286 to the present-a memorial for The Operating System That Just Will Not Die.

The FFTS, in contrast, has a much simpler objective. The real-mode embedded programs I've covered in previous columns can operate without DOS or BIOS services. Although I plan to add BIOS support, so the V86 task won't be completely alone down there in the first megabyte, building a fully functional DOS box isn't in the plan.

Nevertheless, a great expanse of interesting and unexplored territory just opened up ahead of us. This month, I'll define the terms, set up a simple V86 task, and show why we need more than just 16-bit code to make it work.

DOES VIRTUAL ❑ REAL?

The '386 CPU emerges from a hardware reset in real mode. The PC BIOS gets control at the standard reset address, goes through the usual BIOS hocus pocus, and eventually loads a program from diskette. When that program is our special boot sector loader, it pulls off a series of stunts that culminates with the FFTS running in 32-bit protected mode.

The Intel manuals define the precise series of steps required for a graceful transition from protected mode back to real mode, none of which apply to Virtual-86 mode. The CPU must be in 32-bit protected mode before starting a task in V86 mode and, as we'll see shortly, it switches back to 32-bit PM after exiting V86 mode. Thus, contrary to popular opinion, *Virtual-86* mode is not *real* mode.

Bit 17 of the EFLAGS register controls what's formally known as Virtual Machine (VM) mode. Because the "virtual machine" was designed to execute real-mode 8086 programs, Virtual Machine mode is commonly called *Virtual-86 mode.* Just to keep things confusing, that EFLAGS bit is called VM rather than V86.

In normal 32-bit PM operation, the VM bit is zero and the CPU behaves as we've come to expect. When a 32-bit PM task sets VM, however, a new set of rules applies.

First and foremost, V86 mode renders everything we've learned about protected-mode segment registers, descriptors, and selectors completely inoperative. The CPU suddenly generates addresses in classic real-mode style: shift the appropriate segment register's contents left by four bits, add a 16-bit offset, and send the resulting value out as a memory address.

In both real and V86 modes, the CPU can address 1 MB plus 64 KB minus *16* bytes: 000000–10FFEF, the result of adding FFFFO and OFFFF. The original AT includes a gate on address line A20 to restrict addresses to OOOOO-FFFFF, but that gate must remain on in protected mode. As a result, rude V86 tasks can scribble on the first 64 KB (minus 16 bytes) of RAM beyond the 1 -MB line.

Of more importance, V86 tasks cannot address memory beyond that limit. We must copy the V86 task's instructions and data to RAM below the l-MB line before entering V86 mode. The segment registers must point at those addresses with real-mode contents, not segment selectors. Unlike 32-bit PM code, V86-mode programs do not use GDT or LDT descriptors.

The '386 CPU's memory-paging hardware can relocate V86 tasks anywhere in memory and prevent them from writing into that 64-KB block above 1 MB. Activating that machinery is sufficiently complex that

Listing 1—*The FFTS task initialization routine creates a 32-bit code descriptor covering the task's code segment. For V86 tasks, if sets CS:EIP to the proper real-mode address, creates a data segment descriptor at that address, and copies the code. Because V86 tasks do not use descriptors, if rep/aces the 32-bit code segment descriptor with the data descriptor, thus giving the V86 monitor read-wife access to the task's code. The task's initialized data segment receives similar treatment*

```
        MOV      [TSS_PTR.CS],LDT_CODE
        MOV      EAX,[INIT_PTR.CodeStart]
        MOV      [TSS_PTR.EIP],EAX
        CALL     MemGetDescBase,GDT_CODE,GDT_GDT_ALIAS   ; our code seg
        MOVZX    EBX,[INIT_PTR.CodeSegment]; cvt task seg to linear
        SHL      EBX,4
        ADD      EAX,EBX                     ; code seg + task seg
        CALL     MemSetDescriptor,LDT_CODE,[LDTAlias], \
                 EAX,[INIT_PTR.CodeLength],ACC_CODE32,ATTR_32BIT
        TEST     [INIT_PTR.InitTaskInfo],MASK TaskV86
        JZ       @@CSdone                    ; if not V86 task, done!

        MOVZX    EAX,[INIT_PTR.CodeSegment]; V86 needs numeric CS
        MOV      [TSS_PTR.CS],AX
        CallSys  CGT_MEM_MAKELIN,EAX,0       ; linear seg start addr
        MOV      ECX,[INIT_PTR.CodeLength]   ; size of code segment
        CALL     MemSetDescriptor,LDT_UDATA,[LDTAlias], \
                 EAX,ECX,ACC_DATA16,ATTR_16BIT   ;... treat as data
        MOV      EBX,LDT_UDATA               ; aim GS:EBX -> target
        MOV      GS,EBX
        MOV      EDX,ES                      ; save ES around loop
        MOV      EBX,LDT_CODE                ; aim ES:EBX -> source
        MOV      ES,EBX
        XOR      EBX,EBX
@@MoveCode:
        MOV      AL,[ES:EBX]
        MOV      [GS:EBX],AL
        INC      EBX
        LOOP     @@MoveCode
        MOV      ES,EDX                      ; restore ES
        MOV      GS,EDX                      ; make GS safe
        CallSys  CGT_MEM_GETDESC,LDT_UDATA,[LDTAlias]
        CallSys  CGT_MEM_PUTDESC,LDT_CODE,[LDTAlias],EAX,EDX
        CallSys  CGT_MEM_KILLDESC,LDT_UDATA,[LDTAlias]
@@CSdone:
```

I'll stay with a single, well-behaved V86 task for quite a while.

V86 mode also affects how the CPU executes some instructions. Once again, contrary to popular opinion, the CPU is not restricted to just 8086 instructions. Nearly all real-mode instructions are available and you may perform 32-bit operations if you apply an operand-size prefix byte to each instruction. The FS and GS segment registers come in handy, too!

Address offsets must remain within real-mode limits, however. If you address memory using DS:ESI when ESI exceeds FFFF, for example, the CPU generates a protection exception. When you need more than 64 KB in a segment, V86 mode is not the right hammer for the job!

In *INK 57,* I described how the 2-bit IOPL field in EFLAGS restricts access to various instructions. V86 tasks, by definition, run at privilege level 3 (also known as Ring 3) and, like all other user-level tasks, should not have unrestricted access to sensitive system facilities. The CLI, STI, PUSHF, POPF, INT *n,* and IRET instructions cause a protection exception when IOPL < 3. You can set IOPL = 3, bypass the protection hardware, and jam the system if you like.

In 32-bit protected mode, the IOPL field also determines whether a user-level task can execute I/O instructions. In V86 mode, oddly enough, IOPL has no effect on IN, INS, OUT, and OUTS. Instead, the CPU checks the I/O Permission Bitmap in the task's TSS during each I/O instruction. If the bit corresponding to the I/O port's address is set, the CPU generates a protection exception.

Some instructions, such as LGDT, LIDT, and LIDT, are completely off limits in V86 mode because you can't twiddle key CPU registers from a low-privilege task. You may not stop the CPU with a HLT instruction, either! These instructions cause a protection exception if the CPU encounters them, regardless of the current IOPL setting.

The Intel manuals describe which instructions are valid in V86 mode (most are) and what happens when you use the others (a protection exception of one kind or another). For the most part, user program code isn't bothered by the V86 restrictions. Embedded system and operating system code is a little harder to write. We'll tailor V86 mode and make things work out right in upcoming columns.

Listing 2—*The TSS initialization code finishes its job by creating the task's stack, setting up a variety of registers, and loading the I/O Permission Bitmap. V86 tasks require two stacks: one in memory below 1 MB for use by the V86-mode code and another above 1 MB. For simplicity, this code grants access to all I/O ports by leaving the I/O Permission Bitmap filled with zeros.*

```
        CALL  MemAllocPerm,LDT_STACK,[LDTAlias], \
              TASK_STACKSIZE,ACC_STACK32,ATTR_32BIT, \
              GDT_CONST,OFFSET cStackDesc,[TSSSel],0babefaceh
        TEST  [INIT_PTR.InitTaskInfo],MASK TaskV86
        JNZ   @@SSreal
        MOV   [TSS_PTR.ESP],TASK_STACKSIZE-4; set PM task stack
        MOV   [TSS_PTR.SS],LDT_STACK
        JMP   @@SSdone

@@SSreal:
        MOV   [TSS_PTR.ESP],V86_INITSP        ; set V86 stack
        MOV   [TSS_PTR.SS],V86_INITSS
        MOV   [TSS_PTR.StackPtr0.Off],TASK_STACKSIZE-4 ; Ring-0
        MOV   [TSS_PTR.StackPtr0.Seg],LDT_STACK; PM stack
@@SSdone:

; set EFLAGS register in the TSS
        MOV   EAX,[INIT_PTR.InitTaskFlags]    ; EFLAGS register
        OR    EAX,MASK EF_OneBit              ; force this bit ON
        TEST  [INIT_PTR.InitTaskInfo],MASK TaskV86
        JZ    @@NotV86
        OR    EAX,MASK EF_VM                  ; set VM if appropriate
@@NotV86:
        MOV   [TSS_PTR.EFLAGS],EAX            ; ram it into the TSS

<<< general register setup omitted >>>
        MOV   EAX,OFFSET (TSS PTR 0).IOMap    ; bitmap offset addr
        MOV   [TSS_PTR.IOMapBase],AX
        MOV   [TSS_PTR.IOMapEnd],0FFh         ; set reserved byte
```

In short, while V86 mode isn't *really* real mode, it's close enough for us.

CRACKING THE GATE

Just like all other tasks in a '386 system, each V86 task is defined by a TSS. The cleanest way to enter V86 mode is an ordinary task switch, which means the TSS fields define the state of the CPU's registers at the start of the V86 task. A slightly tweaked version of the familiar FFTS task-creation code is enough to set up a V86 task.

Each FFTS task has a small structure defining its code and data segments, shown in *INK 55,* Listing 2. In the last few columns, I've quietly added a few bits and, this month, a new bit identifies V86-mode tasks. Perforce, that bit is zero for 32-bit PM tasks.

The V86 task's segments must be below 1 MB so real-mode seg:offset addresses can reach them. Because we're using real-mode tools to generate the FFTS code, Locate has already assigned an address to the task's code and data segments. All we must do is copy them from the FFTS disk image to the appropriate address. Listing 1 shows how it's done.

A protected-mode program cannot access storage without a read-write data descriptor, even when the addresses are below 1 MB. The initialization routine creates a temporary data descriptor covering the target code segment before copying the data. The offsets in both segments are identical, with ES and GS holding the LDT_CODE and LDT_UDATA sourceand target selectors, respectively.

The routine then copies the data descriptor into the LDT_CODE descriptor, giving the 32-bit PM code read-write access to the V86 code segment.

I'll show why this is vital later on. For now, remember that we'll never execute the code in 32-bit PM and, thus, don't need a code descriptor for it. If you're particularly fussy, you can choose another name for the LDT segment. By the way, this is probably the most roundabout way imaginable to get real-mode code into its proper location.

Snap quiz: how many copies of the V86 code and data segments are in storage after the task initialization code gets done with them? Extra credit: how does the FFTS code know where to put the segments!

The initialized data segment in LDT_DATA receives similar treatment. The V86Demo task this month doesn't use any data, although there is a single placeholder variable. Check the source code on the BBS for all the grim details.

Listing 2 shows the stack setup routine. The V86 code must have real-mode addresses in SS:SP, which I load with 0000:7BFE for historical reasons. That's just below the default disk

Listing 3-This Virtual-86 task writes directly info the video buffer using ordinary real-mode addresses. The In t 20 instruction causes a protection exception that invokes the V86 monitor.

```
    DefV86TaskCS             ; includes USE16 seg size spec

    PROC  TaskProc

    MOV   AX,BIOS_SEG        ; find CRT controller address
    MOV   ES,AX
    MOV   AX,[ES:63h]
    MOV   BX,0B800h          ; assume color
    CMP   AX,003D4h          ; is it so?
    JE    @@SetAddr
    MOV   BX,0B000h          ; nope, reset to monochrome
@@SetAddr:
    MOV   ES,BX              ; aim ES:DI at video buffer
    MOV   DI,(2*80*25)-2     ; ... the very last character!
    MOV   [BYTE PTR ES:DI+1],0Fh     ; make it white on black

    XOR   CX,CX              ; set up char counter
    MOV   DX,SYNC_ADDR       ; set up for scope blips

@Again:
    IN    AL,DX              ; set trace blip
    OR    AL,20h
    OUT   DX,AL

    MOV   [ES:DI],CH         ; pop char into video buffer
    INC   CX                 ; and step to the next one
    PUSH  CX                 ; exercise the stack
    MOV   CX,3000
```

(continued)

Home Control is as simple as 1, 2, 3...

Energy Management
A
Access Control
A
Coordinated Home Theater
☆
Coordinated Lighting
☆
Monitoring & Data Collection

Get all these capabilities and more with the Circuit Cellar HCS2-DX. Call, write, or fax us for a brochure. Available assembled or as a kit.

1) The Circuit Cellar HCS2-DX board is the brains of an expandable, network-based control system which incorporates direct and remote analog and digital I/O, real-time event triggering, and X-10 and infrared remote control. Control programs and event sequences are written on a PC in a unique user-friendly control language called XPRESS and stored on the HCS2-DX in nonvolatile memory.

2) The Relay BUF-Term provides hardwire interface protection to the HCS2-DX. Its 16 inputs accommodate both contact-closure and voltage inputs within the range of ±30 V. Its eight relay outputs easily handle 3 A AC or DC, to directly control solenoids, motors, lamps, alarm horns, or actuators.

3) The PL-Link provides wireless X-10 power-line control to the HCS II system. The PL-Link is an intelligent interface that sends, receives, and automatically refreshes X-10 commands. It works with all available X-10 power-line modules.

Of course, there's a lot more! The HCS II system has phone and modem interfaces, infrared remote control, voice synthesizers, and much more. Whatever your application, there's a configuration to accommodate it.

GET STARTED TODAY WITH AN HCS2 "123-PAK"

The 123-PAK consists of an HCS2-DX board, Relay BUF-Term board, PL-Link board, TW523 power-line interface, PS12- 1 power supply, serial cable, and V3.0 HCS XPRESS software.

Assembled $65 1 **Kit** $461

CIRCUIT CELLAR, INC.

4 Park Street, Vernon, CT 06066
(203) 875-2751 • Fax (203) 872-2204

#117

Listing *3-continued*

```
@Stall:
     LOOP  @@Stall           ; add a non-trivial delay

     IN    AL,DX             ; clear trace blip
     AND   AL,NOT 20h
     OUT   DX,AL
     INT   20h               ; crash into V86 monitor...
     POP   cx                ; restore CX after switch!
     JMP   @Again            ; repeat forever

     ENDP TaskProc
     EndTaskCS
```

bootstrap load point, a magic number that will come in handy later on.

The initialization code also loads a TSS field that we haven't used before: the Ring-O SS:ESP registers. When an interrupt occurs in a Ring-3 task, the CPU must find a Ring-O interrupt or trap gate in the IDT. It switches from the user task's Ring-3 stack to the Ring-O stack defined in the TSS before pushing any values. Our new V86 code is the first task we've seen that doesn't run at privilege level O!

The remainder of Listing 3 sets up EFLAGS and the I/O Permission Bitmap. The code turns on the VM bit in EFLAGS and leaves IOPL set to zero. This operation ensures that the restricted instructions described above will cause a protection exception; the V86 task cannot do something rash like disable all hardware interrupts or clobber the EFLAGS register.

I included the I/O Permission Bitmap in the intial TSS definition a few months ago, but did not activate it. V86 tasks must have a valid bitmap because the CPU refers to it when executing I/O instructions. The Intel doc explains why the byte just after the table must be filled with FF; this was actually a workaround for a bug in early '386SX chips.

Each bit in the I/O Permission Bitmap controls access to a single 1-byte I/O address. If the bit is zero, I/O operations in user-privilege tasks may use that port. If it's one, those tasks cause a protection exception. A 16-bit (2-byte) port has two associated bits and a 32-bit (4-byte) port has four. It's that simple!

For the moment, the bitmap is even simpler because it grants unrestricted access to all I/O ports. Recall that the setup code clears all of storage above 1 MB, including the locations that hold the TSS.

The FFTS task initialization code produces the trace output shown in Figure 1. Pay particular attention to EFLAGS and the segment registers. The LDT contains three data descriptors covering the task's code, initial-

8051 / C251

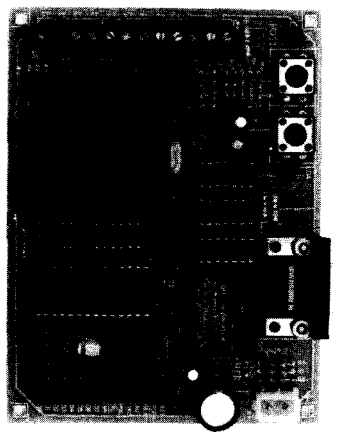
EMBEDDED CONTROLLERS

IGEL Corporation introduces its newest board, the **R-51JX**, designed for Intel's **80C251** chip. RIGEL builds and supports 8 and **16** bit development tools for embedded controller systems. We eature hardware, software, books, and kits, for educational and ndustrlal markets.

HE RIGEL DIFFERENCE

ll of our boards come with:

- **32K EPROM**
- 32K RAM or EEPROM
- Machine screw sockets
- Power on LED
- All system signals on headers
- **I/O** available on terminal blocks
- Sample programs
- Circuit diagrams
- Assembly language IDS software READS
- **BBS Tech** support



Our entire line of 8051 boards are programmable in Assembly, BASIC, **"C"**, and Forth. We also offer a low cost Fuzzy Logic Software code generator and a chip simulator for the 8051 family. Complete systems start at $85

CALL TODAY FOR MORE INFORMATION

RIGEL Corporation,

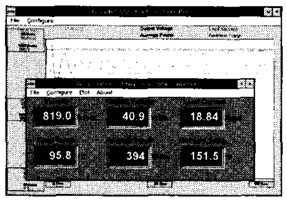
PO BOX 90040, GAINESVILLE, FL 32607

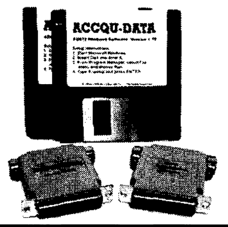
Telephone (904) 373-4629 BBS (904) 377-4435

#118

Printer Port Data Acquisition

- *Simple **installation**
- **No** external power required
- **6** Analog inputs (12 bit)
- 3 Digital I/O lines





- **Use** with the included software or with your own DOS or Windows **programs.** (Windows DLL provided)
- **DDE** to Excel or other applications
- **Custom** software available

AD612	6 channel, 12 bit, Data Acquisition module w/3 digital I/O lines.Includes software for displaying, plotting, and logging to disk.	$99.95
TSC5-T	Thermocouple Signal Conditioner for use with AD612. Accepts 5 type T thermocouple inputs on channels 1-5 and displays ambient temperature on channel 0	$79.95
MSC01	Multi-input Signal Conditioning Module for use with AD612 . Includes 2 type T Thermocouple inputs, 1 Differential input, 1 Tachometer input (for frequency output sensors), 2 voltage inputs, and 3 digital I/O lines.	$89.95
PT-1081	Prototyping board. Connects with AD612, TSC5, or MSC01. Facilitates construction of input signal conditioning and filter circuits. Includes 1 each: 25 position male and female D-sub connectors. (w/ plated through holes for leaded parts)	$19.95
PT-1082	ProtoTyping board. Same as PT-1081 except for surface mount parts.	$19.95
PT-1085	Terminal board. Converts 25 pin D-sub connector to 25 position screw terminal block. Allows easy connection of wires to any of the modules or prototyping boards above.	$19.95
TP4-T	4" Thermocouple probe (Type T)	$29.00 (5 for $115)

ACCQU-DATA

297 Route 72 West Suite C-178 Manahawkin, NJ 08050-2890

FAX-BACK / INFO : (609) **597-1221** BBS : *(908) 774-7499*

TO ORDER : (800) 341 ■ 5590

OUTSIDE US: (970) 872-8355

#120

```
TSS Dump of [Virtual-86 Demo1 Sel=1080 Base=00131000
 Backlink=0000 LDT Sel=1088
 CS:EIP=0526:00000000 EFLAGS=00020202 CR3=00000000
 SS:ESP=0000:00007BFE EBP=00000000 IOMapBase=0108 Trap=0000
 DS=20BD  ES=0000  FS=OOOO  GS=0000
   EAX=00001080 EBX=00000000 ECX=00000000 EDX=00000000
   EDI=00000000 ESI=00000000
 SS:ESP 0/0024:00000FFC1/0000:00000000 2/0000:00000000
LDT Dump of [Virtual-86 Demo] LDT Sel=1088
 0004: 003034DE  00008C00
 oooc: 52600039 00009300
 0014:  OBD00003 00009302
 0024: 3ACOOFFF 0040926A
I/O Permission Bitmap of [Virtual-86 Demo]
 port +      00      20      40      60
 (all ports enabled)
```

Figure l--The initial *TSS* fields of a *V86* task distinguish it from ordinary *32-bit* PM tasks. In particular, note the contents of *EFLAGS and the segment registers. The I/O Permission Bitmap isn't listed because it's filled with zeros that allow access to all I/O ports.*

ized data, and stack segments. There is no uninitialized data segment.

The FFTS task dispatcher starts the V86 task by executing a FAR JMP to an address containing the task's TSS selector. The CPU stores the current task state in the old TSS and loads the V86 task's state from the new TSS. Because the VM bit in the TSS's EFLAGS field is set, the CPU interprets the contents of CS:EIP as real-mode address bits and begins executing the first instruction in the task.

For all intents and purposes, we're back in real mode.

BEYOND THE FENCE

Listing 3 shows the simple V86 task we'll use for the next two columns. The DefV86TaskCS macro includes a USE 16 segment size specifier that forces 16-bit data and addresses. If you use a 32-bit register or data value, the assembler inserts an operand-size prefix byte that switches the CPU into 32-bit mode for that single instruction.

The V86Demo code determines the video buffer address and creates a pointer to the bottom-right character. The segment address is either B000 or B800, just as in real mode, and these values do not refer to magically created GDT or LDT selectors. I hard coded the character offset for an ordinary 25-line, 80-character screen rather than fiddle with all the BIOS values. Feel free to tweak the code for your system.

The task's endless loop wiggles a parallel port bit using ordinary I/O instructions, then increments CX [not ECX!) and writes CH into the video buffer. You get easily visible indications that the task is executing, the code can access the video buffer, and the stack works correctly.

Although the loop may be endless, there must be a way to return control back to the FFTS task dispatcher. In 32-bit PM tasks, we use a FAR JMP to the dispatcher's call gate in LDT[0]. In V86 mode, that won't work because we don't have access to the LDT. In fact, we can't even create a 32-bit gate address, let alone call it!

The CPU exits V86 mode when a hardware interrupt or protection exception occurs. V86 tasks must be carried through the door feet first on a stretcher; there's no graceful way out.

The INT 20 instruction near the bottom of the loop in Listing 3 triggers a protection exception because IOPL = 0. The CPU invokes the infamous General Protection Fault handler at Int 0D. The PM error handler displays the address of the failing instruction, dumps the TSS values, and halts the system.

Feet first and face down!

Obviously, that's not the proper response to a software interrupt instruction. If you plan to run V86-mode tasks in your system, you must provide what's called a "V86 monitor" routine. That monitor intercepts GPFs and other error conditions, examines the situation, and does whatever is required to keep the V86 task running. In this case, the monitor should pass control to the FFTS task dispatcher, which starts the next task in its list.

I must postpone describing the monitor until next month. Rest assured, however, that the code on the BBS has both the V86 task and a simple monitor to keep it going. Download the files and you'll have a headstart on next month's discussion.

RELEASE NOTES

This month, we add a single Virtual-86 taskette to the three 32-bit protected-mode taskettes. The 16-bit code blips a parallel port bit and writes directly into the video buffer, which is enough to demonstrate it works. Most important, the CPU can address memory with real-mode segments in what's otherwise a protected-mode program.

Next month, we'll build a simple V86 monitor routine that tosses the 16-bit task back through the door head first and kicking.

CPL	Current Privilege Level
DPL	Descriptor Privilege Level
EOI	End Of Interrupt (command)
FDB	Firmware Development Board
FFTS	Firmware Furnace Task Switcher
GDT	Global Descriptor Table
GDTR	GDT Register
IBF	Input Buffer Full
IDT	Interrupt Descriptor Table
IF	Interrupt Flag
IOPL	I/O Privilege Level
LDT	Local Descriptor Table
LDTR	LDT Register
NT	Nested Task
OBF	Output Buffer Full
P bit	Present bit (in a PM descriptor)
RF	Resume Flag
RPL	Requestor Privilege Level
TF	Trap Flag
TR	Task Register
TSS	Task State Segment
VM	Virtual Machine (in EFLAGS)

***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.*

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

Creating the Smart-MD

DC Motor Control for the I^{2}C Bus

FROM THE BENCH

Jeff Bachiochi

Want to take control of DC motors without using a lot of I/O bits? Here, Jeff integrates National's H-bridge motor driver with the I^{2}C and PWM functions of Microchip's PIC16C73.

I have many methods. I can use casual conversation: "The sink sure is full of dishes," which is greeted with silence. There's the looking-for-sympathy method: "Does anyone know where I can find a clean spoon." Silence. Then, there's the eliminate-the-excuse method: "Oh look, we're really not out of dish detergent." Silence.

If you have children, you know these methods rarely work. You've got to get straight to the point. "Alicia, do the dishes.. .Now!"

And, it's no different with my sons. I try to add a bit of humor, "Did you lose the lawnmower in the grass again?" And, it backfires, "No, Dad, I can still see the top of the handle."

It's not that kids are dumb. Au contraire, they are extremely intelligent. They are entirely competent to complete every task asked of them. However, they need a direct prod from a higher authority to get started.

Maybe this is where the term "distributed processing" comes from. By applying the right processor (an "able" body) and the right programming (don't mow over Mom's flowers), you can simplify a major task to single command. This leaves the main processor (namely, me) to go on with the other tasks at hand (wash the car, pay the taxes, or finish writing this article).

THE SMART MODULE

I see a growing demand for modules which can be easily pieced together to form a single system. Each module contains the smarts to perform a special task.

Smart modules are strung together with a main processor which collects, analyzes, and directs data while performing higher-level functions. When the system components (smart modules) are local to one another as with a robot, no special transmission drivers are needed.

Although the communication interface could be any number of protocols, I'm using I^{2}C because it is easy to implement on most processors with only two I/O lines.

Photo 1 displays the first module I want to introduce: the H-bridge motor driver, or Smart-MD. This consists of a PIC16C73 processor and two LMD18200T chips. The PIC looks like an I^{2}C slave device to an I^{2}C master. It

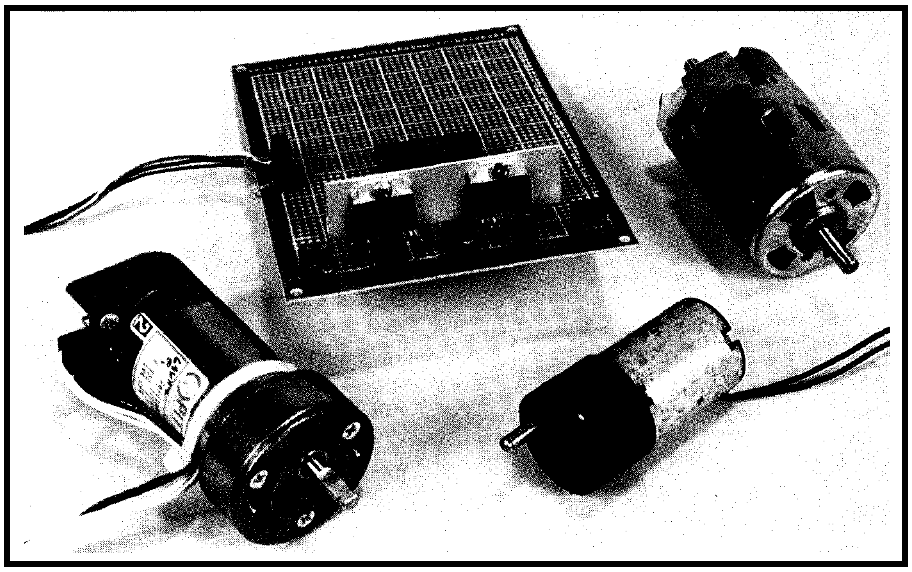


Photo 1—*The Smart-MD module contains just three chips and acts as an PC slave capable of controlling twin 3-A H-bridge motor drivers.*

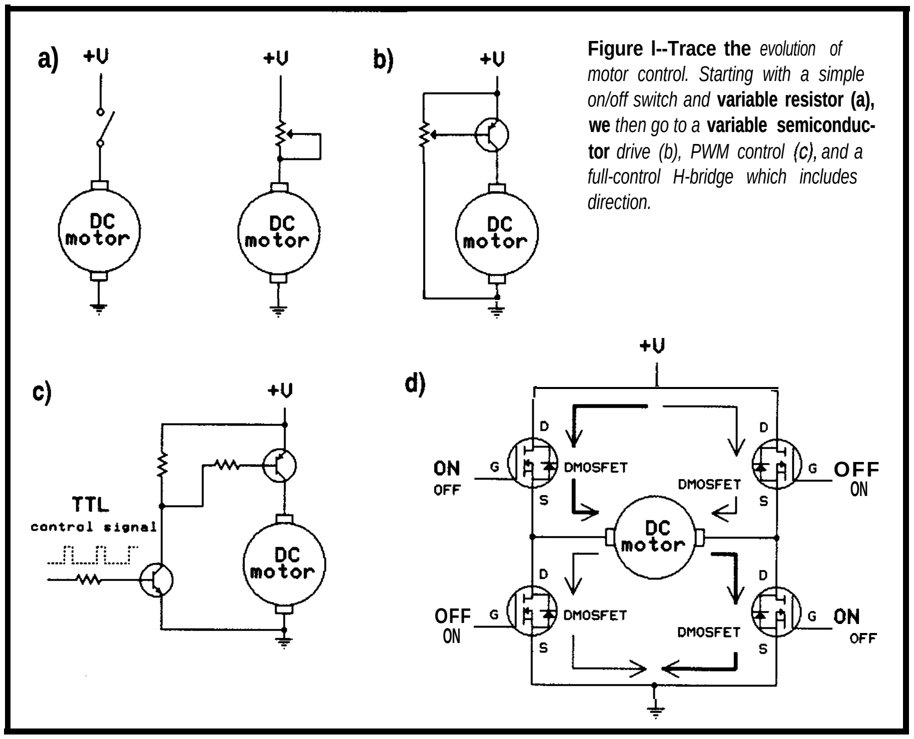


Figure 1—*Trace the evolution of motor control. Starting with a simple on/off switch and variable resistor (a), we then go to a variable semiconductor drive (b), PWM control (c), and a full-control H-bridge which includes direction.*

performs PWM, braking, direction control, and limit switch sensing for two PM DC motors. Each motor driver can supply 3 A of continuous current at up to 55 V.

PM DC MOTORS

I've done projects with stepper motors in the past, but not DC motors with brushes. Stepper motors don't move while a constant DC voltage is applied. They require multiphase control signals which energize field coils arranged sequentially around an armature made of permanent magnet poles.

As each coil is energized, it draws the closest armature pole toward it so the shaft steps or partially rotates and then holds position until the next coil is energized. The rotation speed is based on the speed of the sequencing field coils. The step size (degrees per step) is determined by the number of coils.

In contrast, PM DC motors run full speed while a constant DC voltage is supplied. The PM poles make up the stator and are located around the outside. The coils are wound around the armature and connect to the DC voltage source through opposing brushes, which contact only one step of armature coils at a time.

When a coil is energized, it is drawn toward a PM pole of the stator. This rotates the armature shaft in a step similar to the stepper motor, only now the coil becomes disconnected from the DC source as it rotates away from the stationary brushes. Mean-

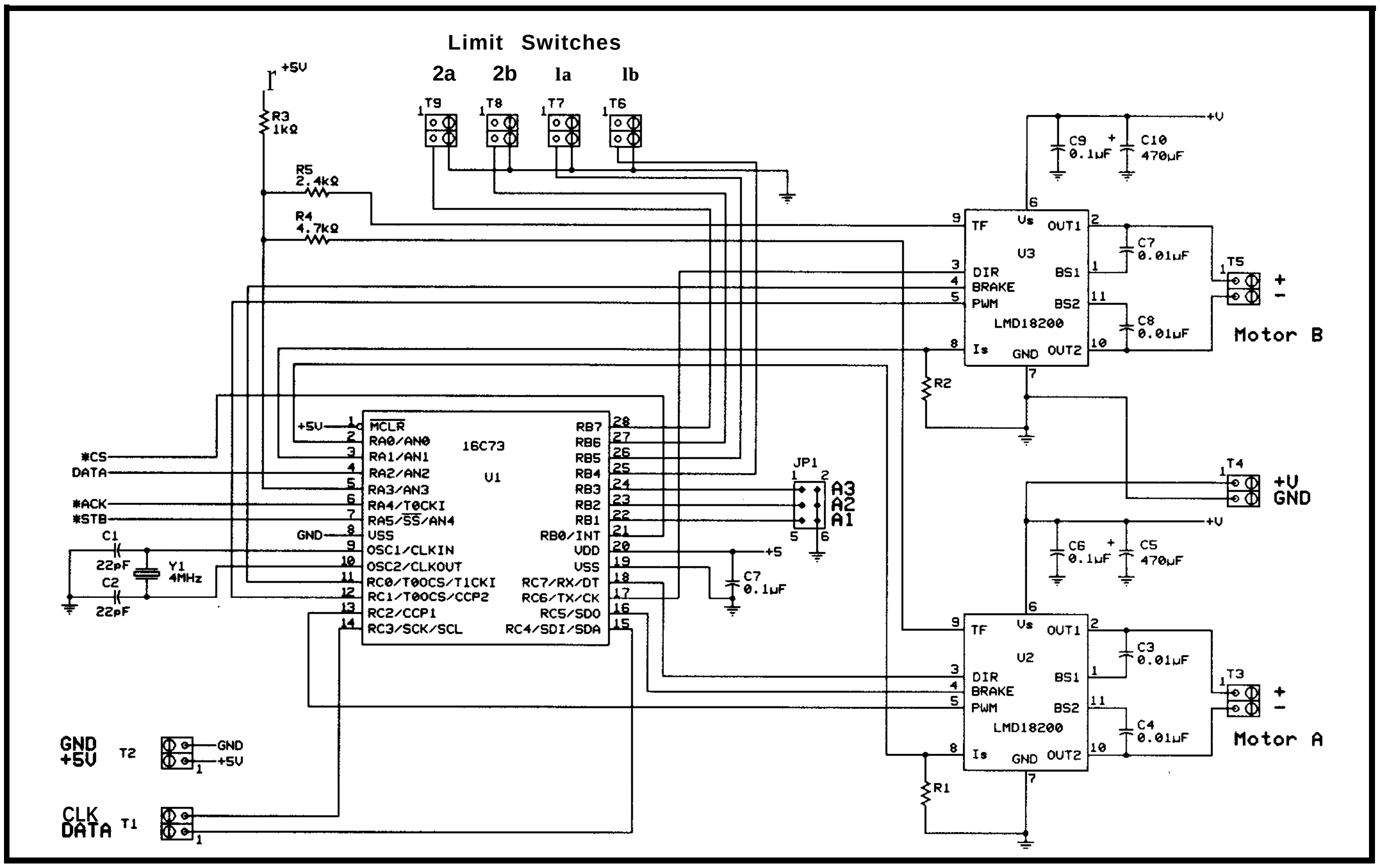


Figure 2—*By using the PC and PWM support built into the PIC16C73 and some dedicated H-bridge chips, the Smart-MD twin motor drive controller requires just three chips.*

while, the next coil becomes energized and continues rotating the shaft. This rotation continues as long as DC voltage is applied. The speed is based on the amount of DC voltage.

The simplest motor control is a switch in series with the motor. Replace the switch with a pot and you can control the motor's speed (Figure la). Both the switch and the pot can be replaced by a transistor. However, when operated in its linear region, the transistor must be able to dissipate a good deal of power [Figure lb). Dropping half the power across the transistor is a total waste-or at least half a total waste.

The trick is to use the transistor as a switch, not a pot, and to halve the power using pulse width modulation (PWM). Turn it on for half the time and off for the remainder. By varying the duty cycle (on time vs. off time), you can have variable motor speed without wasting power (Figure lc). Better yet, the motor gets to run at its rated voltage for ultimate performance,

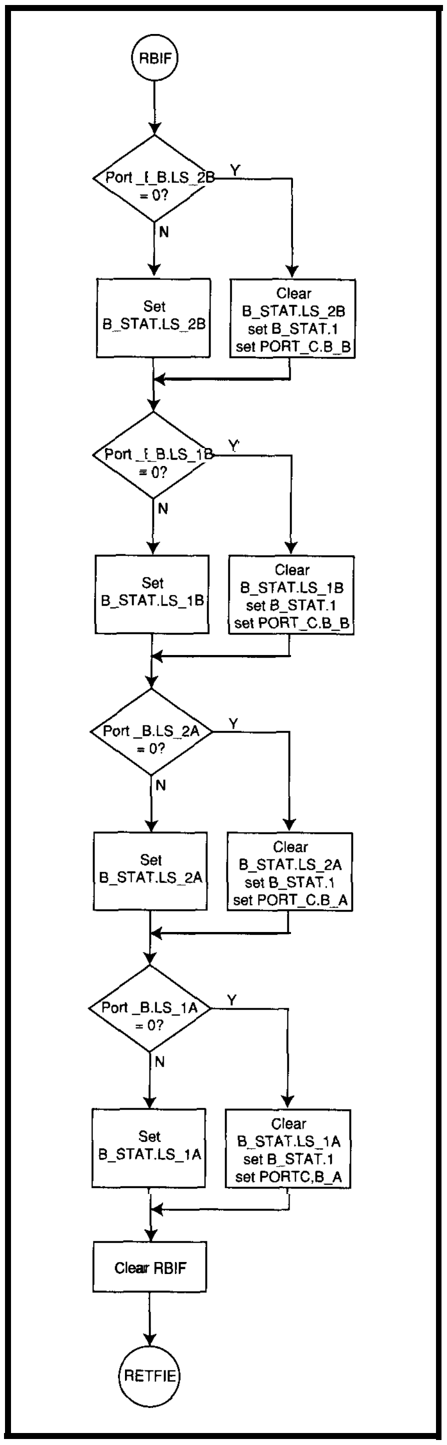
Most transistors have considerable drop even when driven into saturation. This drop can be in excess of 1 V, which represents a lot of wasted heat. DMOS switches reduce this waste to under 0.5 V and provide faster switching times than bipolar transistors.

To reverse the rotational direction of a PM motor, you must reverse the DC power source. This requires four switching devices set up in an H configuration (Figure 1d}. Only two diagonal devices can be switched on at once.

While one set is on, current flows through the motor in one direction. When the opposite set is on, current flows in the opposite direction. If devices on the same side were inadvertently turned on together, they would create a short across the DC source, resulting in high currents and self-destruction.

Fortunately, no one writes code which could cause this to happen. But, if you're concerned, you can prevent this by using a device that contains the H-bridge in one discrete package.

National Semiconductor's LMD-18200T is just the beast. It comes in an 11 -pin power SIP. And no, it's not new; just packed with features.



RBIF
Port _I_B.LS_2B = 0?
Y
N
Set B_STAT.LS_2B
Clear B_STAT.LS_2B set B_STAT.1 set PORT_C.B_B
Port _I_B.LS_1B = 0?
Y
N
Set B_STAT.LS_1B
Clear B_STAT.LS_1B set B_STAT.1 set PORT_C.B_B
Port _B.LS_2A = 0?
Y
N
Set B_STAT.LS_2A
Clear B_STAT.LS_2A set B_STAT.1 set PORT_C.B_A
Port _B.LS_1A = 0?
Y
N
Set B_STAT.LS_1A
Clear B_STAT.LS_1A set B_STAT.1 set PORTC.B_A
Clear RBIF
RETFIE

Figure 3a—*The PIC16C73 can generate an interrupt whenever a Port B input bit changes state. By using an interrupt service routine instead of having to poll the port, code is simplified. On the Smart-MD, the motor limit switches are connected to Port B inputs.*

LMD18200T

The H-bridge is made with DMOS power transistors for lower drop and faster switching speeds. Each switch contains its own intrinsic protection diode and is made up of many parallel devices which equally share total current.

Because of this sharing, total device current can be measured by directing a single cell's output to a current-sourcing output. This setup gives the user a way to measure motor current without adding a series resistor to the circuit and consequently losing precious power. The current-sourcing output gives 377 µA for 1 A of motor current.

Three input bits control the H-bridge: direction, PWM, and brake. The direction bit controls which diagonal pair of switching devices is used and switches the appropriate upper device on. This switching controls the rotational direction of the motor by controling the direction the current passes through the motor.

A logic high on the PWM input enables the lower device so full current flows through the motor. A logic low disables the lower device and interrupts the current flowing through the motor.

The last input, the brake bit, shorts out the motor by turning on either the top or bottom two devices, depending on the direction bit.

One disadvantage of using DMOS over bipolar transistors is they require their gates to be about 10 V higher than their source voltage to fully turn on. When you're looking for simplicity, this requirement frequently complicates the circuitry. However, the LMD18200T has built-in charge-pump circuitry which creates the necessary control voltages so the interface can be TTL compatible.

Although the PWM input bit can be a simple logic high to turn the motor on, applying a PWM signal gives speed and torque control of the motor.

The bit can operate in two modes. The normal mode is called *sign/magnitude.* Here the direction bit directly controls the direction, while the PWM controls the speed or torque from full on (100% duty cycle) down to full off (0% duty cycle).

The alternate mode is the Locked AntiPhase. In this mode, the PWM input bit is held at logic high while a PWM signal is applied to the direction input bit. A PWM input with a 50% duty cycle, although enabling the motor in opposite directions on each half cycle, averages a current flow of zero and thus has no movement.

Changing the duty cycle one way or the other unbalances the average current, which controls the motor's speed, torque, and direction, all from the same single signal (see "NC-based Motor Speed Controller" by Chuck McManis, *INK* 60).

I chose to use sign/magnitude control because it offers the same PWM signal with twice the resolution.

By now you've seen the advantages of using a monolithic drive controller, but I want to mention some of the other built-in advantages. The LMD18200T has overcurrent protection. It shuts off the upper H-bridge devices whenever (if ever) the current reaches 10 A. The device checks for this fault every few microseconds and clears itself if the fault has been removed.

Since the DMOS devices require the charge pumped 10 V over the source voltage, there's a low-voltage lockout which only permits operation when the source voltage exceeds 10 V.

If the device reaches a temperature of 145°C, an output bit is pulled low to signal the user of impending doom. Should the temperature rise to 170°C, the LMD18200T automatically disables all four DMOS devices, preventing catastrophe. A little built-in hysteresis reactivates the H-bridge when the temperature falls.

PIC16C73

I chose the PIC16C73 because it contains an I²C interface, an 8-bit A/D converter, two hardware PWMs, and a total of 22 I/O lines.

The I/O lines are broken down into ports: AO-5, BO-7, and CO-7. Port A is used as analog inputs to the A/D converter. I am using three analog inputs: two for motor-current sensing and one for overtemperature input (see Figure 2). I use the remaining bits of Port A as digital I/O (these will be discussed in more detail next month].

Port B can be configured with weak pullups on all of the bits used as inputs. This configuration avoids the need for an external resistor SIP. The upper four bits on Port B are limit-switch inputs and are treated as change-of-state interrupts. The next three lower bits of Port B are address-selection inputs. Any jumpers placed on the address selector, JP1, ground the input pins and can be read as logic 0. JP1 lets up to eight of these Smart-MD modules be used in one system.

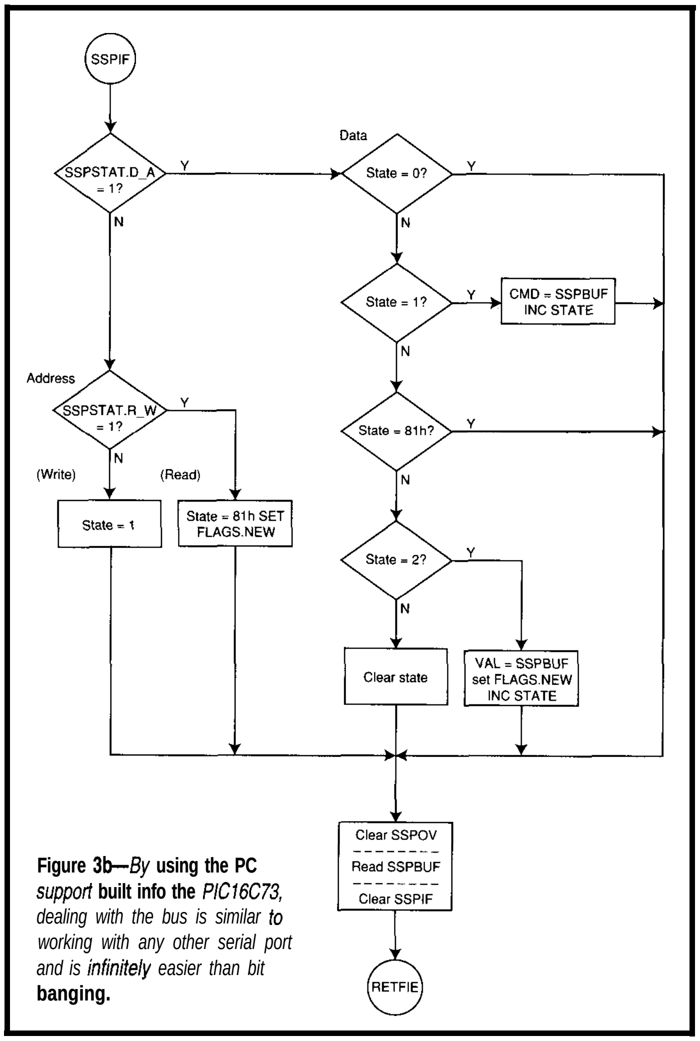


Figure 3b—*By* **using the PC** *support* **built info the** *PIC16C73, dealing with the bus is similar to working with any other serial port and is infinitely easier than bit* **banging.**

Port C has the PWM outputs, the I²C bus connections, motor direction, and brake controls for each motor. Many of Port C's pins are multipurpose. However, once you define them in your application, the possibilities are reduced considerably. Unless a need arises, I like to use a 4-MHz crystal for the simple reason that instructions execute in 1 µs, which makes cycle counting easy.

I²C

The basic I²C transmission consists of an address byte and a data byte. Within the address byte, the upper 7 bits contain the actual address while the least-significant bit is used as a read/write indicator. While this simplifies the protocol, it also limits the number of devices on the bus to a maximum of 128.

Since the I²C protocol was first introduced, enhancements have been made to increase the available addresses and throughput. A particular set of addresses (11110xx) can indicate that the byte following contains an additional 8 bits of address information. I can't imagine using more than 128 devices [where have I heard this before?), so for now, I'm not supporting the expanded mode.

There is a slight difference between the write and read transmissions. Many I²C peripherals are quite dumb and require only one byte of data. The smarter (more complicated) ones have multiple registers for the data. This situation requires that an additional byte be sent when writing (address, register, and data bytes). More data bytes can also be sent; however, this is predetermined by the device.

When reading, the I²C protocol requires the slave to respond right after the address, leaving no room for a transmitted register byte, which is where transmissions differ. Immediately after an address byte (where the LSB is a 1, indicating a read), the master (clock source) looks for a reply, so the last register set up by a write is used as the selected register to read from.

The Smart-MD interprets the register bytes as commands. Its address is set as 0000xxx, where xxx is read from the external address jumpers. When its address is recognized and includes a write, the Smart-MD accepts the following register and data byte. The register byte is interpreted as a command. Each legal command allows some kind of action to be taken using the data byte. Table 1 offers a simple command set.

TWIN PWMS

The PIC16C73 has two complete capture, compare, or PWM modules. When used in PWM mode, the PR2 register is used as the base counter and acts as a reference to control the PWM frequency.

In standard-resolution mode, up to 8 bits can be loaded into the PR2 register. In high-resolution mode, which concatenates the two internal Q clock bits, the PR2 register can create a 10-bit base.

Each PWM module has a timer which is continually compared to the PR2 register and resets when a match occurs. This operation produces a logic 1 on the PWM output pin. Since the PR2 register does not change, the PWM output pin always goes high in exactly the same number of counts (constant period).

A 16-bit register pair (CCPRlL and CCPRlH) creates the actual duty-cycle timing. The user loads a number into the CCPRlL register. Whenever the timer resets, the value in the CCPRlL is loaded into the CCPRlH. This action prevents unwanted glitches from occurring on the PWM output when a duty cycle is altered midcount.

The CCPRlH and the timer are also compared. However, a match on this register pair resets the PWM output to a logic 0. You can see how altering the value loaded into CCPRlL register affects how long the PWM output stays high. Smaller numbers shorten the on time, while larger numbers lengthen it. The value placed in the CCPRlL register automatically reloads CCPRlH every cycle for hands-off operation.

Whenever a write command (P or p) is received, the top bit of the data value is placed on the direction control output bit for the associated motor (P=motor A and p=motor B). The remaining 7-bit value is multiplied by 2 (shifted left once] and placed into the appropriate motor's PWM CCPRlL register. This command controls both the direction and the duty rate for one or the other motor. A read command passes back the data value it last received.

BRAKING

Each LMD18200T has a separate input bit for emergency braking. This command stops the motor even if the direction and PWM bits call for movement.

The brake write command (B orb) enables or disables the brake bit of the associated motor driver, depending on the data value received. Any value other than 0 turns on the brake for that motor driver.

When a read braking command is received for either motor, the status register is returned giving more information than just the brake status. The upper four bits contain the current status of the two limit switches for each motor. These inputs are normally high and are grounded when a switch is tripped.

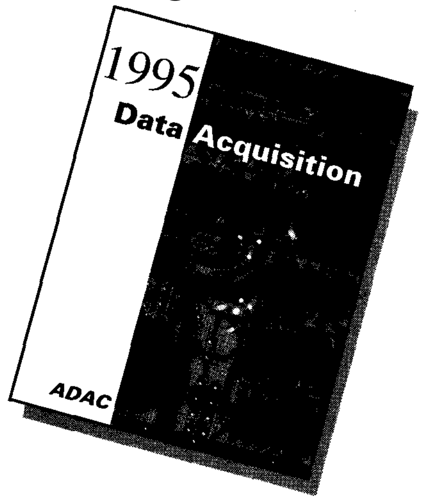
The next two bits indicate the current temperature of each motor driver. Each bit is cleared when the temperature of its corresponding driver exceeds 140°C. The final two bits indicate the present states of the brake output bits for each of the motors. The same status register is sent regardless of whether the command was B orb.

A few words on the limit switch inputs are called for. On the PIC, a change of state on any Port B bit set for input can generate an interrupt. This feature is used to our advantage by letting any limit switch automatically brake its associated motor driver. Even though the brake bits are automatically set, the user can remove them through a brake-write command. Care and logic must be used to avoid continuing past the limit switch.

CURRENT STATUS

Motor current gives the user feedback on the status of the motor. It not

FREE
Data Acquisition Catalog



PC and VME data acquisition catalog from the inventors of plug-in data acquisition. Featuring new low-cost A/D boards optimized for Windows, DSP Data Acquisition, and the latest Windows software. Plus, informative technical tips and application notes. Call for your free copy

1-800-648-6589

ADAC

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

#123

only indicates the speed of the motor, but also how hard it's working. Excessive current suggests a stall, which can lead to overheating. The user may wish to adjust the motor's PWM duty cycle settings based on these current measurements.

The LMD18200T's current output pin generates 377 µA per amp passing through the motor. A resistor from this pin to ground converts the output current to voltage. You should choose a resistor value which produces a 5-V drop at motor-stall currents to prevent exceeding the A/D converter's 5-V maximum input.

Let's say the motor's stall current is 3 A. The current output produced for 3 A is 1.13 mA (3 × 377 µA). The maximum input for the A/D converter is 5 V. This value is divided by the output current (i.e., 5 V / 1.13 mA) to produce 4,424 Ω.

Selecting the closest common resistor value less than our result (4.3 kΩ), the maximum current is fixed at 1.16 mA. With 8 bits of resolution, that's (1.16 mA / 0.377 mA / 256) or 12 mA of actual motor current per bit.

Command A/a (Analog to Digital Converter)

Address	Register		Data	Action
OOOOxxxl	01000001	'A'	xxxxxxxx	Read Current Motor A
OOOOxxxl	01100001	'a'	xxxxxxxx	Read Current Motor B

where data = 8-bit value of Motor current

Command B/b (Brake and status of Motor A/B)

OOOOxxxl	01000010	'B'	xxxxxxxx	Read LS, TF, Brake A/B
OOOOxxxl	01100010	'b'	xxxxxxxx	Read LS, TF, Brake A/B

where data = xxxxxxxx
bit 0: Motor A brake, 1 =ON 0=OFF
bit 1: Motor B brake 1 =ON 0=OFF
bit 2: Motor A temperature, 1=>140°C 0=<140°C
bit 3: Motor B temperature, 1 =>140°C 0=<140°C
bit 4: Motor A limit switch 1,1=OK 0=TRIPPED
bit 5: Motor A limit switch 2, 1=OK 0=TRIPPED
bit 6: Motor B limit switch 1, 1 =OK 0 TRIPPED
bit 7: Motor B limit switch 2, 1 =OK 0=TRIPPED

OOOOxxxO	01000010	'B'	xxxxxxxx	Enable/Disable Brake A
OOOOxxxO	01100010	'b'	xxxxxxxx	Enable/Disable Brake B

where data = 0 = OFF 1 – 255 = ON

Command P/p (Direction and PWM control of Motor A/B)

OOOOxxxl	01010000	'P'	xxxxxxxx	Read PWM of Motor A
OOOOxxxl	01110000	'p'	xxxxxxxx	Read PWM of Motor B

where data = Oxxxxxxx is forward
1 xxxxxxx is reverse
where xxxxxxx = PWM in 1/128 increments (O=O% 128=100%)

ooooxxxo	01010000	'P'	xxxxxxxx	Set PWM of Motor A
ooooxxxo	01110000	'p'	xxxxxxxx	Set PWM of Motor B

where data = present PWM value

Table I--The *Smart-MD module supports three basic commands.*

PIC CODE

The flowchart in Figure 3 gives an overview of what goes on behind the scenes. There are basically two interrupt routines. The first is the Port B change-of-state interrupt. This interrupt protects the mechanics of the motor apparatus by shutting down motion which oversteps the boundaries of the physical system.

The second interrupt routine is the I²C communication. Illegal and unnecessary interrupts are prevented by the hardware. Only transmissions which match the module's address can create an interrupt. Until a stop bit is reached, this and any data following causes interrupts. Each interrupt is serviced by analyzing the PIC's SSPSTAT status bits and determining the present state of the system. Prior to any reception, STATE is zero.

Once an address match occurs, STATE is set to Olh for a write or 81h for a read. Since the address also determines the read/write status, I use the most-significant bit of STATE as an ID. As additional interrupts occur, STATE is incremented by 1 and the data is temporarily stored as CMD (when STATE=01h) or VAL (when STATE=02h).

The main program loop looks for a STATE of 03h or 8 1h. When either is found, the CMD register is used to identify which subroutine should be executed. When finished, the state is cleared and a jump is made back to the loop to await further instructions.

Although the A/D converter can initiate an interrupt, I don't use it here. Since an A/D conversion is only performed when a read (A or a) command is received, a simple poll of the conversion bit *DONE is used for simplicity. Total settling and conversion time for full 8-bit resolution is only about 14 µs.

PARTIAL CONCLUSION

Using only three devices (a PIC and two LMD 18200T H-bridge motor drivers), a Smart-MD module can control two 3-A motors from an I²C bus. This setup is perfect for robot control where the robot's motor needs to be stopped and started by its sensors.

If you need accurate positioning, catch next month's column. There, decoder circuitry tracks the motor's position.

Until then, happy motoring. ❑

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.*

SOURCES

PIC16C73
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199
(602) 786-720
Fax: (602) 899-9210

LMD18200T
National Semiconductor Corp.
P.O. Box 58090
Santa Clara, CA 95052-8090
(800) 272-9959

Digi-Key Corp.
P.O. Box 677
Thief River Falls, MN 56701
(800) 344-4539

I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

EMBEDDED PC

48 **Nouveau PC**

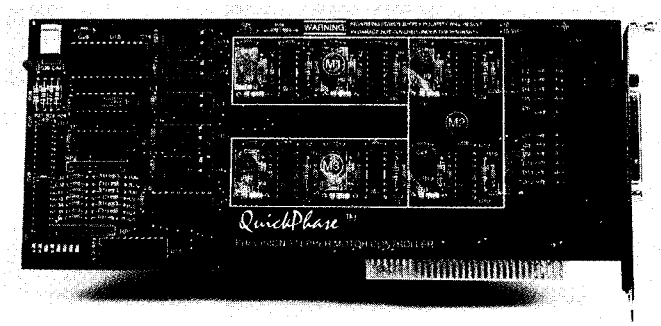
54 **An ISA Bus Simulator**
Stefano Chiti-Batelli

62 **Hot Swapping on the PC**
An Active Extender Card for the 16-bit PC-ISA Bus
David Prutchi

72 **PC/104 Quarter**
Constructing PC/104-based Systems
Rick Lehrbaum

76 **Applied PCs**
Embedded PCs
Who, What, Where, Why, When, and How?
Russ Reiss

PC-BASED MOTION CONTROLLER



Quick-Phase from MicroKinetics is a new stepper-motor controller containing onboard translators and power drivers for up to three axes all on one card. QuickPhase plugs directly into any 8-or 16-bit ISA bus IBM or compatible PC, thereby eliminating the need for external drivers. QuickPhase is ideal for many applications including scientific instruments, CAD/CAM modeling, and robotics.

Features of the controller include programmable acceleration and deceleration, automatic over-temperature protection, end-of-travel detection on all axes, two auxiliary outputs per card, and a “safety shield open” interrupt.

The software is very easy to incorporate into any application. The subroutine libraries provided support C and QuickBASIC and include linear and circular interpolation, ramping, keyboard interactive jog, and electronic gearing.

QuickPhase sells for $339 for 2 axis and $389 for 3 axis. A 5-year parts and labor warranty is included.

MicroKinetics Corp.
1220-J Kennestone Cir. • Marietta, GA 30066 • (404) 422-7845 • Fax: (404) 422-7854 **#510**

‘486 SINGLE-BOARD COMPUTER

Megatel announces a PC/104-compatible **i80486 OEM** Development Kit. The PC/**II+iOEM2-4/104** includes a single-board computer, break-out module, and software.

The heart of the OEM2-4/104 kits are the microcomputers. The PC/II+i features a '486DX processor or the ‘386SLcompatible PC/II+. Operating at either 25 or 33 MHz, these boards include Local Bus SVGA video or LCD controller, Ethernet interface, SCSI, floppy, and serial or parallel ports, 2 MB of flash disk, a full 32-bit DRAM data bus, and an AT-compatible BIOS. The PC/II+i also offers a built-in 8-KB cache with floating-point unit. CMOS reduces power consumption to approximately 6 Wand +5 V.

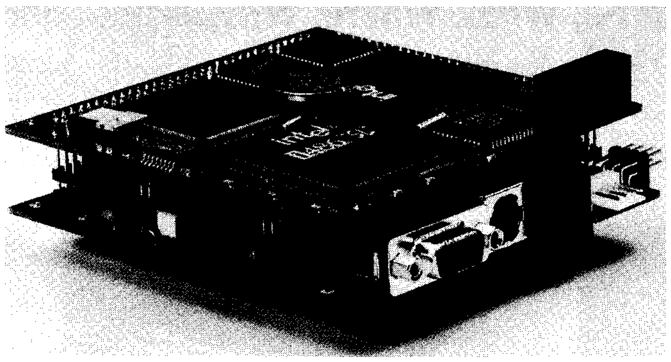
The PC/l 04 break-out modules, known as QTBs, make it possible to connect the SBC to a standard PC/l 04 format. Connections from the QTB module to single-board computers are simplified by the use of a 96-pin peripheral I/O connector and a 64-pin ISA bus connector. If the Ethernet option is installed in the PC/II+i, the QTB/104 and QTB/104AT can also be connected by ribbon cable to the 1 O-pin Ethernet header. The small PC/l 04 format lets you add many available stackplane modules.

All peripheral I/O signals from the 96-pin I/O connector are attached to appropriate peripheral headers or connectors on the QTB/104 or QTB/104AT. All ISA bus signals from the 64-pin ISA bus connector are converted and connected to the standard PC/1 04 bus or ISA edge connector.

The PC/II+i Ethernet interface uses the Intel 82595 single chip Ethernet controller and its buffered slavearchitecture. The boards provide 16-bit bus operation and are fully 802.3 compliantwith **the** AUI and TPE serial interface. Software for the Ethernet interface includes full Novell IPX/SPX support.

A complete legal BIOS in flash memory is provided on either board. It boots standard versions of PC, MS, or Novell DOS. An onboard ROM DOS is also offered. As a result, the PC/II+i runs most popular PC software packages, including Windows 3.1. A copy of a high-performance flash-file subsystem, which provides the onboard capability for a 2-MB solid-statedisk, isalsoincluded.

All OEM/l 04 kits include the SBC board, QTB/104 break-out module, 1- or 2-MB flash disk BIOS, DRAM, board jacket, ROM, Ethernet, and SCSI utility software, cable set, and manuals. The price of the PC/II+iOEM2-4/104 kit is $1495 and the PC/II+i board is from $1 195.



Megatel Computer Corp.
125 Wendell Ave.
Weston, ON
Canada **M9N 3K9**
(416) 245-2953
Fax: (416) 245-6505

#511

Nouveau-PC

edited by Harv Weiner

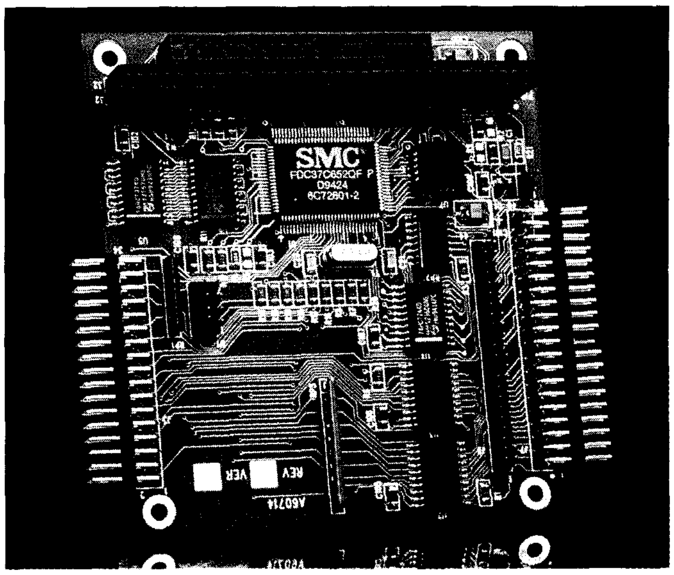
PC/ 104 DISK CONTROLLER

Ampro introduces a low-cost PC/104 module that lets IDE hard disk drives be used with their 8-bit PC/104 embedded PC CPUs. The **Mini-Module/Fl** includes interfaces for both floppy and IDE disk drives and is compliant with the PC/l 04 V.2 specification for compact (3.6" x 3.8") embedded PC modules.

The module is equipped with a 16-bit PC/ 104 self-stacking bus, but can be used in both 8- and 16-bit systems. Special circuitry is included to facilitate the use of normal (16-bit) IDE disk drives in an 8-bit bus system. The necessary BIOS support required for IDE drive operation in 8-bit systems is included within the standard ROM-BIOS on Ampro's 8-bit PC/104 CPU modules. The module operates from a single +5-V supply and supports a wide operating temperature range of 0–70°C, with extended temperature ranges available on special order.

The MiniModule/Fl is priced at $85 in quantities of 100.

#512



PC/ 104 DESIGN GUIDANCE

Also available is a free white paper called "Designing with PC/l 04-A Tutorial." The 14-page tutorial includes an overview of the PC/l 04 standard, describes how PC/l 04 modules are used in typical embedded applications, and provides guidelines for effective and efficient use of PC/l 04 technology. The paper was written by Rick Lehrbaum, Ampro's cofounder and president, who is generally regarded as the father of PC/l 04.

Ampro Computers, Inc.
990 **Almanor** Dr. • Sunnyvale, CA 94086
(408) 522-2100 • Fax: (408) 720-l 305 **#513**

SINGLE-BOARD COMPUTER

Real Time Devices has introduced the **CMi386SX** and **CMi486SLC** PC/l 04compliant cpuModules. These fully integrated PC/AT-compatible single-board computers with an onboard math-coprocessor socket reduce the number of modules required for PC/104 industrial control and data processing.

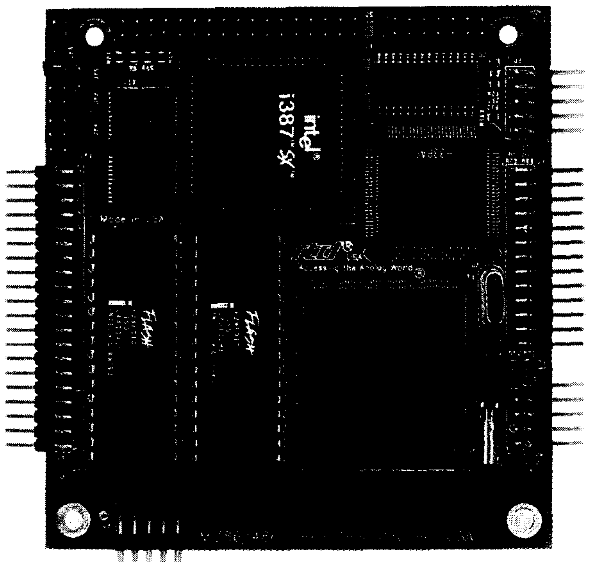
The CMi386SX and CMi486SLC feature an Intel '386SX 25-MHz or TI '486SLC 33-MHz processor with programmable clock rates, onboard expansion socket for optional 80387SX math coprocessor, 2-MB or 4-MB DRAM installed, and two 32-pin sockets for 2-MB EPROM or 1-MB flash solid-state disk. Also included are 16-bit IDE and floppy disk controllers; one RS-232 serial port; one RS-232, RS-422, or RS-485 serial port; bidirectional parallel port; AT keyboard; PS/2 mouse; speaker ports; and a watchdog timer.

Services and functions provided by the BIOS ensure PC/AT compatibility. The BIOS also includes SSD support for SRAM, EPROM, flash memory, NVRAM, and virtual devices. Virtual device support enables the operator to use peripherals connected to another PC- or AT-compatible computer without interfacing them directly to the module's PC/l 04 bus.

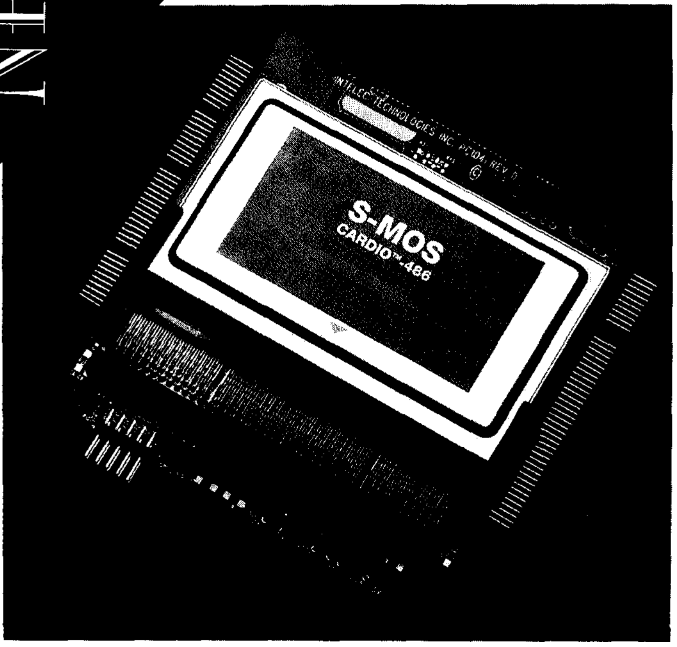
Connection is made through the serial port. A nonvolatile 1024-bit configuration EEPROM stores the system setup and provides 5 12 bytes to the user. Direct plug connection to other modules is made through a standard PC/l 04 stack connector.

Prices start at $578 in quantity.

Real Time Devices, Inc.
200 Innovation Blvd. • State College, PA 16803
(814) 234-8087 • Fax: (814) 234-5218 **#514**



Nouveau-PC



SMALL PC/I 04 MODULE

The S-MOS **CARDIO/Intelec** PC **104i** board is a PC/I 04-compatible single-board computer. This board enables the user to combine the small-size advantage of the CARDIO with the stackable PC/I 04 peripheral products. The 3.6″ × 3.8” board meets the one stack mechanical and electrical PC/I 04 Rev. 2.2 specifications.

The S-MOS CARDIO is a fully functional IBM PC/AT computer about the size of a credit card (3.4″ × 2.2”). Versions are available that include the Intel ‘386 or ‘486SL and Chips and Technologies 8-bit 8680. Upgrading from an ‘x86 to a ‘386 or ‘486 is easily accomplished by replacing one CARDIO module with another. The PC1 04i is a small, PC-compatible single-board computer that achieves more functionality and a greater choice of peripherals, software drivers, operating systems, and application libraries.

The PC1 04i board and the S-MOS CARDIO provide two serial ports, a mouse or keyboard controller, floppy and hard disk controllers, flash BIOS, an LCD controller, and power-entry connector. Also available are compatible cables for standard connections, watchdog capability with an LED indicator, a power-fail indicator, battery backup, and automatic switchover (external battery re quired). The system features low EMI/RFI and low power consumption with 3.3-V power-supply generation onboard.

The PC 104i starts at $290 (q uantities of 100), the S-MOS CARD10486 at less than $800 (in q uantity), and the ‘386 version at less than $400 (in quantity).

S-MOS Systems, Inc.
2460 North First St. • San Jose, CA 95131 • (408) **922-0200** • Fax: (408) 922-0238
E-mail: **jsilverman@smos.com** **#515**

NETWORKED EMBEDDED PC

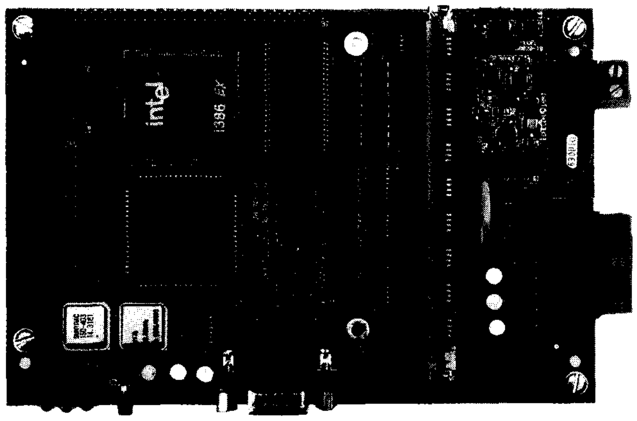
The **CA386-N1** from Coactive Aesthetics pairs the powerful 25-MHz Intel ‘386EX processor with Echelon’s 3 120 Neuron for peer-to-peer networkcommunications. The 3 120 running Echelon’s MIP program and Coactive’s libraries lets CA386-N 1 applications access data on any LONWORKS-compatible node and lets the C386-N 1 act as an embedded “compute engine” for the network.

The 4” x 6″ board is available with 78-kbps or 1.2-Mbps twisted-pair transceivers. The system includes 256 KB, 1 MB, or 4 MB of RAM and 5 12 KB of flash memory. It has eight digital I/O lines, three programmable counter/timers, two RS-232 serial ports, and an optional battery-backed RTC. The 8- or 16-bit PC/104 interface offers expansion with industry-standard peripherals.

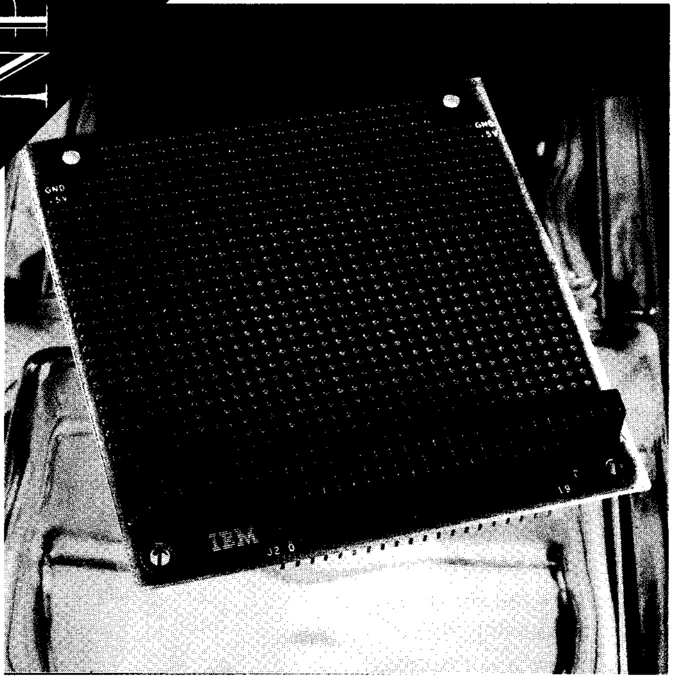
The OEM Software Development Kit for the CA386-N1 includes Intel’s iRMX real-time OS in flash along with utilities for downloading and running programs. Programs are compiled on a PC using Visual C. The package includes Coactive’s NetLIB, a C library for interfacing to LONWORKS. NetLIB lets users declare network variables and use them to communicate with other nodes.

The CA386-N 1 sells for $699 in single quantities.

Coactive Aesthetics, Inc.
4000 Bridgeway, Ste. 303 • Sausalito, CA 94965
(415) 289-I 722 • Fax: (415) 289-1320
E-mail: coactive@coactive.com **#516**



Nouveau PC



PC/I 04 **PROTO-TYPING** CARD

IBM has added a PC/104 card to its line of prototyping cards. The **IBM 91-010** is a PC/I 04 four-layer raw card with internal power and ground planes. Features include a 0.1 ″ grid of plated-through holes of 0.04" diameter for component or socket mounting, peripheral power buses, and fanned-out signal connection from the J 1 and J2 connectors. The connectors are installed and nylon standoffs are included.

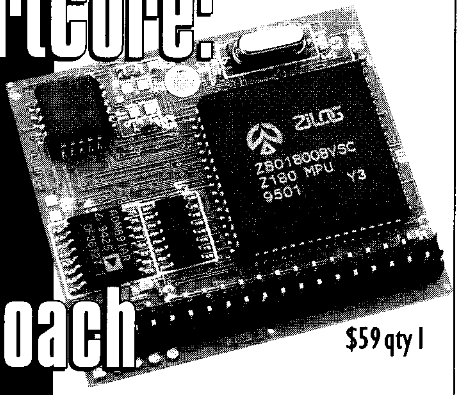
The card is in stock at Marshall Industries and is available from Bell Industries ond Bell Micro Products Distributors. The single-quantity price is $1 10.

IBM Corp.
Microelectronics Div.
1701 **North** St.
Endicott, NY 13760
(607) **755-6565**
Fax: (607) 755-6562

#517

Nouveau PC

◆ SmartCore:™ ◆ the ◆ New ◆ Approach



$59 qty l

$99 Evaluation Kit. See how the SmartCore approach can save you time and money. You'll receive a fully featured SmartCore, trial version of Dynamic C software, evaluation board, accessories, and documentation. Call today for immediate delivery.

The SmartCore™ is a shortcut to a custom-designed controller. It is a C-programmable microprocessor core with memory, supervisor, DMA, and clock built in. All you do is add interface and control logic. Call our AutoFAX 916-753-0618 from your FAX. Ask for data sheet #34.

Z WORLD ENGINEERING

1724 Picasso Ave.
Davis, CA 95616
916.757.3737
916.753.5141 FAX

#202

REMOTE POWER CARD!

3 VERSIONS:

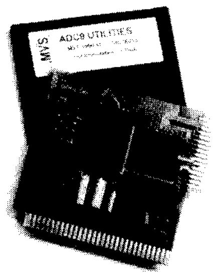
RESET
WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

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

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

95$ 27$ OEM

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



8 CHAN ADC
DATA AQUISITION, SERVO CTL, AUDIO
8-BIT RESOLUTION 22KHZ SAMPLE RATE
SHARP CUTOFF ANTI-ALIAS FILTER
CREATE STEREO BLASTER (.VOC) FILES
95$

2 CHAN DAC
VOICE MAIL, MUSIC, ALARMS, CTL VOLT
8-BIT RESOLUTION 44KHZ SAMPLE RATE
PLAYS MONO / STEREO BLASTER FILES
FUNCTIONS AS DIGITAL ATTENUATOR TOO
75$

MVS
MVS BOX 850
MERRIMACK NH
(508) 792 9507
5 YEAR LIMITED WARRANTY
FREE SHIPPING IN USA

#203

Stefano Chiti-Batelli

An ISA Bus Simulator

You could simply plug in your prototype. However, design flaws can prevent operation or cause permanent damage. Stefano's ISA bus simulator gives the engineer more control over the *development* and test process.

Would you like to develop an interface board for the ISA bus?

Here's a solution to your needs-the PDI (Prototype Development Interface), a full-featured ISA bus simulator that should help you understand and test the inner workings of the most common type of PC expansion boards.

The PDI ISA bus simulator consists of two separate boards: a fairly simple I/O expansion board for the PC (PDI internal module) and an external module. The external module relies on the internal board for its power supply and I/O control lines. Software controls simulator features.

Operating thesimulator is fairly simple. First, you plug the to-be-tested-board (TBTB) into the slot provided by the external PDI module. Next, you run the PDI debugger to test the TBTB.

With the debugger, you can operate at two different levels. You can set the status (high or low] of every single line of the ISA bus or you can take advantage of the built-in macro function to automate repetitive and/or tedious tasks which otherwise require manual control of many different bus lines (e.g., read and write cycles).

The debugger itself takes advantage of this feature. For example, normal I/O operations on the ISA bus (e.g., input from

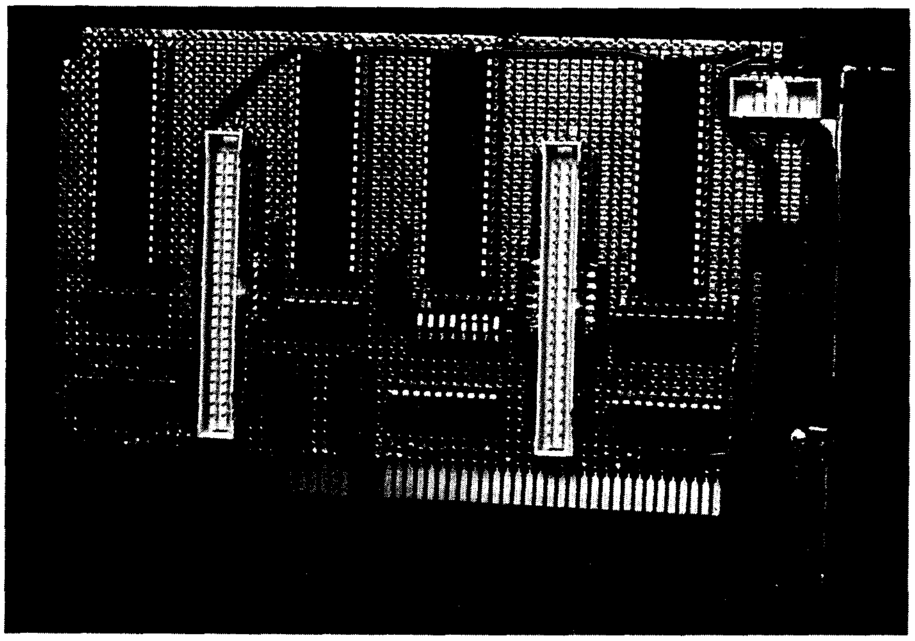


Photo 1: The **PDI** internal module offers a **16-bit I/O** expansion board of up to 96 lines.

or output to I/O ports) aren't hard coded in thesoftware. Instead, they are implemented via macro functions which can be easily modified using any ASCII editor.

The **PDI** internal module is essentially a digital I/O expansion board, hosting four 8255 chips for a total of 4 x 24 = 96 I/O lines. These boards are commercially available (approx. $100). If you are willing to apply the small modifications needed to connect to the PDI external module (power supply lines and a couple of "special" lines such as RESET), you can skip the soldering required for the internal module and concentrate your efforts on the external module. The software should be easily adapted to slightly different hardware.

INTERNAL MODULE

As mentioned, the PDI internal module is a fairly simple 16-bit expansion/interface board for the ISA bus (see Photo 1). **It** provides the I/O lines needed to interface to the external PDI module. You need 64 (65 if you want manual RESET) I/O lines. This number can be lowered if you reduce the features of the PDI so it only accepts 8-bit ISA boards in its simulated slot (see later discussion).

The schematic of the internal module is provided in Figure 1. Since you can buy a **functionally** equivalent board, I'll only briefly describe this board. The simulator software takes the possibility of using a different PDI internal module into account-all I/O operations are performed by a limited number of routines.

The interface board is a 16-bit ISA expansion board. As such, it uses the 1016 line on the 16-bit extension connector to the ISA bus. This line serves as input to the PC and is shared by all 16-bit boards plugged into PC slots. Although it could be open-collector driven or tristated, I chose to use tristated.

Chip U6 compares addresses with the value set by DIP switches (switch closed = low address bit). Its output line SEL is the main activation switch for the rest of the circuit-when low, it's your turn!

Since the board is 16 bit and the 8255 chips are only 8 bit, thev obviously work in pairs. One gets or provides data for **the lower byte of the 16-bit word, while the other chip works with the upper byte.** Couples are **formed by UO + U1** and by U2 + U3. I use 16-bit interfacing because I want to be able to switch (i.e., update with a new value) the whole simulated 16-bit data bus **in a single write operation. Using a commercial 8-bit board sacrifices this option.**

Each 8255 needs four addresses, so we need a total of sixteen addresses to talk to our board. In compliance with the space reserved by IBM **for prototype expansion boards, possible I/O base addresses are all even values between 300h and 3 1 Oh.** I **chooseaddress 3** 1 Oh, **which I use throughout the rest of this article.**

Remember, since our board is 16 bit, we only reference even addresses. For example, with a base address of 3 1 Oh, we **address I/O ports at 3 1 Oh, 3** 12h, 314h,

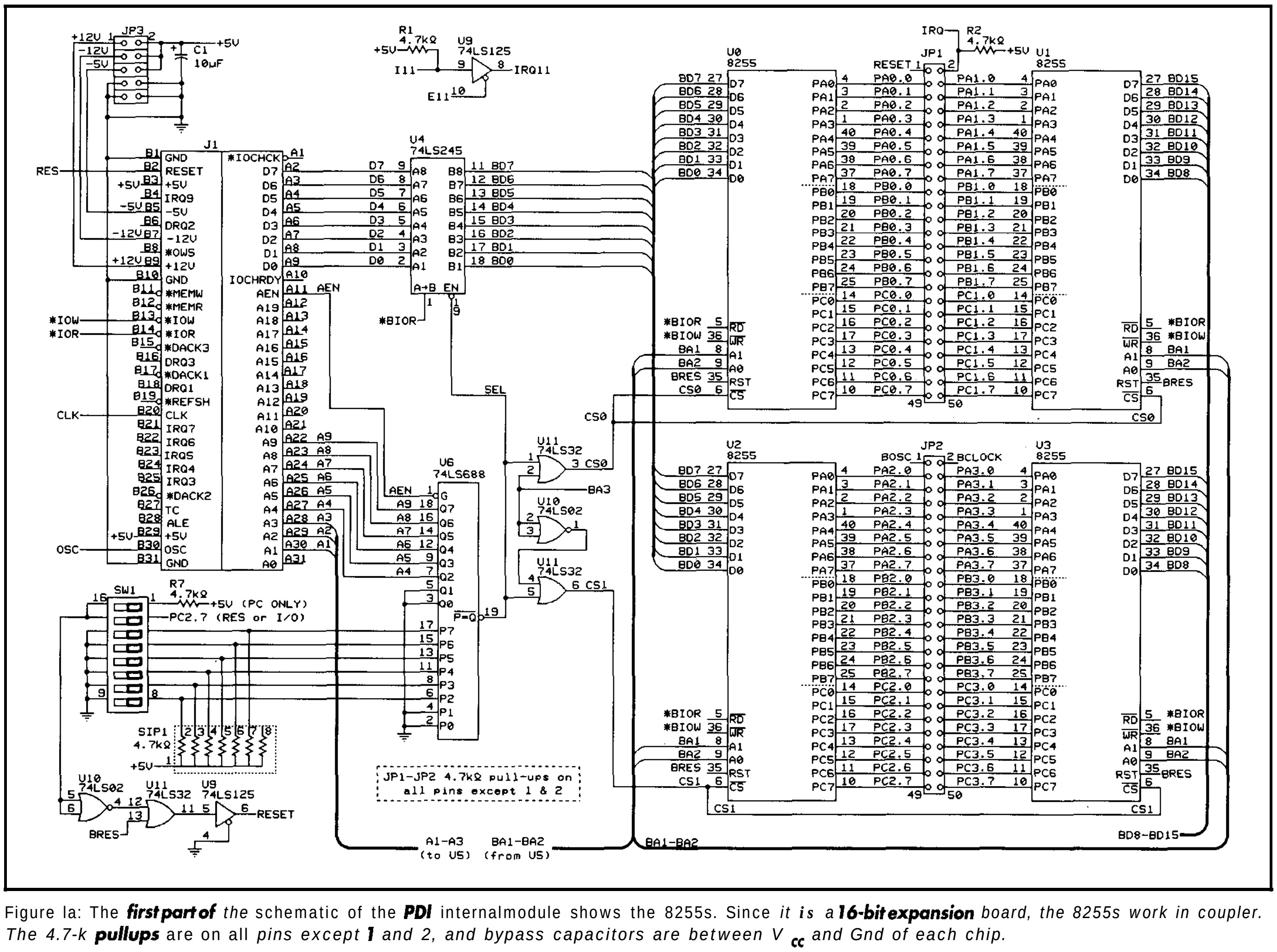


Figure la: The ***first part of*** *the* schematic of the ***PDI*** internalmodule shows the 8255s. Since *it is a* ***16-bit expansion*** *board, the 8255s work in coupler. The 4.7-k* ***pullups*** *are on all pins except* ***1*** *and 2, and bypass capacitors are between V* $_{cc}$ *and Gnd of each chip.*

and so forth. The first pair of 8255s uses the first eight I/O addresses, so the second pair is referenced starting with address 318h.

Reset is a useful feature of any type of equipment. In this case, it is useful to:

a) manually drive the RESET line of the simulator slot and
b) automatically perform a power-on reset whenever the PC is turned on

Reset is accomplished by gates U10b, U11d, and U9b. Since a manual reset requires use of an I/O line, DIP switches can disable this feature, reverting it to a PC-only reset with no user activation.

Apart from I/O lines and reset, the connectors have pins for the real bus lines CLOCK and OSC. The external module provides jumpers so you can choose to connect real or simulated signals to the TBTB. I have done this, though some picky TBTB boards might mind if their clocks aren't fast enough.

last but not least, the internal PDI module provides two kinds of support for interrupts: externally activated or locally activated. The latter occurs most frequently in the case of 8255 chips being used in a mode other than Mode 0 (see 8255 datasheet). This use probably means that the PDI internal module is not used in conjunction with the external module, but for another kind of task. The externally activated interrupt refers to the IRQ line, which comes from an external device.

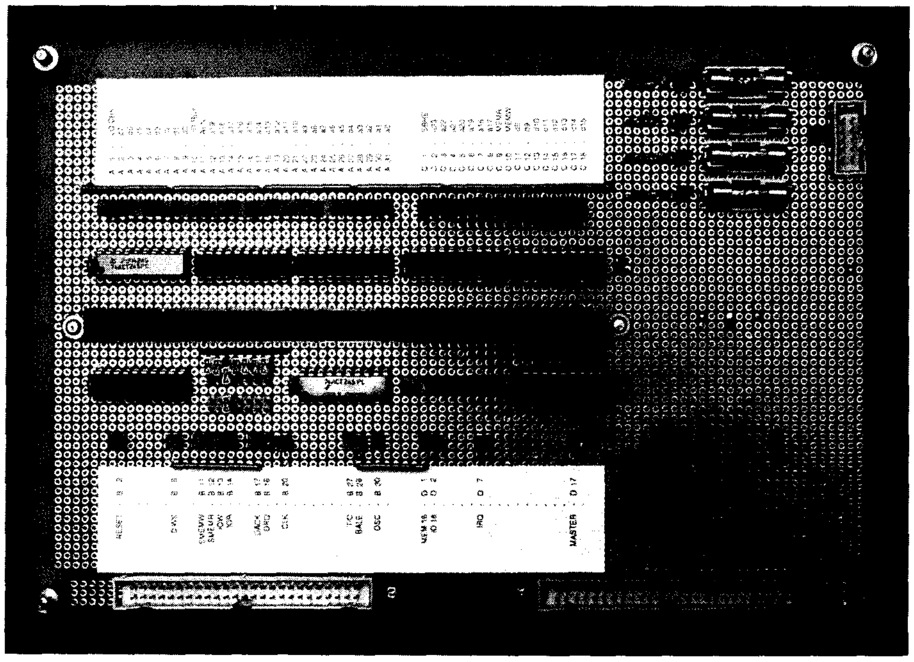


*Photo 2: The PDI external module holds the simulated **16-bit** ISA slot.*

Jumpers enable or disable the interrupt sources.

Since they aren't used for the simulation, you might wonder why I put these interrupt features in the PDI internal module. I wanted to design a general-purpose I/O expansion board, however I wanted to provide additional features. Since I did not need all 96 I/O lines, I designed for future applications that might include simulating or simply talking to the digital interface the TBTB board provides.

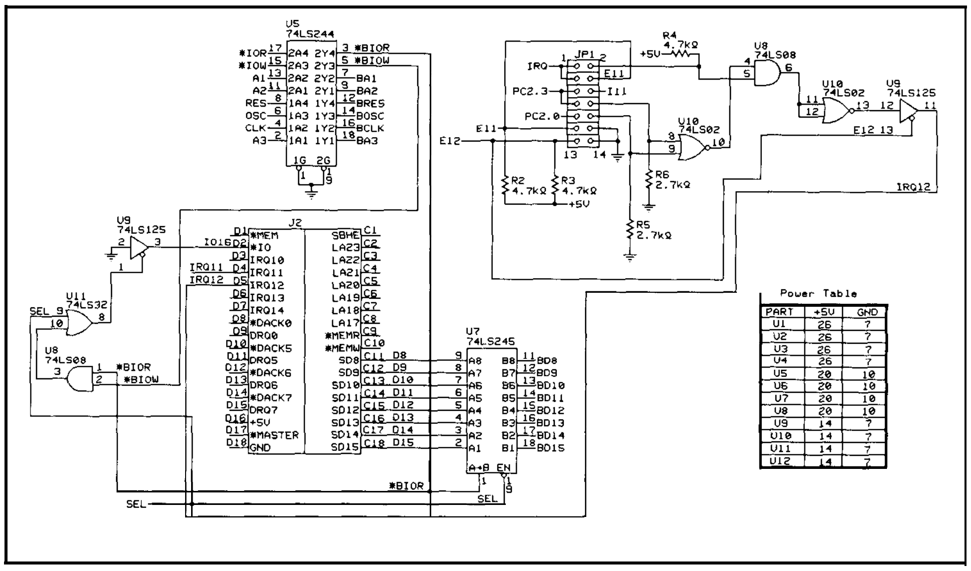
In this scenario, an interrupt feature [i.e., the IRQ line or a special configuration of the fourth 8255) is a valuable addition. If required, jumpers are provided to **enable** selected interrupt sources. These sources can be connected to interrupt channels 1 1 or 12 of the ISA bus.

Since we aren't using all 96 I/O lines, the free ones might be used for other simulation or interfacing tasks. For example, suppose you want to develop an LCD interface board for the PC. The PDI lets you simulate not only the ISA bus, but also the LCD itself, though only after a little extension of the software to allow control of the extra (above the "first" 64) I/O lines of the 8255s.

Since an external board may require a nontrivial amount of power, connect Gnd and Vc, (see the power supply flat-cable connector) to all contacts of the ISA bus which provide them. The PDI external module has fuses on all power lines.

EXTERNAL MODULE

The PDI external module is shown in Photo 2. I used a double-Eurocard-sized prototyping board. Apart from the 16-bit ISA slot and the flatcable connectors, the big show (literally) is provided by the many flat LEDs. They are powered by buffers which connect them to the ISA bus lines (see Figure 2). Since its pin ordering is more regular, I used the 74LS245 chips, instead of the 74LS244, as LED current buffers. With careful placement of the buffer chips and connectors [and perhaps use of the 245's DIR line), you can wire the simulated bus lines of the external



Power Table

PART	+5V	GND
U1	26	7
U2	26	7
U3	26	7
U4	26	7
U5	20	10
U6	20	10
U7	20	10
U8	20	10
U9	14	7
U10	14	7
U11	14	7
U12	14	7

Figure 1 b: The second part of the PDI internal module includes the **16-bit** bus signals and **IRQ selection.** *Line names **starting** with a "b" (e.g., bCL K) refer **to** buffered lines.*

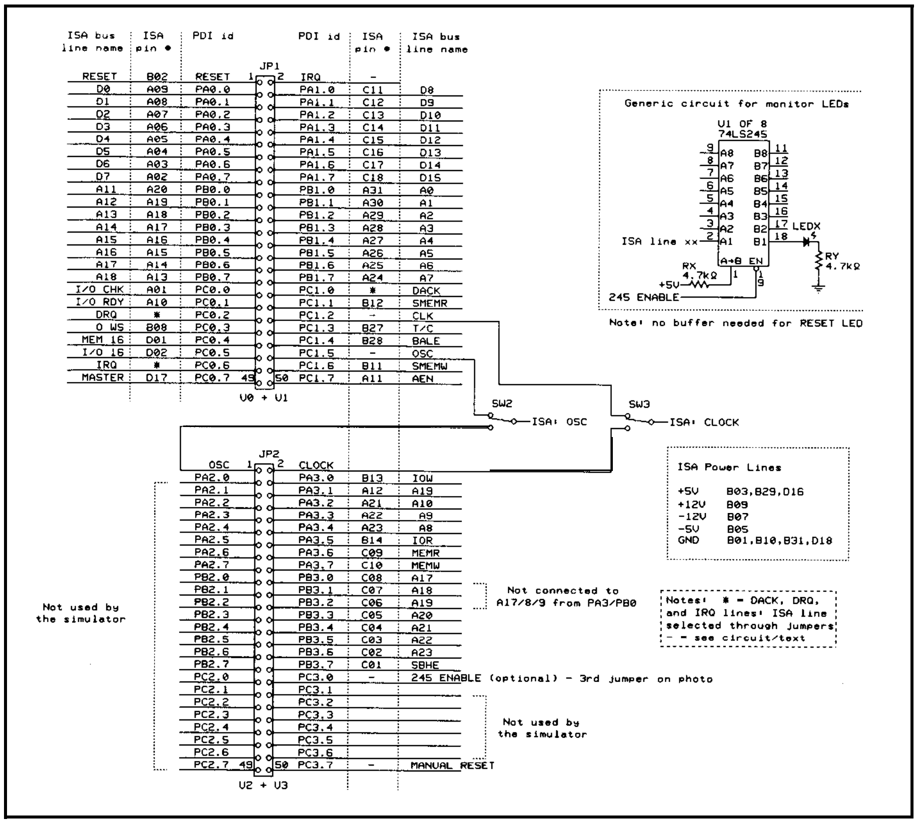

ISA bus line name
ISA pin #
PDI id
JP1
JP2
U0 + U1
U2 + U3
SW2
SW3
ISA: OSC
ISA: CLOCK
Generic circuit for monitor LEDs
U1 OF 8 74LS245
LEDX
RY 4.7kΩ
RX 4.7kΩ
245 ENABLE
Note: no buffer needed for RESET LED
ISA Power Lines
+5V B03,B29,D16
+12V B09
-12V B07
-5V B05
GND B01,B10,B31,D18
Not connected to A17/8/9 from PA3/PB0
Notes: * - DACK, DRQ, and IRQ lines: ISA line selected through jumpers; - - see circuit/text
Not used by the simulator
245 ENABLE (optional) - 3rd jumper on photo
MANUAL RESET

Figure 2: **This partial schematic *depicts the*** main sections of the ***PDI*** external **module.** Not shown are ***the*** fuses for **power-supply** connections and resistors between ***the*** data bus lines of ***the*** simulated ***slot*** and the corresponding lines of the ***PDI*** internal module.

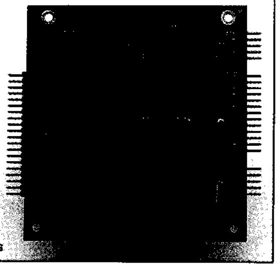
Listing 1: *Update ()* and Ge*tB i t ()* are two of the ***points*** of ***contact*** between **PDI *software* and hardware. Should you decide to use a commercially** available 8255 board, you'll probably have to modify these (and other) routines according to the board documentation. Various hardware access *#defines* are shown to help you decipher ***the*** code.

```
#define PDI_BASE  0x310      // IO base address
#define U01        0x00
#define U23        0x08
#define PORTAs     0x00
#define PORTBs     0x02
#define PORTCs     0x04
#define CTRL_BITs0x06
#define UO  0                // Ports[] vector access defines
#define  U1 4
#define  U2 8
#define U3 12
#define PA  0
i/define PB  1
#define PC  2
#define CB  3

unsigned char Ports[16]= { // 8255 registers are read-only  Ports[]
  0x00, 0x00, 0xFF, 0x99,   //holds the last written values
  0x00, 0x00, 0xFF, 0x90,
  0x00, 0x00, 0x00, 0x9b,
  0xE1, 0x80, 0x80, 0x80,
};


void Update(int flag)       // write Ports[] to the 8255s

  if (flag) {               // if flag is set then also a 'mode'
change
   outport(PDI_BASE+U01+CTRL_BITs, (Ports[U1+CB]<<8)+Ports[U0+CB]);
   outport(PDI_BASE+U23+CTRL_BITs, (Ports[U3+CB]<< 8)+Ports[U2+CB])
  }
```

(continued)

Sets the Pace in PC/104 CPU and DAS Technologies

33 MHz 486 SLC **$496**

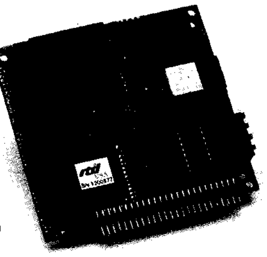


Features:
486SLC
33 MHz
387SX
2MB DRAM
SSD
EEPROM
WDT
IDE
FDC
2 Serial
EPP
PS/2 mouse
Keyboard
Quick boot
Virtual devices

CMi486SLC-1 Fully Integrated PC-AT with Virtual **Device** Support

When placing your order, mention this ad and receive a **387SX math coprocessor** FREE!

200 kHz 12-bit DAS **$498**



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

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

Make your selection from:

9 **cpuModules™**

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

7 **utilityModules™**

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

18 **dataModules®**

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

&Real Time Devices USA
200 Innovation Boulevard • P.O. Box 906
State College, PA 16804-0906 USA
Tel: 1 **(814)** 234-8087 • **Fax:** 1 **(814) 234.5218**
FaxBack®: 1 (614) 235-1260 • **BBS: 1 (814) 2349427**

RTD Europa	**RTD Scandinavia**
Budapest, Hungary	Helsinki, Finland
Fax: (36) 1 212-0260	Fax: (358) 0 346.4539

RTD is a founder of the PC/104 Consortium and the world's leading supplier of PC1104 CPU and DAS modules

#205

module via flat-cable segments, thus speeding board construction.

The LEDs let you quickly judge the high (LED on) or low (LED off) status of each ISA line. You can modify the schematic so each LED turns on only if the corresponding line is in its on state (usually a 0 value). This modification requiresthe use of inverters or connection of these LEDs between buffer output and V_{CC} instead of Gnd. However, it is my experience that the current way of functioning is consistent and requires little time to adapt.

Please note that the use of flat LEDs is recommended since spacing between the ISA slot pins is only 0.1". Adjacent LEDs must fit or you have to place the LEDs in two columns.

As a general rule, LED color indicates the bus line: data (red), address (yellow), and control (green). Exceptions to this rule include the RESET and DRQ lines and the "fuse-not-burned" LEDs for V_{CC}, ± 12 V, and -5 V. In total, you need 20 red, 22 green, 27 yellow LEDs.

Figure 2 offers detailed information about flat-cable connector pin use (matching the flat-cable connectors found on the PDI internal module) and ISA bus lines. Also included is the typical simulated ISA line (buffer and LED). A special case is provided by data bus lines, or optionally by each ISA line which can be either input or output, depending on bus condition (notably the MASTER line).

Listing 1: *continued*

```
outport(PDI_BASE+U01+PORTAs, (Ports[U1+PA] << 8) + Ports[U0+PA]);
outport(PDI_BASE+U23+PORTAs, (Ports[U3+PA] << 8) + Ports[U2+PA]);
outport(PDI_BASE+U01+PORTBs, (Ports[U1+PB] << 8) + Ports[U0+PB]);
outport(PDI_BASE+U23+PORTBs, (Ports[U3+PB] << 8) + Ports[U2+PB]);
outport(PDI_BASE+U01+PORTCs, (Ports[U1+PC] << 8) + Ports[U0+PC]);
outport(PDI_BASE+U23+PORTCs, (Ports[U3+PC] << 8) + Ports[U2+PC]);
```

These bidirectional lines might require special attention since the TBTB (or even the simulator software) might not function prop erly. It might happen that one or more of these bidirectional lines can be driven by both the simulator and the TBTB in two different states! Therefore, a resistor of approximately 330 placed between each PDI and ISA bus line helps reduce hardware damage by limiting current.

Not all ISA lines of the simulator slot have an LED since I decided to only support one interrupt channel and one DMA channel. However, these lines aren't fixed. Jumpersareprovided.Youcanchoosetoconnect to any possible INT or DMA line.

Unused input lines are pulled up. The LEDs corresponding to INT and DMA are placed in correspondence to a related line (INTx and DRQx/DACKx), which is near the configuration jumpers (see Photo 2).

Since TBTB insertion and extraction can require considerable force, don't forget to place supporting hardware. By doing this, you won't be as likely to bend the slot-holding board.

Though not shown in Figure 2, fuses should be placed [along with power-monitoring LEDs) on each power-supply line. Possible generous values could be 2 A (+5 V), 1 A (±12 V), and 0.5 A (-5 V). The figure for +5 V includes current needed by the PDI external module itself (buffers and LEDs).

THE SOFTWARE

The simulator software is written in Borland C. It compiles to a DOS program with an old-style interface similar to DOS. You type in commands which get interpreted as soon as you hit Return.

Before commenting on a couple of core routines, let me briefly discuss the overall operation. As mentioned, you can control each simulated bus line individually. For example, the command AEN=0 forces the AEN line (address enable) low. Format is somewhat flexible. A space can replace the = or you can use lower case.

You can also check the status of each line. For example, the command MASTER shows the current high or low status of the corresponding bus line. If the involved line is an output (and as such it is controlled by the PDI hardware), you see the last written value, not its actual voltage. The two values might differ with a hardware malfunction (i.e., TBTB or PDI failure). This difference

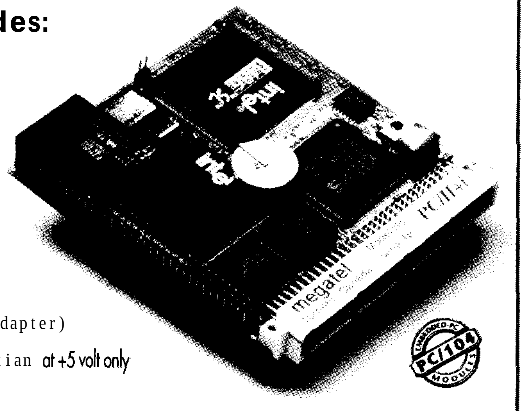
WORLD'S SMALLEST

486™ DX

Embedded PC with Floating Point, Ethernet & Super VGA Only 4"x 4"

The PC/II +i includes:

- 486DX™ CPU at 25MHz or 33MHz clock frequency
- Full 8K Cache with Floating Point
- Ethernet local Area Network
- Local Bus Super VGA Video/LCD
- Up to 2MBytes Flash™ with TFFS
- 4 or 16MBytes User DRAM
- PC/104 or ISA Bus compatible option (with adapter)
- 4"x 4" Format; 6 watts powerconwmptian at +5 volt only



486DX and Flash are registered trademarks of Intel Corp. as are PC, AT of IBM, megatel of Megatel Computer (1986) Corp

(416) 245-2953

125 Wendell Ave. • Weston, Ont. • M9N3K9 • Fax: (416) 245-6505 **megatel®**

#206

Listing 2: The Get *CmdLine()* routine gets the next command from either the keyboard (or a DOS pipe) or the currently executing ***macro file.*** Please note the use of the DOS ***0Ah*** (Buffered Keyboard Input) function call. This call lets the ***PDI*** program take advantage of any *DOSEDI* ***T-like utility you may have*** in your system.

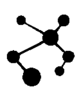
```
// Input line buffer: this data structure uses a DOS call which
// handles line input and is compatible with DOSEDIT, DOSKEY, etc.
typedef struct{
  unsigned char          size:
  unsigned char          got;
  char                   line[79];
} BuffKeybInp;

int   quit = 0;                   // set if exiting from simulator
int   InMacro = 0;                // set if executing a macro
int   ms_pause = 0:               // # of ms to pause
char CmdLine[128] = ":";          // the (default) command line
char *CmdPtr;                     // its pointer
char Token[20];
char MacFileName[60] = "pdi.mac", CurrMacroName[20],
     MacroParam1[40], MacroParam2[40];
FILE  *MacFile:

char *GetCmdLine(void)

  BuffKeybInp LineBuff;
  union    REGS     regs;
  char     Line[100], *p, tc;

  if (InMacro){
   if (feof(MacFile)) {
     EndMacroO;
     printf("\nUnexpected end of macro file '%s'!", MacFileName)
     *CmdLine = 0:
     return(CmdLine);

   if (fscanf(MacFile, "%[^\n]", CmdLine) != 1){
     EndMacroO;
     printf("\nRead error in macro file '%s'!", MacFileName);
     *CmdLine = 0:
     return(CmdLine);

   fgetc(MacFile);     // discard '\n' char
   if (!strcmp(CmdLine, "@")){
     EndMacroO;
     if (strcmp(CurrMacroName, ":")) // i.e. if not auto-startup
       printf("\nEnd of macro!          ---");
     *CmdLine = 0;
     return(CmdLine);

   else {          // not at end of macro
     if ((p = strchr(CmdLine, '%')) != NULL) {// if parameters
       tc = p[1];// '1' or '2' (parameter number)
       *Line = 0;
       strncat(Line, CmdLine, (pCmdLine))
       strcat(Line,(tc == '1') ? MacroParam1 : MacroParam2);
       strcat(Line, (p+2));
       strcpy(CmdLine, Line):

     printf("\n-%s", CmdLine);


  else {                 // not executing a macro!
   printf("\n-");

   LineBuff.size = sizeof(LineBuff.line) 1:
   LineBuff.got = 0;
   LineBuff.line[0] = 0x55;

   regs.h.ah = 0x0A;   // Buffered Keyboard Input
   regs.x.dx = (unsigned) &LineBuff;
```

(continued)

SIMPLE MULTITASKING EXECUTIVE

Your Complete x86 Solution!

Multitasking in Real and Protected Mode



smx© Full-featured, high-performance. preemptive kernel. Customized to x86 processors. Ideal for demanding applications. Real mode works stand-alone or with DOS. 16-bit, segmented protected mode works with *pmEasy16* or 286 I DOS Extender.™ 32-bit, flat protected mode works with *pmEasy32* or TNT.™

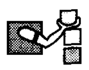
Protected Mode Environment

pmEasy™ 16- or 32-bit protected mode entry, DPMI services. application loaders, Periscope/32© and Soft-Scope© debugger support.

DOS-Compatible File System

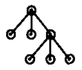
smxFile™ Full-featured file manager, IDE, floppy, and RAMdisk drivers available.

Dynamically Loadable Modules



smxDLM™ Runs independent executables as tasks which may be downloaded or loaded from disk.

Networking

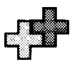


smxNet™ TCP/IP stack. Fast UDP. Packet driver interface. NE2000, SMC, and SLIP drivers available.

Task-Level Debugging

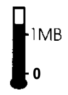
smxProbe™ Provides tracing and symbolic debugging. Works with or without code debuggers.

C++ Classes



smx++™ Class library built upon smx. Provides fully OOP-compatible kernel interface.

Extended Memory, Real Mode



smxEMS™ Allows copying data between real memory and extended memory buffers or accessing extended memory via a window.

User Interface

smxWindows™ Text windowing. Dresses up user interfaces.

Low Cost & Easy to Use

Royalty-free licenses. Supports common, low-cost C compilers and debuggers. Great manuals. Source code available.

Reliable

On market 6 years. 100's of applications. Extensive error checking 30-day free trial.

1-800-366-2491

http://www.earthlink.net/~mdi

MICRO DIGITAL, INC

fax 714-891-2363 Garden Grove, **CA USA**

bbs ***714-893-5118*** mdi@earthlink.net

#207

can probably be seen from the LED status.

Help is provided as a list of supported commands with syntax. There are other commands controlling features apart from bus lines.

Please note, however, that only basic features are directly supported by the software. While you can read and write memory or I/O (by hand or with the provided macros), more complex cycles such as DMA or MASTER cycles are not supported at a high level. In other words, you can perform these cycles, but you have to manually control the simulator (via single-line commands) or write your own macro.

The lowest-level routines are Up - date ( ), shown in Listing 1, and Get B i t ( ) Eventual software conversion to different I/O expansion boards (PDI internal module) will probably also require modifications to the routines Set Data ( ), SetAddr20(),DataIn(), and DataOut().

L. pd a t e ( ) transfers new output data from the Ports [1 array to the 8255 registers, while Get B i t ( ) reads the data lines declared as input. A flag in Up- date ( ) tells whether to also update the 8255s' mode registers (see the "Modifications" section).

Another important routine included in Listing 2 is GetCmdLine(), a name which fully describes its task. For line editing and command history retrieval, this routine can take advantage of system TSRs such as the popular DOSEDIT utility (but not DOSKEY, sorry!). The routine uses the Buffered Keyboard Input DOS function (OAh) to get user commands. It all works since many of the TSRs hookthemselvestothe DOS function, thus expanding its capabilities.

Listing **2:** continued

```
    intdos(&regs, &regs);

    LineBuff.line[LineBuff.got] = 0x00;
    strcpy(CmdLine, LineBuff.line);

  return CmdLine;
```

MACROS

In my opinion, thenicestfeatureofthe simulator is its macro support. Macros are simply a collection of consecutive commandswhich you group together for convenience. Listing 3 shows what a typical macro file looks like. Also shown is the "read from I/O port" macro, one of the most-used macros.

Macros let you easily extend the features of the simulator without C coding. C is only necessary if you decide to extend the simulator's command set. For instance, you mightwantto support macro nesting, looping, or (conditional) branching.

Each macro can accept up to two parameters of either text or numbers. With text, they cannot hold commas (since commas separate macro parameters), and with numbers, they must be expressed in hexadecimal notation.

The simulator can execute a macro command at full speed. Alternatively, each macro command can execute after a fixed delay (e.g., one command per second) or be single stepped (by pressing Return).

HERE WE GO

We're finally ready for a test run of the simulator. Included with the default macro file (PDI .MAC) are a couple of macros which can be used to printcharacterson a printer. The printer should be connected to the PC via a parallel interface board (LPT1) which is inserted into the simulated ISA slot.

The first macro (pi n i t) initializes the printer. Next, the p macro prints each single character to the printer. For example, since the characters "PDI" are ASCII 50, 44, and 49(hex), these values are given as parameters to three consecutive p macro calls: p ,50; p ,44; and p,49.

Don't forget to send carriage return (p,0d) and line feed (p ,0a) characters. Laser printers also like a form feed (hex Oc) to force them to eject the printed page.

MODIFICATIONS

Modifications and improvements are obviously possible. You may simplify the project by only simulating an 8-bit bus or decide to reduce or eliminate LEDs.

I can also suggest hardware changes which would help you avoid some inconvenience caused by a peculiar characteristic of the 8255 chip. Let me explain.

Imagine you programmed an 8255 (I'm talking about Mode 0) so port A is configured as an 8-bit output port. Port A could be changed from output to input and then, having performed some input task, back to output. Unfortunately, as soon as

Listing 3: A macro *may be used to* ***perform*** *i n p u t o p e r al/O opnosr tfsr.o m̄ h e* *" I n p u t I/On port" macro is commented assembly sty/e.* ***Please note*** this commenting ***is not allowed*** in a real macro file. I ***do it here to clarify macro command interpretation for you.***

```
default macro file ('pdi.mac') for pdi.exe
 first line of a macro file must be '::'
 each macro starts with a line = ':MACRONAME' (case is ignored!)
 each macro ends with a line = '@'
 free comments are allowed between macros
 macro ':' is special as it is automatically executed at startup
 macros may have 2 parameters: '%1' & '%2' in the macro text.

   To invoke a macro with a parameter: ':macroname,parameter'
   Please note that parameters are separated with commas!

Special startup macro (may be modified as you need)

@

Input  from I/O port
:i     ; name of macro, invoke with ":i,nn"(nn is address in hex)
di     ; declare data bus as input
a %1   ; set address bus to value of macro parameter
aen 0  ; activate AEN line
ior 0  ; activate IOR line
d      ; read data bus
ior 1  ; IOR inactive
aen 1  ; AEN inactive
@      ; end of macro
```

you write a new value into the 8255 mode register, even if it is exactly the same (e.g., redeclaring an output port as output), a data port declared as output loses its output values! Obviously, you can quickly reprogram the output register, but you still get an output glitch on the lines.

In our case, this relatively small problem happens when we switch direction of the data bus, thus requiring an 8255 mode change. Since the PDI internal module is a 16-bit board, two 8255s are involved, so a potential maximum of 48 output lines undergo this glitch. Of course, the situation is typically not so tragic, and I haven't seen any serious problem while testing the PDI.

Nevertheless, if you want to be sure that this cannot cause any problems, the only solution is to use two lines-one for output and one for input-for every bidirectional bus line.

You also need a tristate buffer between the output line and simulated bus line. That way, you never have to reprogram any 8255 mode register, but you do have to control the tristate buffer (with a common line), thus further increasing the number of required I/O lines for the simulator!

From a software point of view, porting to Windows would be the biggest modification. Windows would offer fancy things such as graphical simulator status indication (you could use colored buttons which mirror LED status changes), line control via mouse clicks, and so on.

I leave the special flare to you as you build your own PDI. EPC

Stefano Chiti-Batelli ***has been*** involved in ***personal computer and microcontroller programming for 15 years.*** His experience ***ranges from pinball machines and video games to security control systems. Stefano is currently completing a degree in mathematics.***

REFERENCES

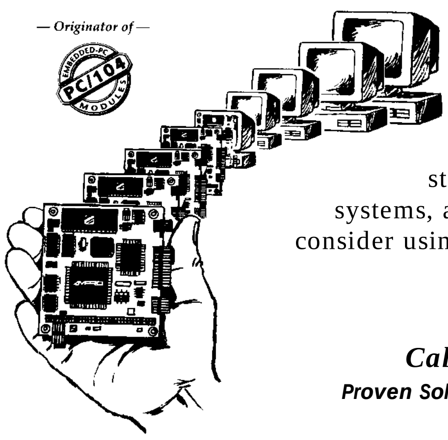
Nisley, Ed. "Blindsided by Technology: The '386SX Embedded Firmware Project." INK, 31, February 1993, 48–55.

Solari, Edward. AT Bus Design. San Diego: Annabooks.

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

Don't Limit
Your Embedded System Designs!



If your embedded system applications require PC-like functions, i.e. GUI, networking, mass storage, standard operating systems, and off-the-shelf software, consider using the PC architecture as a macrocomponent.
Let us show you how.

Call 1-800-966-5200 today.
Proven Solutions for Embedded Control

the PC architecture that fits in the palm of your hand

AMPRO®
COMPUTERS, INCORPORATED
990 Almanor Avenue
Sunnyvale, CA 94086

#208

Byte Craft

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

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

We respond to your
C compiler needs.

MPC for PIC16/17Cxx
C38 for MELPS 740
COP8C for COP8
C6808 for MC68HC08
Z8C for Z8
C6805 for MC6805

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

#209

David Prutchi

Hot Swapping on the PC

An Active Extender Card for the 16-bit PC-ISA Bus

Nothing is worse than powering the motherboard up and down to test a development board Design flaws at best can prevent operation and at worst cause permanent damage. Join David in looking at a safer, faster solution.

The thing that annoys me the most when testing and debugging an add-in card for the PC is the constant need for powering the computer up and down to reconfigure jumpers, make circuit changes, and reposition probes.

Standards are being developed under the Plug-'n'-Play philosophy [1] which let users remove and replace add-in cards and peripherals while a PC is fully powered without damaging the added hardware or the computer. Software standards are also being introduced so that *hot swapping,* as this procedure is commonly known, does not crash the system.

Until a safe hot-swapping standard is widely accepted and implemented, testing, debugging, and configuring add-in PC cards involves incessant waiting-again and again-for bootup and program initialization unless, of course, you experiment with the hot-swapping extender card (see Photo 1) described in this article.

Through a relatively simple hardware arrangement, it is possible to safely hot swap an add-in card on any PC. All you need is a means of isolating the PC's data, power, address, and control lines from those of the add-in card under test.

The hot swapping extender card uses high-speed solid-state switches to connect the digital signal lines of the 16-bit PC-ISA bus to the add-in card under test. Power lines are also switched in the card, and a specialized IC safeguards the computer's

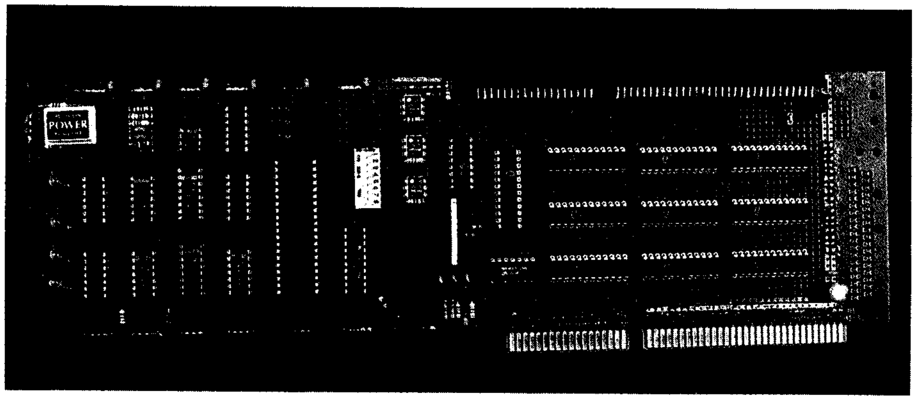


Photo 1: The hot-swap extender card fits in a standard 16-bit ISA slot, while the add-in card under test sits on top of the extender card. Hot-swapping with the exte
without
works well for
PCB is available for systems where timing *l* .

supply from power transients during a hot swap.

To hot swap this extender **card,** simply press a push-button switch to power the add-in card down and up again without causing the computer to crash. Timing logic in the extender card connects or disconnects the logic and power lines in a sequence that ensures the add-in card can restart its activity after being replaced on a fully operational computer.

Alternatively, the operation of the extender card may be controlled through software. Toward the end of this article, I'll show you how to use this feature to implement a highly automatic test setup.

SWITCHING THE 16-BIT ISA BUS

The 16-bit Industry Standard Architecture (ISA) bus comprises the original 62-contact, 8-bit bus of the PC/XT and an extra 36-pin connector to carry the additional signals available for system expansion when the PC/AT was introduced. A total of 16 data lines, 27 address lines, 1 1 interrupts, 7 DMA channels, and an assortment of control lines and power buses make up the 98 pins of the full ISA bus.

During a typical transaction between the PC and an add-in card, the PC issues an address which, depending on the type of data exchange, is validated through either the address latch enable (ALE) or address enable (AEN) lines. The issuing of a valid address is accompanied by either a read or write signal through which the processor or DMA controller communicates to the add-in card the direction of data transfer. I/O read and write signals (/IOR and /IOW) indicate that data is to be transferred to or from the data bus. Memory read or write signals (/SNENR and /SMEMW) indicate CPU or DMA controller need a data transfer to or from main memory.

The add-in card issues handshake signals to the motherboard to indicate parity error states on an I/O channel (/IOCHCK), request the insertion of additional wait states (IOCHRDY), and request completion of the current bus cycle without additional wait states (/OWS).

For DMA transactions, the add-in card requests access to DMA devices through lines DREQO-DREQ3 and DREQ5-DREQ7, while the PC acknowledges such requests through lines /DACK0-/DACK3 and /DACK5-/DACK7. The PC issues a high pulse on the TC line when the terminal count for a DMA channel is reached. In addition, the /MASTER signal can be used in combination with a DRQ line by add-in

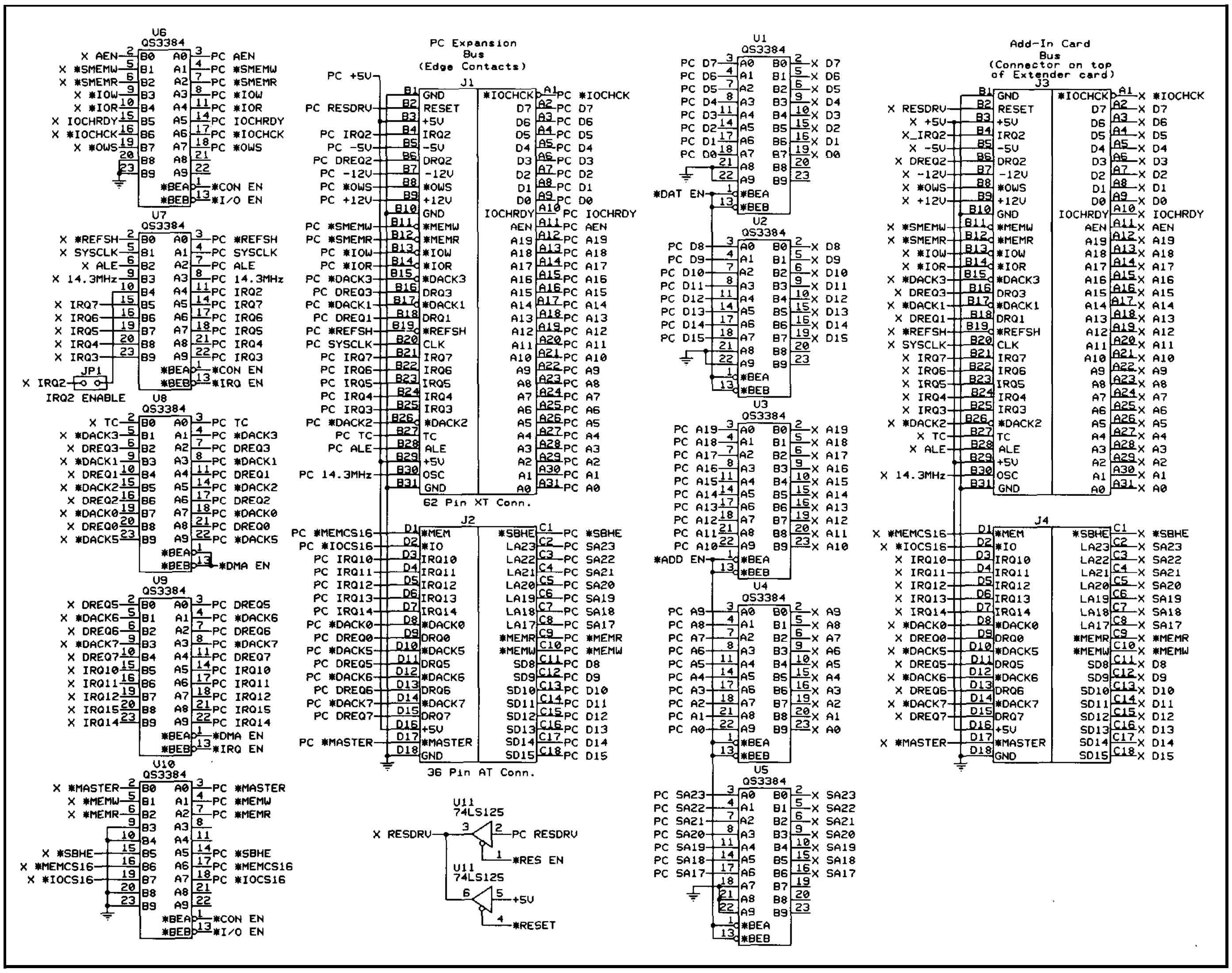
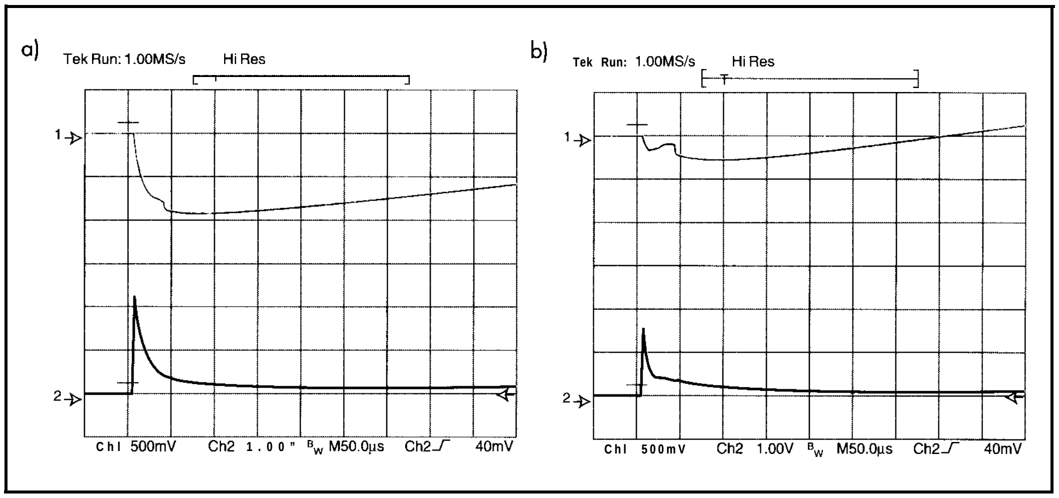


Figure I: The data, address, and control lines of the ***16-bit*** PC ISA bus (***J*** 1, ***J2***) are switched on and off the add-in card's bus (***J3, J4***) through ***QS3384*** high-speed CMOS analog switches.

Figure 2: Suddenly connecting an add-in card to the PC's +5-V power line can glitch the system power. Directly connecting an add-in card to the power line (a) results in the voltage glitch (Channel 1) and inrush current (Channel 2) shown by a scope. With the help of a UCC3912, the same glitch (b) is shown again. Channel 1 is offset by +5 V. Channel 2's vertical scale is 10 A per division.



boards to take control of the bus through a DMA channel.

To provide compatibility with XT add-in cards, 16-bit data exchange between the motherboard and an add-in card can take place either as two 8-bit or a single 16-bit transfer. The system high-bus enable /SBHE line is pulled low when the upper eight data bits are used. A 16-bit access to memory locations or I/O locations is requested by the add-in card through the /MEMCS 16 and /IOCS 16 lines. /MEMR and /MEMW on the 36-pin connector indicate to the processor that the add-in card needs memory access to the full 16 MB, rather than to the l-MB access available to /SMEMR and /SMEMW.

The other signals available in the bus are used for system-wide control. RESDRV is a reset driver signal used to reset system logic at powerup or as a response to a power bus fault. The SYSCLK line carries the system clock, while the 14.3-MHz line carries the system-oscillator signal to the

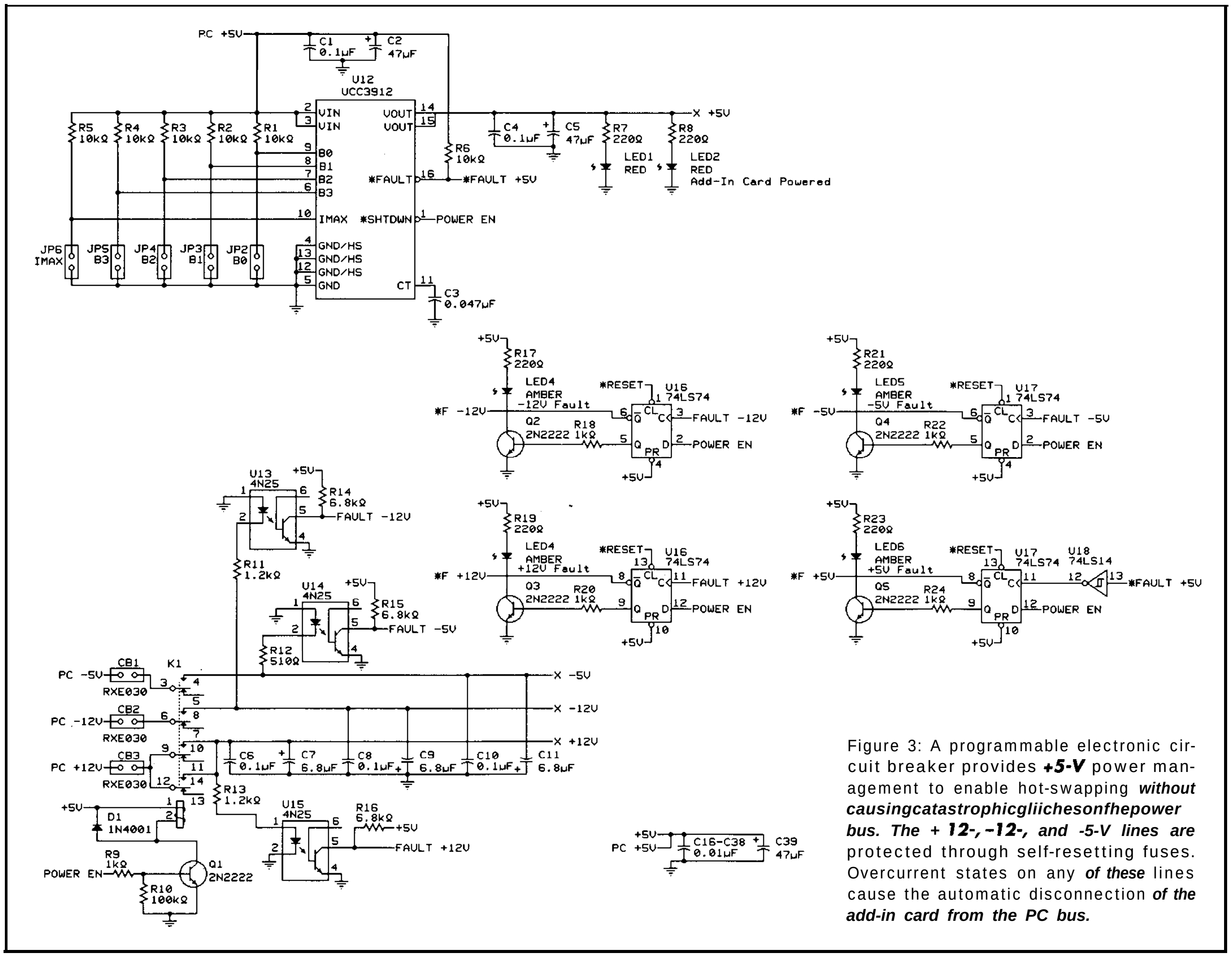


Figure 3: A programmable electronic circuit breaker provides ***+5-V*** power management to enable hot-swapping ***without causingcatastrophicgliichesonfhepower bus. The + 12-,-12-, and -5-V lines are*** protected through self-resetting fuses. Overcurrent states on any ***of these*** lines cause the automatic disconnection ***of the add-in card from the PC bus.***

Trip current [A] (typical)	B0 (JP2)	B1 (JP3)	B2 (JP4)	B3 (JP5)	Max. output current [A] I_{MAX}=0 (JP6)	I_{MAX}=1 (JP6)
	0	0	0	0		
0	1	0	0	0	0.02	0.02
	0	1	0	0		
	1	1	0	0		
0.25	0	0	1	0	1.25	
0.5	1	0	1	0	1.5	
0.75	0	1	1	0	1.75	
1.0	1	1	1	0	2.0	
1.25	0	0	0	1	2.25	
1.50	1	0	0	1	2.50	4.0
1.75	0	1	0	1	2.75	
2.0	1	1	0	1	3.0	
2.25	0	0	1	1	3.25	
2.50	1	0	1	1	3.50	
2.75	0	1	1	1	3.75	
3.0	1	1	1	1	4.0	

***Table 1:** Jumpers JP2–JP6* select the *trip current and maximum transient current of the +5-V power line. A jumper* in *place* corresponds *to* a logic zero on *the controlled line.*

expansion bus. Finally, the /REFSH line becomes active whenever a refresh cycle is in effect.

Signal direction across the 16-bit ISA bus is not always easy to establish. Moreover, line loading and driving are dependent on the address locations decoded by the add-in card, as well as by any pending interrupt and DMA requests. Considerable amounts of specialized logic would be required on an active extender card to interconnect the buses through tristate logic transceivers. Instead, bidirectional analog switches or electromechanical relays provide a simple link between the buses of the PC and the add-in card.

In the hot-swap extender card (see Figure 1), the data, address, and control lines of the PC's ISA bus are tied to those of the add-in card by Quality Semiconductor's QS3384 bus switches U1–U10. Each QS3384 is made up of ten high-speed CMOS analog switches that are controlled in banks of five switches each through TTL-compatible signals /BEA and /BEB.

With its corresponding control signal disabled, a switch within the QS3384 provides an isolation of more than 100 M When enabled, however, the resistance across the switch drops to approximately 5 effectively connecting the associated line on both buses without adding propagation delay or generating additional ground-bounce noise.

Bus lines have been partitioned into various functional groups. Partitioning is required so logic signals connect or disconnect in a sequence that ensures the add-in card can restart its activity after being replaced on a fully operational computer. The 16-bit data bus is handled by U1 and U2, which are enabled through the extender card's /DATA_EN signal.

The /ADD_EN signal switches the address lines by U3–U5. /IRQ_EN switches interrupt request lines (with the exception of IRQ2) through the lower banks of U7 and U9. The lower banks of U6 and U10 handle the connection of I/O control signals under the command of /I/O_EN. DMA requests and acknowledgments, including the associated DMA Terminal Count signal (TC), are switched by U8 and the upper bank of U9 whenever /DMA_EN becomes active. With the exception of the system reset line RESDRV, all other control signals are switched under the command of /CON_EN.

RESDRV is unidirectional from the PC to the add-in card. Signaling occurs by means of a 74LS125 tristate buffer under the control of /RES_EN. In addition, an "artificial" reset signal is issued through U11b to the add-in card under test whenever /RESET is strobed by the extender card's control logic.

HOT SWAPPING

Not causing bus conflicts while hot swapping goes beyond simply switching the signal and power lines at the right time. Capacitive and inductive inrush currents established when an unpowered card is directly connected in an operational system cause glitches on the power buses.

YOU DON'T GET TO THE TOP BY THINKING SMALL, SO WHY MESS AROUND WITH INFERIOR TOOLS. ADOPT TEAM PARADIGM AND TAP INTO THE FINEST TOOLS, BACKED BY UNLIMITED FREE TECHNICAL SUPPORT.
TO BE CONTINUED...

#210

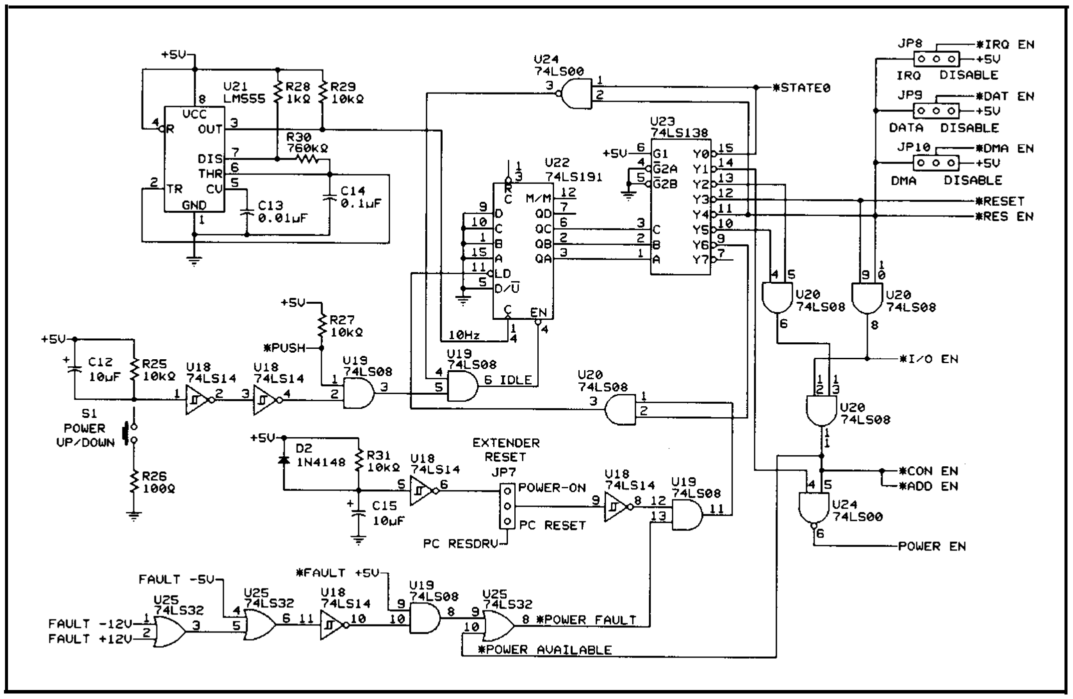


Figure 4: A simple state machine connects or disconnects the logic and power lines in a sequence that ensures that the add-in card can restart its ***activity* after** being replaced on a fully operational computer. A push-button switch, overcurrent fault signals, and PC-controlled signals are used as ***inputs for the*** activation ***of the control*** logic.

These glitches can lead to system failures such as data corruption and logic lockups.

Figure 2a presents the glitch caused on a PC's +5-V power line by the sudden and direct addition of a relatively power-hungry add-in card. This add-in card acts as a 1.6-A, 183-μF load, causing a transient inrush current that peaks at approximately 22 A! The corresponding voltage glitch on the PC's +5-V power bus peaks at0.8 V, resulting in a transientdrop of the power bus to +4.2 V. The minimum +4.5 V required for normal operation restores only 0.5 ms later. This obviously violates the operational limits established for TTL logic, and always caused the PC to freeze, report BIOS errors, or reboot itself.

A new IC by Unitrode, the UCC3912 integrated electronic circuit breaker, automatically limits the peak load current on hotswap to a preprogrammed value. In addition, this IC implements a fully automatic circuit breaker that can be programmed to trip within a range of 0.25–3.0 A with transient output of up to 4 A. As shown in Figure 2b, using the UCC39 12 to power the same add-in card as before, the glitch amplitude is only 0.27 V, effectively eliminating the possibility of causing annoying logic faults.

In the extender card, the desired +5-V fault current is programmed through jumpers JP2–JP6 (see Table 1). Power on and off are controlled through the /SHTDWN TTL-compatible input of the UCC3912. Overcurrent or thermal shutdown conditions issue the /FAULT signal and are used by the extender's control logic to automatically isolate the PC bus from the add-in card.

PC add-in cards using ±12- and -5-V power buses seldom demand more than 0.25 A from any single supply line. The limited use of these power buses minimizes the risk for glitch-induced faults due to hot swapping.

In the extender card, Raychem's PolySwitch self-resetting fuses CB1–CB3 protect the PC and the add-in card under test from shortcircuitsand otherovercurrent fault conditions on these power buses. A small, four-pole relay connects the PC's ±12- and -5-V power lines to those of the add-in card.

PolySwitch devices are positive-temperature-coefficient resistors that rapidly increase resistance in response to excessivecurrent flow [2, 3]. Optoisolators U13–U 15 monitor changes in resistance of the PolySwitch devices to issue TTL-compatible indications for overcurrent fault conditions. Just as with

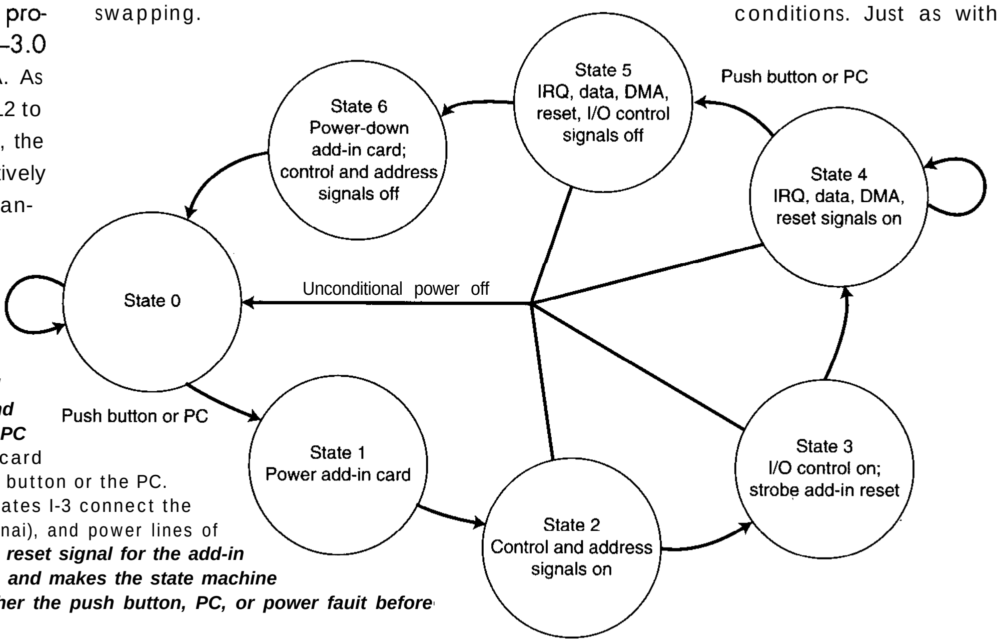
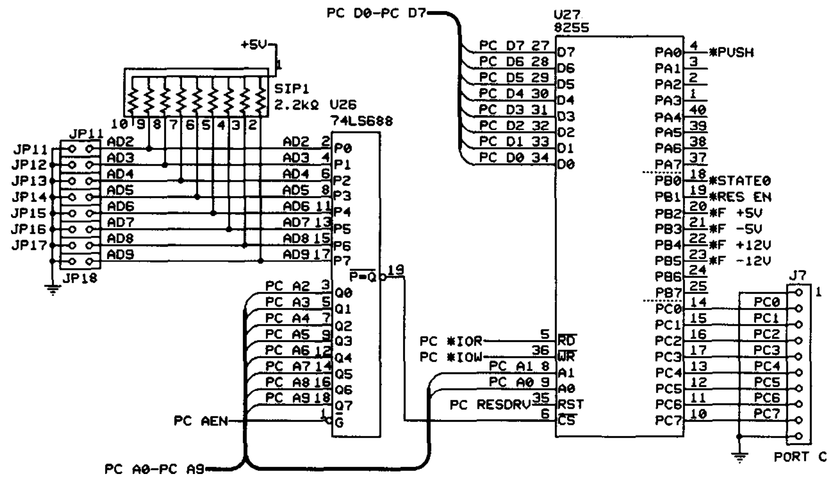

State 0
State 1
Power add-in card
State 2
Control and address signals on
State 3
I/O control on; strobe add-in reset
State 4
IRQ, data, DMA, reset signals on
State 5
IRQ, data, DMA, reset, I/O control signals off
State 6
Power-down add-in card; control and address signals off
Push button or PC
Push button or PC
Unconditional power off

Figure 5: ***The*** sequential states of the state ***machine control the power and signal flow between the add-in board and the PC. In state 0, the extender keeps the PC*** bus isolated from that of the add-in card while waiting for an input from the push button or the PC. Once an activation signal is received, states l-3 connect the address, ***control*** (except for the reset signai), and power lines of the bus. State 3 produces an ***artificial reset signal for the add-in card. Finally,*** *state 4 fuiiy links the buses and makes the state machine* ***wait for a*** deactivation ***signal from either the push button, PC, or power fauit before proceeding to*** bus disconnection.



Part	+5V	GND
U1	24	12
U2	24	12
U3	24	12
U4	24	12
U5	24	12
U6	24	12
U7	24	12
U8	24	12
U9	24	12
U10	24	12
U11	14	7
U12	2,3	4,5,12,13
U13	-	-
U14	-	-
U15	-	-
U16	14	7
U17	14	7
U18	14	7
U19	14	7
U20	14	7
U21	8	1
U22	16	8
U23	16	8
U24	14	7
U25	14	7
U26	20	10
U27	26	7

Figure 6: An 8255 parallel peripheral ***interface lets*** *sofhvare control the operation of the extender card. An input port enables* ***software*** *to read the state and fault conditions of the extender card, simplifying the design of automatic* ***test platforms.***

overcurrent faults in the +5-V bus, these cause the extender card's control logic to automatically isolate the PC bus from the add-in card. latches U 16 and U 17 record the tripping of overcurrent fault detectors and visually indicate an error condition through LED3-LED6.

CONTROLLING THE EXTENDER

The circuit of Figure 4 implements a simple state machine to connect or disconnect the various logic and power lines of the ISA bus in a sequence that ensures the add-in card can restart its activity after being replaced on a fully operational computer. Synchronous counter U22 counts pulses generated by U2 1, a 555 configured as a 1 O-Hz clock. Depending on JP7, the counter is unconditionally reset to zero either at powerup by U 18c or by the PC's system reset line PC_RESDRV.

U23 decodes the S-bit count of U22 to seven possible low-level true states. As shown in Figure 5, when U22 is reset, state 0 causes the output of U19b to go high, preventing U22 from counting. When push-button switch S1 is momentarily pressed, however, U22 is enabled, allowing the count to proceed.

As the count increases, U20b-d and U24b first enable connection of the power lines on state 1. Then, on state 2, control and address lines are connected, and in state 3, the I/O control lines of the add-in card are activated.

Additionally, state 3 strobes the X_RESDRV line through U 1 1 b, resetting the add-in card. The strobe signal also resets any LED fault indicator tripped during the previous cycle. When state 4 is reached, the PC reset, data, interrupt request, and DMA lines are connected to the add-in card. State 4 also freezes the count of U22, allowing the card to operate normally.

Pressing the push button again enables the state machine to enter state 5, where the lines activated by states 3 and 4 are deactivated. Finally, progressing to state 6, U22 is reset to zero, and the add-in card is isolated from the PC expansion bus.

FOR EMBEDDED C/C++ APPLICATION5 USING THE AWESOME BORLAND & MICROSOFT COMPILERS, I BET MY LIFE ON PARADIGM LOCATE & PARADIGM DEBUG. (SAN YOU TRUST ANYTHING ELSE?
TO BE CONTINUED...

#211

The rest of the logic protects the PC from short circuits in the add-in card under test. Just after power is available to the add-in card, an overcurrent faulton any of the power lines causes the unconditional and asynchronous reset of the counter.

TEST AUTOMATION

Powering the computer up and down to reconfigure jumpers, make circuit changes, and reposition probes may be a tolerable annoyance when testing one add-in card. But, as soon as you decide to try your entrepreneurial skills by commercially producing a batch of your own design, testing and reworking with the least possible waste of time becomes a must. Moreover, you may want to automate the test protocol so that a pass/fail evaluation of your add-in cards can be carried out at the assembly house in a time-efficient manner.

As shown in Figure 6, an 8255 PPI enables software to control the operation of the extender card. An input port lets software read the state and fault conditions of the extender, making it possible for you to design an automatic test platform for your product.

Listing **1:** This sample program demonstrates confroiiing and inquiring the hot-swap extender card from the **PC.** The program runs under ***QuickBASIC.*** Notice ***that*** the default base address for the ***8255*** is ***300h.***

```
REM Control program for the hot-swap extender card
bas = &H300: REM base address for the 8255
REM base -> port A
REM base+1 -> port B
REM base+2 -> port C
REM base+3 -> control register
OUT bas + 3, 128 + 8 + 2 + 1: REM PA output, PB and PC input
OUT bas. 1: REM set line 1 (notPUSH) of port A to high
main:
CLS
PRINT "[1] Power UP add-in card"
PRINT "[2] Power DOWN add-in card"
PRINT "[3] Read current status of extender card"
PRINT: PRINT "[4] EXIT": PRINT
INPUT menu
IF menu = 1 THEN GOSUB powup
IF menu = 2 THEN GOSUB powdown
IF menu = 3 THEN GOSUB status
IF menu = 4 THEN END
GOTO main
powup:
REM Power-up add-in card
REM First verify that card is not already powered up
status = INP(bas + 1): REM read status
IF (status AND 2) = 0 THEN
  PRINT "Add-in card already powered up": BEEP
  FOR delay = 1 TO 4000: NEXT delay
  RETURN
END IF
```

(continued)

Strobing line 0 of port A has the same effect as actuating the push button on the extender card. The lines connected to port B convey information regarding the activation status of the extender card, as well as of any overcurrent fault warnings that had been issued. Port C is available through connectorJ5, and you may use it as a logic interface between your automated test program and probes connected to the add-in card under test.

listing 1 provides an example of how to write your own extender card control routine for an automated test program. The program first configures the 8255 and sets high line 0 of output port A. Activation or deactivation requests are handled by comparing the current state of the extender card to the requested state, and issuing a strobe on port A if necessary.

INSTALLING AND OPERATING THE EXTENDER

Set the base address and configuration jumpers before inserting the extender card into a free expansion bus slot in the PC. An appropriate address for the ex-

Q. ***How do you know you're getting the most from your development tool purchase?***

A. ***Compare Avocet Systems with the competition.***

- A Broad Line of High-Quality Products at Competitive Prices
- Free On-Line Technical Support for Registered Users. No Voicemail!
- Attractive Multi-User Discount Prices & Our "50%+" Educational Discount Plan
- Unconditional 30-Day Money-Back-Guarantee

Now **call the obvious choice!**

AVOCET
SYSTEMS; INC.

The Best Source for Quality Embedded System Tools

(800) 448-8500
(207) 236-9055 Fax (207) 236-6713

ANSI-C COMPILERS
8051 • 68xxx • 68HC11
6805 • Z80/180/64180
6801 • 6809
H8/300 & More

MACRO ASSEMBLERS
8051 • 68xxx • 68HC11 6805
• Z80/180 • Z8 • 8096/196
6800/6801 • 6809
H8/300 • 8048 & More

SIMULATORS/DEBUGGERS
8051 • 680xx • 68HC11 6805
• Z80/180 • 6800
6809 • 8048 • 6502
8085 & More

AND HARDWARE
EPROM Programmers &
Emulators by EETools, Tribal,
Softaid, Cactus Logic

#212

Listing 1: continued

```
GOSUB pushb: REM Strobe pushbutton signal
RETURN
powdown:
REM Power-down add-in card
REM First verify that card is not already powered-down
status = INP(bas + 1): REM read status
IF (status AND 1) = 0 THEN
  PRINT "Add-in card already powered down": BEEP
  FOR delay = 1 TO 4000: NEXT delay
  RETURN
END IF
GOSUB pushb: REM Strobe pushbutton signal
RETURN
status:
REM Read and display extender card status
CLS: PRINT "STATUS"
status = INP(bas + 1): REM read status
IF (status AND 1) = 0 THEN PRINT "Add-in card powered down"
IF (status AND 2) = 0 THEN PRINT "Add-in card powered up"
IF (status AND 4) = 0 THEN PRINT "Fault on +5V line detected"
IF (status AND 8) = 0 THEN PRINT "Fault on -5V line detected"
IF (status AND 16) = 0 THEN PRINT "Fault on +12V line detected"
IF (status AND 32) = 0 THEN PRINT "Fault on -12V line detected"
FOR delay = 1 TO 4000: NEXT delay
RETURN
pushb:
REM Strobe low push-button signal to toggle state of extender card
OUT bas, 0: REM Strobe push-button signal low
FOR delay = 1 TO 500: NEXT delay
OUT bas, 1: REM send signal back high
RETURN
```

tender card PC interface can be selected between 300h and 3 1 Fh, which is defined by IBM as the location for prototype circuits. The example program assumes that the extender card has a base address of 300h, which is set with jumpers JP11–JP15, while leaving JP16–JP17 open.

Place jumpers on JP1 and JP8–JP10 to enable connection and disconnection of all bus lines to the add-in card. In addition, configure jumpers JP2–JP6 to program the desired fault and inrush currents for the +5-V bus. JP7 selects the source for the unconditional reset of the extender.

Connecting U18d to the output of U18c resets the extender at powerup, while letting the add-in card remain connected to the expansion bus during warm boots. In the other position of JP7, the extender card isolates the add-in card from the expansion bus every time the PC issues a system reset.

Install the extender card, but don't insert an add-in card on connectors J3–J4 yet. On powerup, verify that LEDs 1 and 2 are off, indicating that the extender card is inactive. Actuating the push button activates the extender and lights up LEDs 1 and 2. Pushing the switch once again deactivates the extender.

You can verify that the PC controls the extender card by running the demonstration program. If conflicts with other cards occur, change the base address of the extender, modifying the program accordingly. If all works well, you may plug an add-in card on J3–J4 while LEDs 1 and 2 are off, and then activate the extender.

Once LEDs 1 and 2 turn on, simply run the application software that uses the add-in card, and proceed to probe the card. Remember, however, that some add-in cards (e.g., disk controllers, video adaptors, etc.) require initialization by the BIOS after being reset.

To hot swap these cards, run a dedicated program duplicating the BIOS initialization of the add-in card under test. Only then can you run the application software that uses the add-in card. Alternatively, configure JP7 to connect U18d to the output of U18c, and warm boot the PC each time after activating the extender card.

Finally, always make sure you deactivate the extender before removing or reconfiguring an add-in card under test.

TO BE CONTINUED.

Be a hero – put Team Paradigm to work on your next embedded x86 design and reap the rewards you deserve. Team Paradigm has the ability to deliver all the embedded system pieces from Intel to AMD, C/C++ or assembly language, and all the Borland/Microsoft development tools. Plus Paradigm DEBUG for your favorite in-circuit emulator and real-time operating system.

So forget about making excuses and instead enlist Team Paradigm for your current or next x86 project. We deliver the finest embedded x86 development tools, backed by unlimited free technical support. No one is more serious about your success than Paradigm Systems. Call today and get the details before you waste any more of your precious time.

PARADIGM 'Nuff said.

Proven Solutions for Embedded C/C++ Developers

1-800-537-5043

Paradigm Systems
3301 Country Club Road, Suite 2214, Endwell, NY 13760
(607) 748-5966 / FAX: (607) 748-5968
Internet: 73047.3031@compuserve.com

©1995 Paradigm Systems, Inc. All rights reserved.

#213

IN CONCLUSION

Today's PCs often come with a blend of buses besides the basic 16-bit ISA bus. The same principles described for hot swapping on the ISA bus can be applied to hot swapping on other buses. With additional information on the signals, power, and timing requirements of these standard buses [4], you can design a modified extender card that lets you successfully hot swap any add-in card.

In fact, you can even design an active extender that enables you to use hardware you could not normally interface to your computer. By taking advantage of the current-limiting characteristics of the QS3384 [5], it is possible to interface standard 5-V TTL logic to the new 3.3-V TTL logic used in portable and "green" computers.

Despite the increasing popularity of some of these new buses, ISA is alive and well. Most add-in cards currently in the market are compatible with either the 8-bit or 16-bit ISA bus, and I dare to predict this will not change soon.

Yes, I do believe that eventually PCI or some other enhanced bus will become the interface of choice for desktop systems. **However, the considerable effort required to design a fully compliant PCI interface in** combination with the lack of Plug-'n'-Play hardware and software standards offer little hope of promptlygeneralizing direct hot swapping on the expansion bus.

Until then, I hope this extender card eliminates the need for waiting-again and again-for boot-up completion and program initialization everytimeyou make the slightest change to an add-in card under test. EPC

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* maybe *reachedatdavidp@mails.* imed. corn.

REFERENCES

[1] D. Strassberg, "Plug and Play," *EDN*, **40**(5), March 1995, 33-43.

[2] T. Fang and S. Morris, "Conductive Polymers Prolong Circuit life," Design News, **84**(21), Nov. 1992, 99-I 02.

[3] J. Bachiochi, "Sacrifice for the Good of the Circuit: Strengthening the Weak link," INK, 60, July 1995, 82-85.

[4] S. Bigelow, "Understanding PC Buses," INK, 50, Sept. 1994, 44-5 1.

[5] Quality Semiconductor, "Bus Switches Provide 5-Volt to 3-Volt logic Conversion with Zero Delay," Application Note AN-I 1, June 1993.

SOURCES

Unitrode Integrated Circuits
7 Continental Blvd.
Merrimack, NH 03054
(603)424-2410

Quality Semiconductor
85 1 Martin Ave.
Santa Clara, CA 95050
(408)450-8080

Raychem Corporation
300 Constitution Dr.
Menlo Park, CA 94025
(800)272-9243

Active Extender card kit for 16-bit PC-ISA bus [includes PC board and all components)...$225
Active Extender card (assembled and tested version). $295
A four-layer PCB [includes PC board and all components). $65
Plus applicable taxes, postage, and handling.

Adnav Electronics
58 Chicory Ct.
Lake Jackson, TX 77566
(409)292-0988

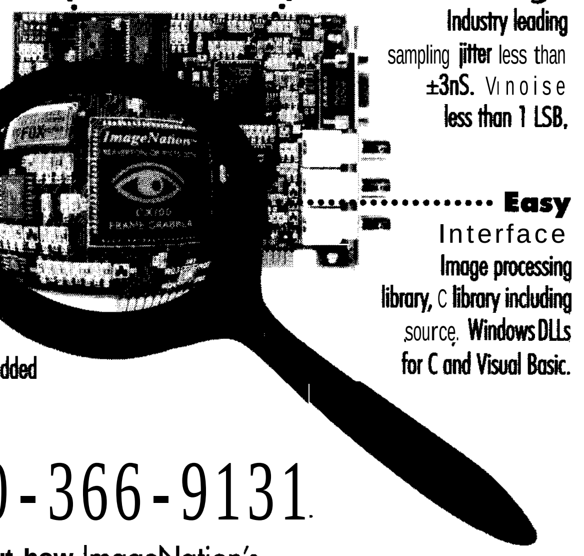
IRS

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

Anatomyofa
Great Frame Grabber

Low Price
Forget high priced on-board processing. Our designs exploit the host CPU without sacrificing performance.

Precision
Image
Industry leading sampling **jitter** less than **±3nS.** Vi noise **less than 1 LSB.**

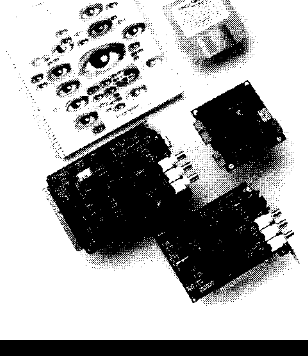


Easy
Interface
Image processing library, C **library including** source. **Windows DLLs for C and Visual Basic.**

Small
Board
Half dot **ISA, PC/104 and STD boards for embedded and compact designs.**

Call 800-366-9131.

Call and find out how ImageNation's **quality products, responsive technical support and elegant software get your application quickly to market.**



ImageNation™
Vision Requires Imagination"

I-800-366-9131 • Phone: 503-641-7408 • Fax: 503-643-2458
E-mail: info@ImageNation.com
Homepage: http://www.ImageNation.com/-image
P.O. Box 276, Beoverton, OR 970750276

#215

PC/104QUARTER

Rick Lehrbaum

Constructing PC/104-based Systems

From plug-in memory cards for portables, PCMCIA *has grown to include hard disk drives, fax/modems, and other peripherals. Its use in the embedded world is gaining popularity, especially with new* PC/7 04 *offerings.*

Decided to use an embedded-PC architecture in that new development project? Perhaps, it's a portable data logger, a vehicular performance monitoring system, or a telecommunications protocol converter.

Maybe you've picked up your own copy of the *PC/104* Resource Guide and found CPU modules, display interfaces, data acquisition and control modules, and other PC/I 04-related accessories and peripherals you can use to design and build your system. No doubt, your supply of PC/I 04 product information is steadily growing.

So, when you selected your components, did you choose PC/I 04 modules for the embedded PC in your project because they fit within a space where normal desktop motherboards and expansion cards could't possibly go? Or, did you opt for PC/104 modules because their ruggedness, reliability, operating temperature, and quality exceed ordinary PC/AT motherboards, expansion cards, and peripherals?

Whatever the reason, you'll likely discover that **by using** PC/I 04 modules, you're able to focus more on the specifics of your application, especially the unique I/O, packaging, and application software. You'll be through with wasting a lot of time and money "reinventing the wheel."

MACROCOMPONENT BUILDING BLOCKS

You may have seen a picture like the one in Photo 1, comparing PC/I 04 modules to Lego building blocks. Although this image points out how easy it is to build

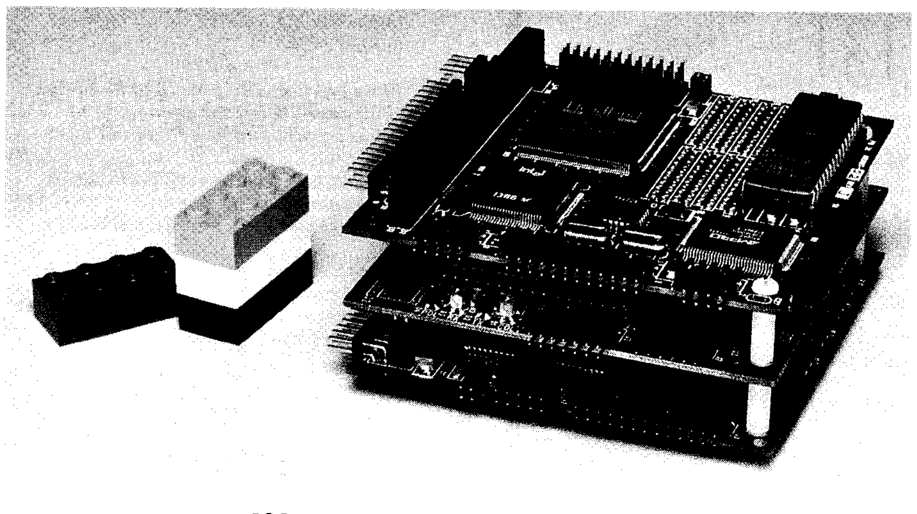


Photo 1: Self-stacking PC/ 104 modules are used like embedded ***system*** building blocks.

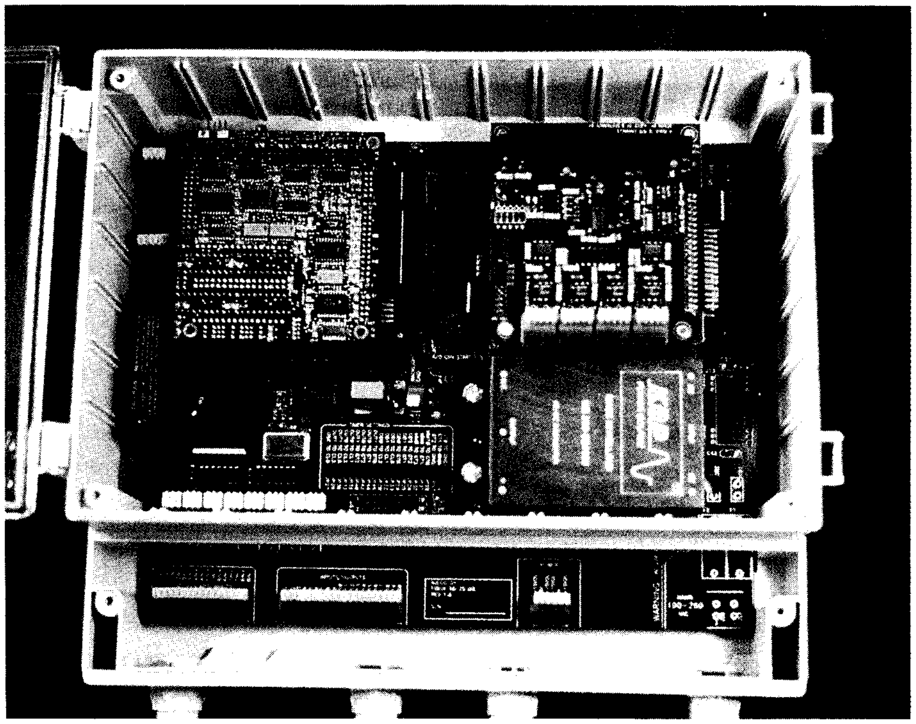


Photo 2: ***PC/104*** modules can plug into application-specific baseboards like macrocomponents. Application baseboards can contain data acquisition and control interfaces, keypad scanning logic, power conditioning circuitry, and any other electronics needed by the application.

embedded systems with **PC/I 04** modules, it's wrong to conclude that every bit of the electronics in your embedded system needs to be on **PC/I 04** form-factor modules.

In fact, due to the specialized nature **of** most embedded applications, systems often require unique functions or interfaces not available in off-the-shelf **PC/I 04** modules. In short, be prepared to do some PC board design in most cases.

Actually, the most common way to use PC/I 04 modules is as macrocomponents plugged into a custom, PC/104-expandable application baseboard. To meet specific application needs, the baseboard contains various nonstandard functions. As shown in Photo 2, the application baseboard could include signal-conditioning logic, circuitry to scan switches or push buttons, drivers for LEDs or LCD displays, DC power supplies or voltage converters, and serial and parallel I/O ports.

The system in the photo monitors flow rates in a gas pipeline. In this case, the application baseboard includes two PC/104 stack locations for a CPU and an isolated digital I/O module. A hybrid power-supply module, positioned directly on the baseboard, powers the electronics. Signal conditioning circuitry, real-world connector interfaces, a battery for the CPU's battery-backed clock, and miscellaneous components are also included directly on the baseboard.

Like most application baseboards, this one adopts the shape of its enclosure. As you can see, an application baseboard is like a system motherboard. However, instead of plugging I/O into a motherboard, consisting mainly of a CPU, the CPU plugs into a motherboard that consists mainly of the application's specialized I/O.

Besides providing efficient **system** logic and interfaces not readily available on PC/104 modules, an application baseboard can reducecosts by minimizing the number of PC/104 modules required. Also, a PC/I 04-expandable application baseboard that integrates several required functions might be the only practical way to fit everything needed by your application into the space available.

IF YOU CAN'T BEAT 'EM, JOIN 'EM

You can also expand PC/I 04-based embedded systems with desktop PC expansion cards. The easiest and least-expensive way to accomplish this is with a passive adapter cable. Such a simple interconnection is possible because the **PC/I 04** bus is **essentially** a normal PC/AT bus-it just has a different connector. Photo 3 shows a PC/104-to-ISA bus adapter cable.

Photo 4 presents a multislot PC/AT bus backplane that takes this approach to the extreme! Using this backplane, you can combine a stack of PC/I 04 modules with up to six PC/AT bus plug-in cards. A word of caution: with such a setup, scope out bus signals once you determine the system's final configuration. You need to ensure over-and undershoot, ringing, andcrosstalk are at acceptable levels. Also, be sure to maintain control over what may be added or altered in the future!

How can PC/104 with its 4-mA bus drive mix with PC/AT bus plug-in cards given the 24-mA drive of the PC/AT bus? Even though this is true, the maximum load

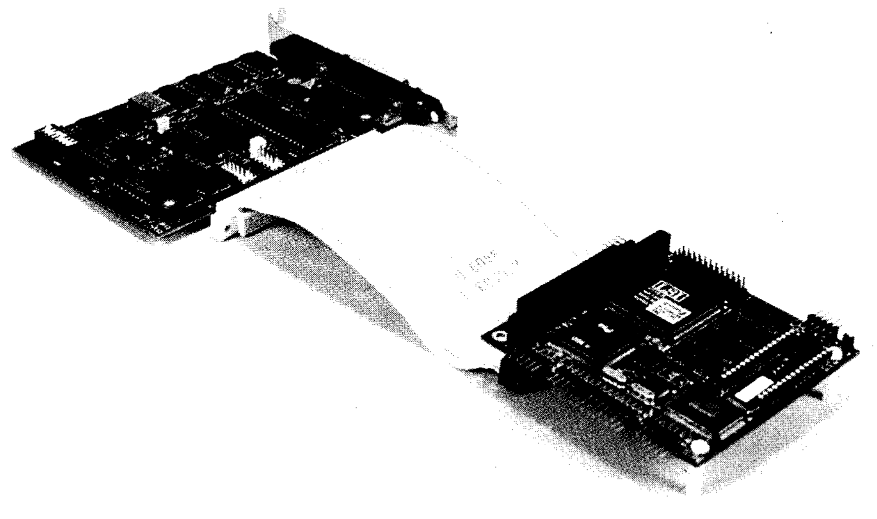


Photo 3: The signals of the PC/ ***104*** bus correspond ***1*** -to- ***1*** to the signals of the standard ISA bus signals. As a result, a simple ribbon cable can be used to connect between a ***PC/104*** CPU and a standard ISA bus expansion card. To make this cable, you need to use a *special, cable-mount* ***edgecard*** *connector from* ***PCD*** *Connectors.*

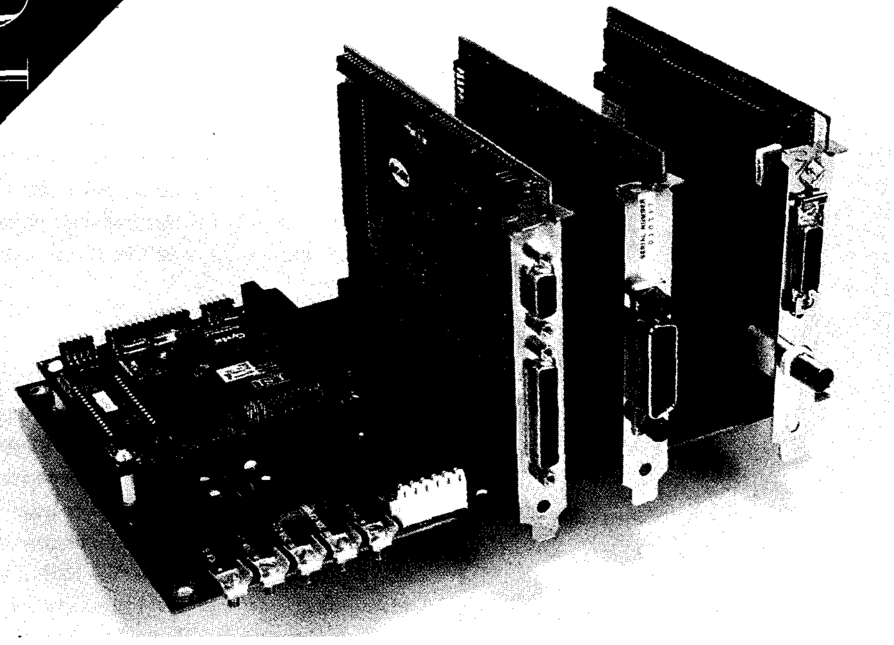


Photo 4: This ISA bus passive backplane from Douglas Electronics lets a ***PC/104*** module stack drive up to six standard ***8-*** or ***16-bit*** ISA *bus expansion cards.*

of a normal PC/AT expansion card is precisely the same as that of a PC/104 module: 0.4 mA. For this reason, it theoretically makes no *electrical* difference whether you add standard PC/AT bus cards or PC/104 modules to a PC/I 04-based system.

A TALE OF TWO BUSES

Another interesting alternative for expanding PC/I 04-based systems **results** from teaming up PC/I 04 and PCMCIA. These popular PC-related **standards** combine in a highly synergistic way to meet the needs of many types of embedded applications.

Table 1 compares some of the key attributes of PC/104 and PCMCIA. PC/104's compactness (3.6" x 3.8" x 0.6"), ruggedness, reliability, and inherent PC compatibility enable it to provide the core computer functions required by your embedded-PC-based application. A small stack of PC/I 04 modules typically includes the entire PC/AT motherboard logic (CPU, BIOS, DRAM, system bus), serial and parallel ports, disk controllers, display controllers, and a LAN interface. Also, because PC/104 modules mount rigidly to each other and other system components, the system you build with them should withstand the shock and vibration requirements of your application quite well.

Many of the characteristics that make PCMCIA suitable to the mobile computing also make it a good adjunct to PC/I 04 in embedded applications. In particular, PCMCIA's small size, low power requirements, and ruggedness are especially valuable. As well, PCMCIA cards are **especially appropriate** to PC-compatible hardware and software environments—after all, that's what the "PC" in PCMCIA is all about!

You might want to use PCMCIA cards to expand PC/I 04-based systems with the options popular for laptop computers. The growing variety of PCMCIA cards in this category include:

- ATA hard disks
- flash and battery-backed SRAM cards
- modems
- LAN adapters
- wireless communications interfaces
- multimedia interfaces
- GPS receivers

On the other hand, you'll find a better assortment of PC/104 modules for functions and interfaces specific to real-world data acquisition and control ap plications. Some examples of available PC/I 04 modules are:

- analog and digital I/O
- relay drivers
- counter/timers
- stepper and servomotor controllers
- touch interfaces
- video frame grabbers
- field buses: IEEE-488, Profibus, CAN, MIL-STD-1553, ARINC-429, Echelon LON
- CRT and flat-panel display interfaces
- digital signal processors (DSPs)
- power-supply modules for industrial and vehicular environments

STACK 'EM UP!

An important difference between PC/104 modules and PCMCIA cards is how they physically mount within the system. PC/104 modules stack rigidly with each other (and other system components), making them somewhat "permanent" system components.

In contrast, PCMCIA cards are **intended to be used as removable devices. This makes PCMCIA cards an ideal way to provide removable, yet rugged, mass-storage devices. When the system's operating software is located on a PCMCIA card, it's easy to reconfigure the operation of the**

	PC/1 04	PCMCIA
Size (sq. **in.)**	13.7	7.2
Max. component thickness (in.)	0.435	Type 1: 0.13 Type 2: 0.20 Type 3: 0.40
Power (typical)	1–2 W	0.1-0.3
PC compatible	yes	yes
Supports DOS drivers	yes	yes
Rugged, shock proof	yes	yes
Rigidly attached	yes	no
Operating temp.	0–70°C	0–55°C
Self stacking	yes	no
Readily removable	no	yes

Table 1: A *comparison of PC/* 104 *and PCMCIA* quickly points out how well they fit ***together*** in embedded *applications where space, power consumption,* and reliability ore important.

	PC/1 04 and PCMCIA	Normal PC/AT
Size (cubic in.)	50	1500
Weight (lb.)	1	40
Power	5–10 W	50-100 w
PC compatible	yes	yes
DOS/Windows compatible	yes	yes
Rugged, shock proof	yes	no
Operating temperature:		
Standard	0–55°C	10–50°C
Extended	–20–+85°C	N/A
Expansion "slots"	2-3 PC/1 04 & I-2 PCMCIA	5–6 ISA
Removable storage media	flash memory	floppy disks

Table 2: Sometimes ***the*** *ideal system involves a combi-*nation of ***PC/1*** 04 and ***PCMCIA*** technology. The combination ***offers*** all the punch-and ***then*** *som-f a* typical PC/AT ***system*** at a fraction of the *size,* weight, and power.

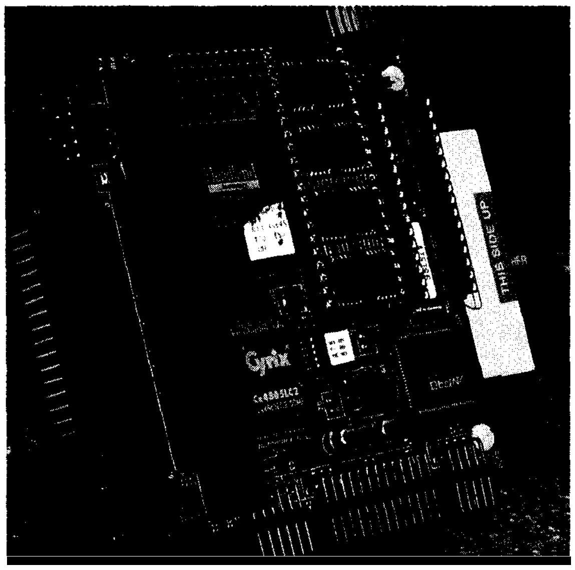


Photo 5: PCMCIA *can be used like a miniature backplane bus in **PC/104** applications. This slot* PCMCIA ***interface** from **Ampro** lets **you** combine one or two* PCMCIA ***cards** with a **PC/104 CPU.***

system or update the software to fix bugs or enhance features.

PCMCIA slots also let you add options or upgrades in a PC/104-based embedded system. Over time, you can pretty much count on the growing popularity of PCMCIA in the laptop and other mobile computing mass markets to spawn an increasing variety of PCMCIA modules.

LIKE A TINY BACKPLANE

Photo 5 shows a two-slot PCMCIA interface combined with a PC/104 single-board PC/AT computer. This stack can be expanded two ways:

- by stacking one or more PC/104 modules on a CPU module
- by inserting one or two PCMCIA cards in the two-slot PCMCIA interface.

You could think of this hybrid system's dual-slot PCMCIA interface as a tiny backplane supplementing PC/104's stackability.

In a sense, the PC/104 module stack shown in Photo 5 is a lot like a "normal" PC/AT system. Typically, a PC/AT system consists of a CPU, RAM, and several ISA expansion slots on a motherboard, mass-storage interfaces, and a display controller. To configure the system for a specialized application, insert various ISA expansion cards in the five or six spare slots on the motherboard.

As Table 2 indicates, the equivalent PC/104 and PCMCIA system contains the same functions, provides similar expandability, occupies 1/30 the volume, consumes less than 1/10 the power, and weighs 1/40 as much!

ROLLING YOUR OWN

If you need custom expansion modules, you'll quickly discover that PC/104 is a better choice than PCMCIA. PC/104 provides a standardized, easy-to-implement form factor to facilitate design.

Although PC/104's horizontal and vertical dimensions are strictly defined by the module's spec, in most cases you won't have a hard time meeting its requirements using normal components and a bit of care.

On the other hand, squeezing your bright idea into a custom PCMCIA card is a daunting task. It is likely to be a difficult and costly process because of PCMCIA's severe restrictions on component thickness and other packaging requirements.

CONCLUSION

Unlike ordinary bus standards, PC/104 offers a greater degree of flexibility and does not restrict you to a typical "stack and rack" mentality. To fully appreciate the wide variety of ways you can use PC/104 modules, think of them as macrocomponents, which can be used both as plug-in components and self-stacking building blocks.

Other popular PC-oriented standards such as PCMCIA combined with PC/104 help you achieve even greater flexibility and functionality. PCQ.EPC

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

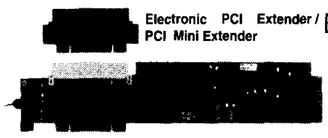
IRS

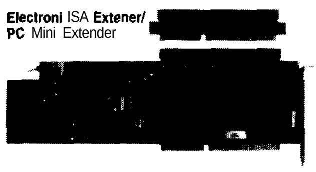
422 Very Useful
423 Moderately Useful
424 Not Useful

HOT ***SWAPPING***

ELECTRONIC EXTENDERS

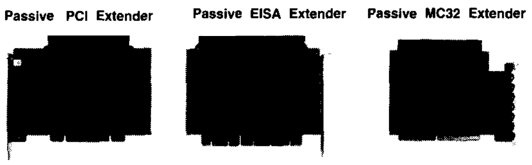
For PCI, ISA, EISA, VESA, Micro Channel and NuBus.





☑ Insert/Remove Cards w/the PC Power On!
☑ Save Time Testing and Developing Cards
☑ Save Wear on Your PC from Rebooting
☑ Adjustable Overcurrent Sensing Circuitry
☑ NO Fuses, All Electronic for Reliability
☑ Single Switch Operation w/ Auto RESET
☑ Optional Software Control of All Features
☑ Breadboard Area for Custom Circuitry

Full line of passive extenders for PCI, ISA, EISA, VESA, MC and NuBus.



☑ Classic Extenders with All Lines Clearly Marked for Convenience
❑ Save Time Testing, Developing, and Repairing Computer Adapter Cards
☑ Available Optional Tes Pin Headers Under Top Connector

1 Year Warranty and 30 **Day Money Back Guarantee**

AZ-COM INC.

3343 D VINCENT RD
PLEASANT HILL, CA 94523 USA
800-209-2418 FAX 510-947-1900

Call our 24 Hour Fax-On-Demand System for FREE Information.

#216

Applied PCs

Russ Reiss

Embedded PCs

Who, What, Where, Why, When, and How?

If you *want to install* embedded PCs, *you've got to* start at ***the beginning***—*outlining* ***the*** *basics and making sure everyone is speaking the* same language. *Subsequent columns will delve into the nitty gritty of embedding a PC.*

No doubt, you've been using embedded microcontrollers and PCs for years, although for totally different tasks. While the idea of getting your personal computer inside an embedded design sounds a little far-fetched, recent technological breakthroughs have made PCs almost commonplace in the embedded world.

With the miniature, low-power, flexible embedded PCs now available, a seemingly unlimited number of applications benefit from the use of embedded PCs.

And, there are many compelling reasons why you might wont to use an embedded PC over either a non-PC-based computer, a desktop PC, or an embedded microcontroller.

That's what this column is all about: applying these powerful, flexible, fast-to market, easy-to-program, convenient-to-interface solutions to your new designs. I plan to explore a wide range of topics related to selecting, specifying, and applying embedded PCs. I'll present and contrast alternative solutions, discuss some of the not-so-evident aspects, point out products available in the marketplace, and attempt to bring it all to life with real examples. But, before I get carried away, I want to make sure we're all on the same wavelength.

This month, after defining a few terms, I'll take a look at exactly what embedded PCs are, who uses them, where, why, when, and how. Following this, I'll give you a sneak preview of what's coming in future columns.

EMBEDDED PCs DEFINED

Embedded PCs represent a powerful and fast-growing subset of what is more generally known as embedded systems, embedded computers, or embedded controllers. These terms each imply something **slightly** different, and you should be aware of the distinctions between them before you can appreciate how embedded PCs fit into the picture.

An embedded system typically describes a device which packages computer

embedded systems is the user's lack of awareness about the presence of the computer. To the user, the application is what's important, not that a computer implements the solution.

Embedded computer and embedded

is incorporated into the solution or device. As with microcomputerand microcontroller, the distinction between them st

control-ter is typically used when a

ler does not possess adequate performance capabilities.

So, what then is an embedded PC? It's an embedded computer which possesses most of the attributes of a personal computer. Though not restricted to IBM PC-compatible systems, these systems represent the largest percentage of embedded PCs available today.

What characterizes embedded systems is not the type of computer used, but the computer's relation to the application it is used in. Typically, the computer is buried in another device or system and, while it is typically the critical component of the system, the user is oblivious to the specific computer chosen.

These days, embedded systems involve a wide range of computers. At one end are applications which embed microcontrollers into everyday appliances like sewing machines, microwave ovens, automatic lawn sprinklers, and TVs. At the other end are very powerful, special-purpose supercomputers embedded in specialized equipment which perform, for example, a dedicated portion of a complex spacecraft navigation or control mission.

But, these scenarios represent the extremes. Neither category of applications would generally benefit from the use of an embedded PC.

So, what kinds of applications do embedded PCs suit? How do we recognize them? What factors favor an embedded PC over an alternative approach? While the detailed answer to these questions will unfold in future articles, let me identify some generic answers here.

WHO

An embedded PC is perceived quite differently from the point of view of the designer versus that of the end user.

Users are often completely unaware of the computer's presence. To be sure, each user has an opinion about the system, but none really cares that a computer—often an embedded PC-runs the inside show.

However, if the users have computer savvy, they might recognize the familiar Windows GUI or appreciate that data conveniently pipes into a spreadsheet program for reporting results on a conventional laser printer. Few would suspect, however, that inside, at a detailed technical level, the device was very much like their familiar desktop PC.

While embedded systems are characterized more by the application than the computer, the computer acts as the heart of the entire system. To the system designer and implementer, the computer is the focus of attention. However, since the computer is often buried among sensors, actuators, user interfaces, and other I/O devices specific to the application, it must be treated as a system moreso than just a computer.

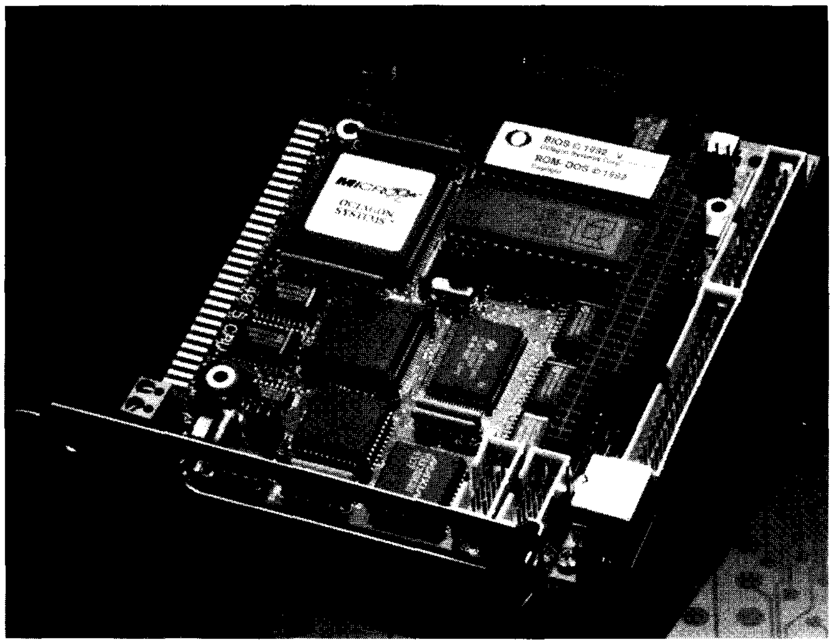


Photo 1: Octagon's 40 IO Micro PC board plugs into conventional ISA slots or operates standalone with ***onboard*** *'486 CPU as well as serial and parallel ports, floppy disk controllers, and memory.*

One of the goals of designers and engineers is often to hide the computer from the user. This goal does not stem from a desire for secrecy, but because users should get to deal with on instrument on their own terms. The computer should aid and not interfere with the task at hand. This is a particular challenge with embedded PCs since a PC-especially from a software perspective-comes with a beguiling collection of familiar tools which the implementer often uses for shortcuts.

For example, using a conventional VGA monitor for the user console is tempting because it is already fully integrated into the system and fully supported by interface hardware, device drivers, and high-level language tools. However, the question of whether or not a VGA display is the appropriate user interface should receive proper attention during the system design phase. It should not simply appear in the system by default.

This default response often occurs with peripherals such as mice, too. While a mouse is a convenient and useful for a desktop computer, it often is cumbersome and slow for an embedded system! Though more difficult to interface and support in software, another sort of custom I/O capability might be more convenient for the user.

However, the wider range of I/O devices available for the PC means that creative and novel solutions benefit from the presence of a PC. For example, if an application could benefit from limited speech recognition, all thedesigner needs to do is specify a ready-to-go, off-the-shelf board, which usually comescompletewith drivers and support software.

And if you need to network multiple embedded systems, what could be more convenient than a conventional PC LAN? All the parts are there (at amazingly low cost), fully proven, supported, and ready to drop in. Even wireless links are available.

But the trick, once again, is to blend readily available resources into an appropriate system through a judicious mix of hardware and software. Extreme care and caution must be taken to avoid the temptation of forcing inappropriate restrictions on the end user for the sake of implementation expedience!

WHAT

So what characterizes an embedded PC? What makes it the best choice in some instances rather than an embedded computer or controller?

Contrary to popular opinion, processor speed is not usually the overriding

Photo 2: Interactive Display Systems packages a complete PC, keyboard, and color **LCD** dispiay in its **SlimLine** rack-mount **system.** When **the display** is folded down into storage position, the *unit occupies* ***only*** 3.5" ***of rack*** *space.*

determining criterion. More often, a microcontroller lacks powerful mathematical capability, memory addressing capability, or off-the-shelf peripheral interfaces. Certainly, you could add floating-point math, banked-memory controllers, and display and disk controllers to a microcontroller. But, this is time consuming from both a hardware and software development point of view.

A non-PC-compatible microcomputer might offer the processing power needed, but it often comes up short on available peripheral interfaces and software support tools. Also, learning the peculiarities of a new microcomputer is time consuming and expensive.

An embedded PC is the best choice when a number of its attributes suit the application at hand. One would rarely use an embedded PC in the design of electronic controls for a microwave oven or an automatic lawn sprinkler system. Any number of embedded controllers represent a less costly, simpler, and completely adequate solution.

On the other hand, many other applications (e.g., medical instruments, data acquisition and processing systems, vehicular controls, operator displays and user interfaces, advanced position-control systems, automatic teller machines, auto mated toll booths, video and other games, and so on) beg for many of the features of a common PC. The obstacle until just recently has been the PC's size, packaging, high power consumption, and lack of ruggedness.

A few applications can actually be implemented successfully by simply repackaging a conventional desktop PC motherboard with its ISA, EISA, VESA, and PCI slots and peripheral interface cards. I supervised the implementation of an automated teller machine that successfully used this approach in a design intended for the benign environment of a bank lobby.

However, the poor mechanical stability, connector reliability, and heat flow associated with conventional PC motherboards would make this approach totally unsuitable in a harsh environment.

Here's where a wide variety of embedded PC formats shine. VME, STD bus, PC/104, I-Pack, SBX, and proprietary single-board solutions all strive for higher reliability in a more suitable form factor. Octagon's Micro PC form factor shown in Photo 1 illustrates the combination of conventional PC ISA bus, PC/I 04 expansion slots, and dedicated peripheral controllers all on one board.

Some novel packaging approaches exist too. You can use a flat-panel display with an entire computer housed inside. Or, when I/O requirements are minimal, consider a "computer within a keyboard" solution, complete with IAN capabilities. For space-orsecurityconscious rack-mount applications, interactive Display Systems offers a system with pop-up display, slide-out keyboard, trackball, power supply, and full-capacityembedded PC all in a unit that slides into 3.5" rack height (Photo 2).

Regardless of which form factor you select, you'll find yourself in familiar territory when you boot the system and see the familiar BIOS and operating system signs (unlessyou'vesuppressed them). The board, stack, or box you select to meet your environmental and packaging needs will undoubtedly be more rugged and reliable than the PC on your desk. While it probably won't resemble it, it sure as heck operates like it!

WHERE

Where does one use embedded PCs?

- where the environment is more restrictive
- where the system might be exposed to physical extremes of temperature, shock, vibration, or humidity
- where space is at a premium or mounting constraints are awkward
- where power is available from other than a steady, clean 1 10-VAC (220-VAC) wall outlet
- where available power is minimal
- where the system must put itself to sleep for extended periods and then wake up to perform a task autonomously without human intervention

In other words, embedded PCs work in potentially every application outside of a home and office setting!

Embedded PCs are also ideal when the embedded system must communicate with other systems. For example, through IAN support, factorycontrol systems link data-acquisition systems and management-reporting systems or facilitate the transfer of remotely collected data back to the laboratory where it is further processed by conventional PCs. The commonality of disk and data formats between the remote and local systems is a real convenience and time saver.

They are also ideal where it is desirable to offer a range of processing capabilities. Since PCs are in manywayssoftware compatible with everything from a lowly 8088-based unit to thelatest Pentium chip, it is simple to configure an assortment of systems with a spectrum of capabilities (and prices) while retaining the same core software.

This flexibility lets users keep costs low to meet current needs, while retaining the option of upgrading in a modular fashion to a more advanced system in the future without completely scrapping earlier investment. It also provides a path of growth to incorporate new, still nonexistent technologies into an embedded system.

WHY

Why use embedded PCs? Well, because they

- pack a lot of processing power in a small space
- operate from low power
- come fully equipped with BIOS and OS support
- are familiar and friendly to use
- support high-level languages and debugging tools
- come packaged in a wide variety of formats
- are supported by a wide range of peripheral interfaces
- are more rugged and reliable than desktop computers
- run a plethora of off-the-shelf software
- provide a softwarecompatible range of processing capabilities
- offer a migration path for the future
- are familiar to a large number of design engineers, programmers, and product users

Need I say more?

WHEN

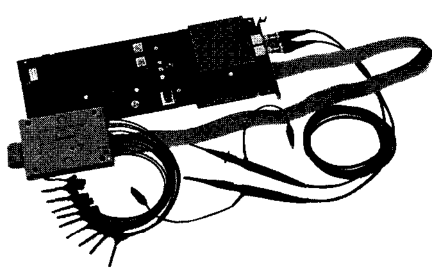
When is an embedded PC the best choice? I've touched on a few physical factors like ruggedness, reliability, immunity to high (or low) temperatures and high humidity, and compact size. But, there are other, more subtle factors that often bend the decision in favor of an embedded PC approach.

Embedded PCs improve time to market, **especially** when development resources are scarce. There are probably more design engineers and programmers familiar with the PC than any other single-computer platform. The learning curve is much less than for any other solution.

These advantages are not without risk, however. Would-be embedded-system implementers often have no knowledge of or experience with time-critical, real-time embedded systems. Many have completely

PC-Based Instruments
200 MSa/s DIGITAL OSCILLOSCOPE

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



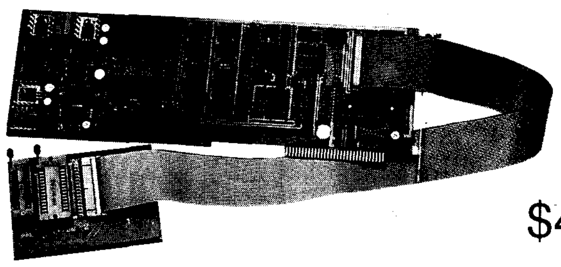
DSO Channels
2 Ch. up to 100 MSa/s
or
1 Ch. at 200 MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

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

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

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
PIC
etc..

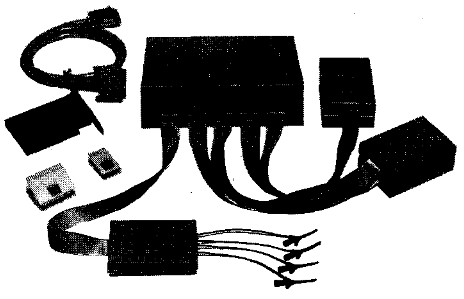


$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

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



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

Call (201) 808-8990
Link Instruments
369 Passaic Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786

#217

left the "details" of I/O up to an operating system that not only may be found lacking, but may get in the way.

My point: implementing most embedded systems is quite different than implementing a payroll or billing system. You still need an embedded-system engineer, not just a PC programmer to pull off most successful designs.

The embedded-system engineer has available a plethora of tools-especially in software-which are widely accessible, less costly, and more user friendly and familiar than tools for competing systems. And, I'm not talking simply assemblers, C compilers, and diagnostics.

In some cases, complete end-user software already exists which can be purchased at reasonable prices and handles much or all of the system requirements. Consider the proliferation of what might generally be called operator console software such as Wonderware's InTouch, pictured in Photo 3.

These programs can instantly transform a conventional (or flat-panel) monitor screen into a customdesigned front panel for your new system complete with the dials, gauges, bar graphs, annunciators, and push buttons accessible to the user via a convenient touch panel. There are also full-blown programs for data-acquisition, FFTs, and displaying histograms and strip charts on a screen or printer. Some software is designed specifically for the embedded PC while others carry over from the desktop PC.

With embedded PCs piggybacking the desktop PC marketplace, there is a built-in guarantee of new and more powerful solutions arriving each day. An embedded PC offers an unparalleled migration path. One can be sure that tomorrow will bring more **powerful, fully compatible CPUs,** and an ever-growing choice in peripherals and controllers with phenomenal performance and lower cost.

and HOW...

How do you use embedded PCs? That's what this column is all about.

In coming EPC sections, I'll look at many important issues that arise when applying embedded PCs. For example, next quarter, I'll present the various board and bus configurations which are available and consider novel schemes for packaging your nifty embedded PC.

In a following installment, I'll look at the issue of selecting, interfacing, and programming displays for embedded systems. After all, it's unlikely that you'll want (or be able) to stick that big fat VGA CRT monitor into your next miniature, portable thingamajig.

In later columns, I'll study techniques for powering embedded PCs with other than conventional line-powered supplies, and look into unique peripherals for embedded PC applications. I'll investigate topics ranging from the familiar PCMCIA slots to specialized rugged hard drives, flash-memory systems, and ROM-DOS to exotic peripherals such as GPS navigation satellite receivers.

Finally, real examples will illustrate many of these issues. I wrestled long to find a suitable vehicle that could encompass such a wide range of topics yet not be so particular that it would become boring. Finally, I realized that o long-standing pet project of mine-the computerization of my old RV-represents an ideal vehicle (pun intended).

Computers in an RV can serve many purposes: engine, battery, and lighting monitoring; security; navigation; and mundane needs like word processing and telecommunications. Furthermore, an RV possesses a number of attributes, including limited space, high shock and vibration, fluctuating and electrically noisy DC power, and need for day/night operation, all criteria that provide fertile ground for investigating the application of embedded PCs.

While it is unlikely that anyone will want to precisely duplicate the examples I'm presenting, I'm hopeful you will find them novel, interesting, and instructive. I trust the principles will carry over to your own applications. APC.EPC

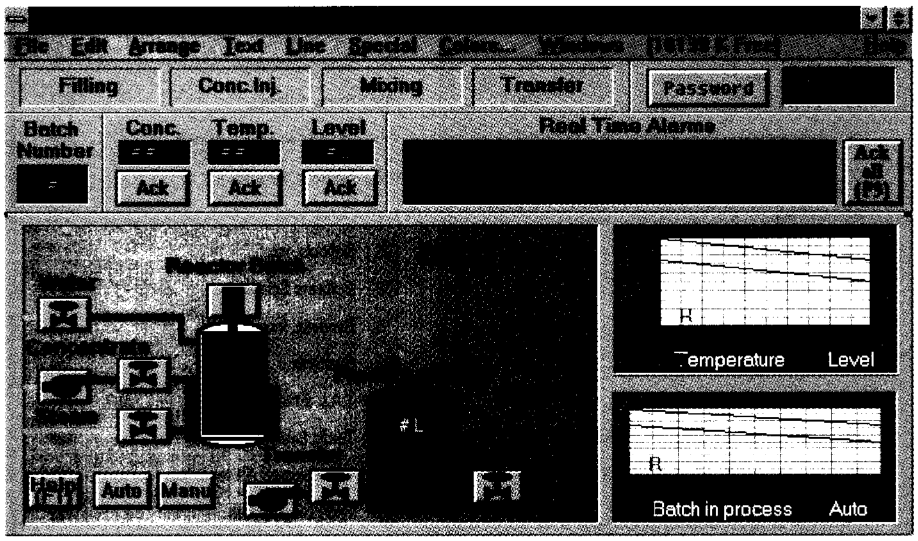


Photo 3: Considerable ***time*** and programming effort can be saved by using off-the-shelf PC ***software*** in embedded systems. ***Wonderware's InTouch,*** for instance, permits rapid implementation of ***operator control-panel displays in diverse applications.***

Russ Reiss holds *a Ph.D.* in *EE/CS* and has been active in electronics for over *25 years as industry consultant,* designer, college professor, *entrepeneur, and company president.* He *may be* reached at *russ.reiss@circellar.com* or *70054.1663@compuserve.com.*

SOURCES

Octagon Systems
65 10 West 91 Ave.
Westminster, CO 80030
(303) 430-l 500
Fox: (303) 426-8 126

Interactive Display Systems
Division of General Digital Corp.
198 Freshwater Blvd.
Enfield, CT 06082
(203) 741-7171
Fax: (203) 741.7071

Wonderware Software Development, Corp.
16 Technology Dr., Ste. 154
Irvine, CA 92718
(714) 727.3200
Fox: (714) 727.3270

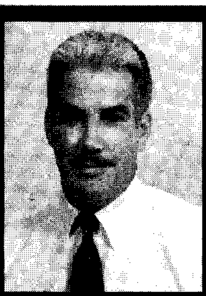
I R S

425 Very Useful
426 Moderately Useful
427 Not Useful

IC de Light

SILICON UPDATE

Tom Cantrell



Texas Instruments boasts the first and only light-to-frequency converter which integrates a photodiode with signal conditioning so that it produces digital input. Curious? Then, put your wager in with Tom.

The kids are parked in front of the TV, so now's my chance to sit down in front of the old Mac and get to work on my next *INK* article.

Sure enough, I don't even make it past the happy Mac face before the mild-mannered morning takes a bad turn. A large crashing sound is accompanied by what sounds like a primal scream in a group-therapy session.

Storming into the living room, I stumble into a combat zone including martial arts (accompanied by the requisite battle cries), sword play (never considered a mop a lethal weapon until now), and lobbing of Lego hand grenades.

What's this? I'm sure I left the TV on PBS. Where's the purple dinosaur or the old guy who wears the sweaters?

Instead, the kids are getting inspired by the Mighty Barphin Dour Strangers, who deserve nothing better than a mighty big shot of morphine (somewhere between coma and lethal) as far as I'm concerned.

As I write this, debate once again rages on the influence of TV on kids. One side claims it's much ado about nothing while the other blames it for most societal ills.

Me, I come down squarely in the middle. First of all, anyone who imagines violent TV doesn't affect kids either doesn't have kids or never sees them. On the other hand, I suspect letting politicians choose what we view is rather an iffy proposition.

I believe in censorship, as long as it's Big Daddy and Mommy, rather than Big Brother at the controls. Unfortunately, Big Daddy or Mommy often have other Big Stuff to do.

So, I've pondered the problem of how to control the kids' viewing habits without being chained to the remote. Dismissing the obvious (shotgun the TV) or infeasible (teach kids to be respectful and obedient) leaves technology as the only hope.

Cable boxes have a channel lockout feature. VCRs can be programmed (where is that !@#$ing manual?) to turn on and off and record a particular channel. So, I imagine it's only a matter of time before some clever TV engineer (likely one with kids) comes up with the bright idea of making a kid-proof TV (i.e., one which can lock out certain channels at certain times). ***Editor's note: This feature already exists for high-end TVs.*** Sanitizing the TV would take a trivial amount of extra logic (if any)

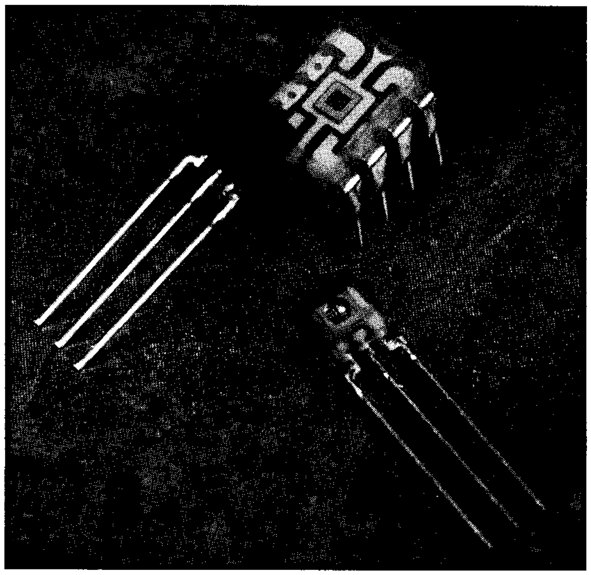


Photo 1—*The TSL23x series of light-to-frequency converters includes the '235 and '245, which are three-lead devices, with the '245 (dark package) optimized for IR applications. The 8-pin '230 features programmable sensitivity and output scaling.*

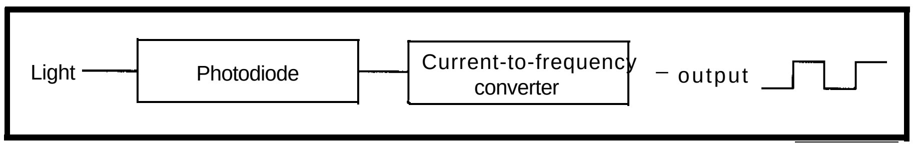


Figure I--The *TSL23x light-to-frequency* devices are among *the first* sensors *to* completely bridge *the analog-to-*digital gap.

since modern units already have a CPU and EEPROM.

However, the emergence of kidproof TV sets won't help the millions with the older models. So, my bright idea is to make a little box (safely mounted high on the wall) with a micro, real-time clock, and IR LED that acts as a censorial Cyclops by prudently selecting the mildest-mannered program every half-hour.

Diving right into the design, the first step is to come up with a way to learn the channel-select codes for my particular TV. Fortunately, the subject of IR and remotes has been well-covered in previous *INK* articles, explaining a variety of arcane formats and modulation (typically at 40 kHz) schemes.

However, we really don't have to understand the details to get the desired result. Instead, it's quite adequate to simply train the gadget to mimic the TV remote [i.e., capture what the remote outputs to select a channel and, without the need for any interpretation, play it back).

Thus, like a learning remote, my cyborg censor can work with any and all TVs.

A BIT OF LIGHT

In the old days, I'd collect a handful of components including a photodiode, op-amps, and their requisite resistors and capacitors, and have a go at it. Actually, I'd probably give Steve Ciarcia or someone equally blessed with analog know-how a call and beg for help.

But, as of a couple of years ago, things got a lot easier. Refer back to "Op-amp Terminators" *(INK 43)*. There, I described the Burr-Brown OPT201, a device integrating all the aforementioned components in a single 8-pin DIP which easily connected to an A/D converter.

Now, thanks to the TSL23x series of intelligent optosensors from Texas Instruments, the task of turning light to bits is even easier. Following the "Go digital, young man" catch phase, TI ups the integration ante by eliminating the need for an expensive A/D converter port. Indeed, TI claims their light sensor is the first to totally bridge the analog-to-digital gap.

The offerings pictured in Photo **1** include the TSL230, TSL235, and TSL245 and are based on a common architecture. As you can see in Figure 1, the chips convert light input to a digital frequency output. At only $1.75 for the '235 or '245 (in 1000s) and about a buck more for the '230, a ride on the digital bandwagon isn't going to empty your wallet.

The '235 and '245 are packaged in three-lead packages, which include power (2.7-6 V), ground, and output. The latter is a square wave whose frequency corresponds quite linearly to the light intensity, as shown in Figure 2.

The main difference between the devices is the photodiode spectral response, as shown in Figure 3. The '235 covers the UV and visible spectrum while the '245, thanks to its dark package, is optimized for IR response. The latter sounds like just the ticket for any gadget that wants to work with the worlds ever-growing collection of remotes.

The '230, packaged in an 8-pin DIP (see Figure 4), features the same spectral (UV to visible) response as the '235. The extra pins hook to additional features including programmable sensitivity and output scaling. SO and S1 can select three levels of sensitivity: 1, 10, and 100 times. The fourth combination of SO and S1 (both low) puts the '230 into low-power mode, which cuts power consumption from low (3 mA active) to trivial (10 µA).

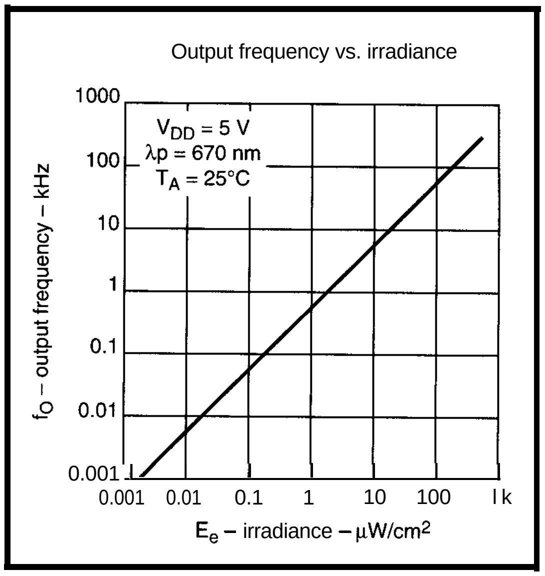


Figure **2—The** device's *output is quite linear. In the case of the '235, it's* roughly *a kilohertz per µW/cm².*

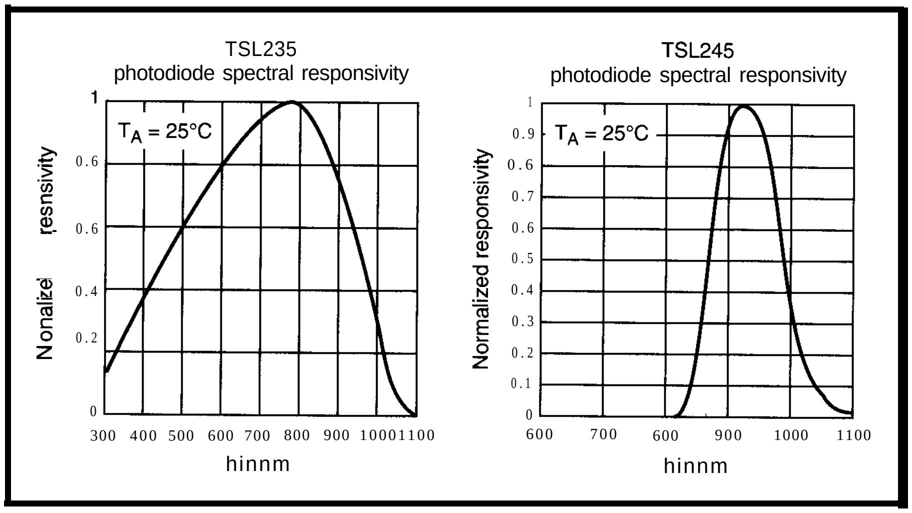


Figure *3-The '235 and '245 are essentially* **the** same, except *the latter* is optimized *for IR applications.*

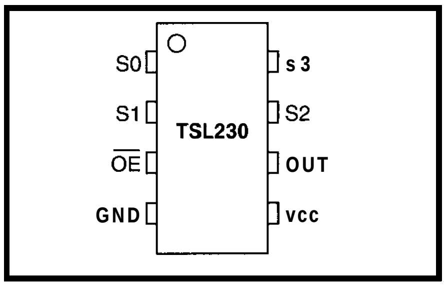


Figure 4—*The* **extra pins** *of the '230 let* **designers dynamically adjust** *the* **sensitivity** *and output-frequency* **divide** *ratio.*

You might suspect SO and S1 control (as in a discrete design) a programmable gain amp. Actually, the technique TI refers to as *electronic aperture* is much simpler. It turns out there are actually 100 closely matched photodiodes (10 x 10 matrix) on the chip. Selecting the sensitivity is simply a matter of enabling either 1, **10,** or all 100 of them.

The photodiodes' output is in turn fed to a switched-capacitor charge-metering circuit (i.e., A/D converter surrogate) that produces a frequency output.

S2 and S3 set the scaling of the frequency output as a divide by 1, 2, 10, or 100, allowing designers to personally tune the tradeoff between measurement speed and the timing capabilities of the attached micro. All but the divide-by- 1

or the low half can be measured.

F i n a l l y , t h e r e ' s a n
E n a b l e) p i n t h a t l e t s t h e

color applications.

WE INTERRUPT THIS PROGRAM

Taking a light reading is conceptually simple (i.e., just measure the f r e q u e n c y o f t h e o u t p u t) , b u t t h e r e a r e some gotchas to watch out for depending o n y o u r a p p l i c a t i o n , a n d t i m i n g resources.

The speed with which a single reading can be taken obviously depends on the light level and can vary from a second or so (dark) to 1 us (full-scale illumination). If the speed of r e s p o n s e i s n ' t c r i t i c a l , t h e u s u a l oversampling and averaging techn i q u e s c a n b e u s e d t o f i l t e r o u t r a n d o m or periodic noise such as the 60 Hz c o m m o n l y f o u n d i n A C

Whether using a dedicated timer/counter, a general-purpose input, or an interrupt request, the general principle is the same. You can either measure how many pulses occur in a given time or measure the elapsed time for a given number of pulses. In either case, it's no real brain teaser, as shown by sample code for the PIC (Listing 1).

With a digital interface you may wonder, as I did, why TI didn't just include a counter to take care of the housekeeping. However, I realized that the decision to do so is hampered by the same concern that keeps analog sensors from stampeding to digital overnight.

When faced with the proposal to include an A/D converter, sensor folks always come back with "how many bits?" Though the concern won't stop the inexorable move to digital, it is valid. For one customer, 8 bits of resolution may be adequate, while another might need I2 bits.

For the '230, the analog to analog question revolves around timing resolution. In other words, the resolution of the '230 is infinite, subject to the minor constraint that you have an infinite-resolution timer handy. For example, it turns out the previous PIC routine achieves the equivalent of 8-bit resolution, limited by the speed of the code that increments the count (LOOP3). O u t p u t

The TI literature shows that parts with dedicated timer/counters (examples they give include the TMS370 and MC68HC11) can achieve up to 16-bit resolution. However, a possible

t h e c o u n t e r / t i m e r (t y p i c a l l y , 8 bits) may overflow, limiting the range of measurement. Fortunately, the f r e q u e n c y - s c a (S2 and S3) f e a t u r e can be exploited to constrain the '230 output and, along with sensitivity a d j u s t m e n t (S0, S1), maximize d S dynamic range.

KEEP IT CLEAN

Included with the literature was a '230 simulator that runs on PCs (both DOS and Windows versions are on the disk). Given the simplicity of the device, it's not surprising the simulator is a snap to use. As shown in Photo 2, it's easy to set up the '230 sensitivity and scale factors, define a light source (wavelength and intensity), and observe the resulting output.

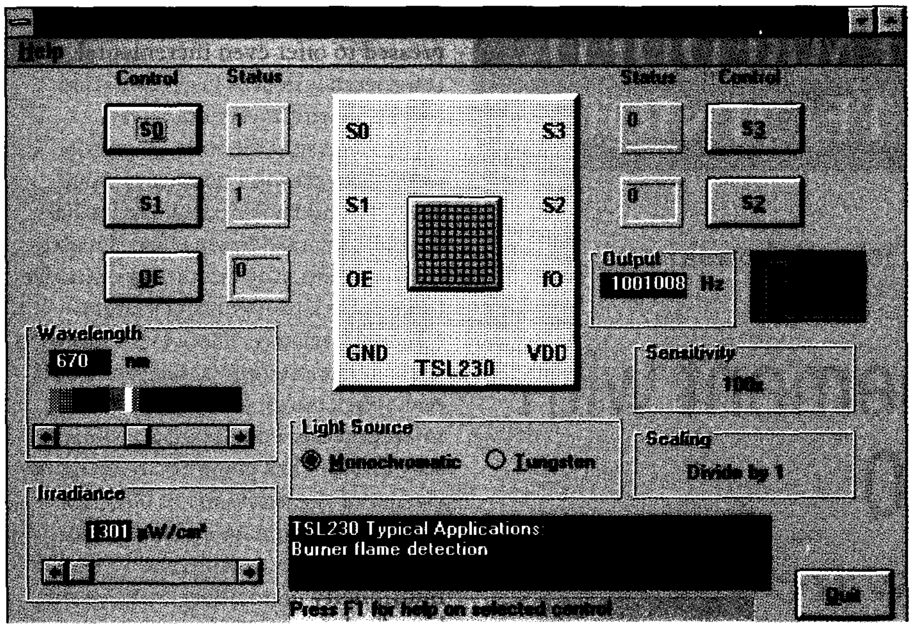
About the time I was ready to grab the soldering iron, I realized my TV cop idea had a few flaws. Sure, I considered the fact the kids would manually change the channel, but figured I could just repeatedly send the programmed channel command every 30 seconds or so.

However, I suspect it wouldn't take more than a few nanoseconds for the little conspirators to figure out all they have to do is block the TV's IR window. A casual wipe down with a PBJ sandwich would be all it takes to jam (er, jelly?) the link.

Instead, I now rely on a neural network. In other words, I've recently had luck encouraging (OK, bribing) the oldest kid with computer games

Listing I--The *simplest possible code (in* this case *for a PIC) simply finds an edge and then starts* counting *until the next* *Note the* **restriction** *that* **divide-by-I mode can't** *be* used since *the code relies on the fact that divide-by-2 (or greater) modes generate a square-wave output.*

```
; 8-bit 16C5x od measurement routine for PIC
; Assumes input signal is 50% duty cycle (divide by 2)
; Does not account for overflow
       MOVLW    FFH            FF is hex for 255
       MOVWF    PCNT           Initialize period counter
LOOP1  BTFSC    PORTA,0     ;  Check port
       GOTO     LOOP1          Wait for low level
LOOP2  BTFSS    PORTA,0     ;  Check port
       GOTO     LOOP2          Wait for high level
LOOP3  INCF     PCNT           Begin counting
       BTFSC    PORTA,0     ;  Check port
       GOTO     LOOP3          Count while high
; Value in PCNT (8 bit) represents period/Z
```



Status: S0 1, S1 1, OE 0, S3 0, S2 0

Output 1001008 Hz

Wavelength 670 nm

Sensitivity 100x

Light Source ● Monochromatic ○ Tungsten

Scaling Divide by 1

Irradiance 1301 µW/cm²

TSL230 Typical Applications:
Burner flame detection

Press F1 for help on selected control

Photo 2—The PC-based simulator shows the TSL230 output under various setup (scaling and sensitivity inputs S0–S3) and lighting (wavelength and intensify) conditions.

(educational, of course) to police the TV. Indeed, he's become quite zealous and can deal with the younger ones' plots or unexpected program schedule changes far better than any CPU.

Nevertheless, there are plenty of other light-sensing applications that can take advantage of TI's know-how.

Thanks to the ever-lower price of technology, you may find the chips popping up in unexpected places-cars, games, toys, cameras, and even your dishwasher. In the latter application, the '230 analyzes an illuminated water sample to control the flow, temperature, and timing of the wash cycle. ❑

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 *at* (510) 657-0264 *or by fax at* (510) 657-5441.

CONTACT

Texas Instruments, Inc.
Literature Research Center
P.O. Box 172228
Denver, CO 80217
(800) 477-8924, Ext. 3411
(303) 294-3747, Ext. 3411

I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

DESIGN CIARCIA WORKS

Does your Big-Company marketing department come up with more ideas than the engineering department can cope with? Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?

Steve Ciarcia and the Ciarcia Design Works staff may have the solution. We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems.

Whether you need an on-line solution for a unique problem, a product for a startup venture, or just experienced consulting, the Ciarcia Design Works is ready to work with you. Just fax me your problem and we'll be in touch.

REMEMBER... A
CIARCIA DESIGN WORKS
FAX: (203) 871-8986

Microcontroller Networks (µLANs)

With Cimetrics' 9-Bit µLAN you can link together up to 250 of the most popular 8- and 16- bit microcontrollers (8051, 80C196, 80C186EB/EC, 68HC11, 68HC16, PIC16C74).

The *9-Bit µLAN* is:

- Fast-A high speed (62.5k baud) multidrop master/slave RS-485 network
- Flexible-compatible with your microcontrollers
- Reliable-robust IS-bit CRC and sequence number error checking
- Efficient-low microcontroller resource requirements (uses your chip's built-in serial port)
- Friendly-Simple to use C and assembly language software libraries, with demonstration programs
- ***Complete***—includes network software, network monitor and RS-485 hardware
- Practical- applications include data acquisition and distributed control

Microcontroller, PC Master, or Laptop
Intel 8051 HVAC/R Lighting Control
Intel 80C186EB/EC Point-of-Sale
Motorola 68HC11 Process Control Transportation
Microchip PIC16C74 Fire Detection Security

9-Bit RS-485 Multidrop Network

Cimetrics TECHNOLOGY

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

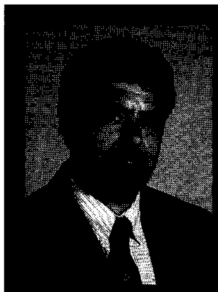
#124

EMBEDDED TECHNIQUES

John Dybowski

Power Management with the DS87C530

Part 2: The Software



John wraps up his column with a focus on the software and firmware issues of DS87C530 applications. Using a data-logging example, he shows how to take advantage of its power-management capabilities.

Oast month's column focused on the hardware aspects of a general-purpose computer with inherent soft power-management capabilities. My discussion centered around Dallas Semiconductor's DS87C530 and DS87C520 processors.

Looking at modern power-management techniques from the perspective of a microprocessor shouldn't be surprising since this is the source of most, if not all, dynamic signaling in systems. In a CMOS system, remember any significant power consumption results from dynamic dissipation. In a properly implemented design, quiescent current flows at leakage levels amounting to microamps.

If we agree that system-wide power management is an extension of power control at the processor level, it should be apparent why I am so enthusiastic about Dallas's accomplishments. With the interest of keeping power consumption in check, I find it surprising that other controllers don't even come close in terms of power-management capabilities.

A cursory survey of the competing devices reveals that nothing significant specifically deals with the power issues plaguing many embedded applications. Even some of the newest embedded processors seem hard pressed to offer even incremental improvements over last-generation power-control methodologies.

I suppose it all boils down to a semiconductor manufacturer's perspective and background. Dallas has expended considerable effort developing and perfecting circuits specifically targeted at administering system-critical functions. Their expertise with nonvolatile memories, nano-powered real-time clocks, and a variety of microprocessor and memory-supervisory circuits has given them an appreciation of the problems of effective embedded design on a global scale. This experience culminates in the multiplicity of ancillary features they have built into their high-speed processor line.

Power management in embedded systems is one of those things you either absolutely need or just don't care about. To those experienced in the field, the material I'm about to present should uncover few surprises. Those unfamiliar with the discipline may gain new insight and develop an appreciation for the realm where a few microamps can spell the difference between design success and market failure.

In a fully CMOS system, power management is synonymous with clock management. What we are trying to control is dynamic dissipation. Quiescent dissipation should take care of itself as long as a few basic guidelines are followed. Of course, these criteria assume the system is composed entirely of micropower components. Obviously, selecting the right peripherals can make or break it.

The DS87C530 supports the traditional 80C3 1 Idle mode and Stop mode. Additionally, you can select an alternate clock source (e.g., a ring oscillator) and invoke reduced throughput modes such as PMM1 and PMM2 (power management modes 1 and 2). Many features can be combined to yield radical power savings at degraded performance, moderate power consumption with somewhat compromised performance, or any nuance between these extremes.

Listing 1—*LOG.ASM demonstrates burst mode operation in a data logger device. The code was written specifically for the DS87C530 running at 33 MHz.*

```
* DS87C530 register equate table
PMR      EQU    $0C4               Power Management Register
STATUS  EQU     $0C5               Status Register
RTASS    EQU    $0F2               Real-time Alarm SubSecond Register
RTAS     EQU    $0F3               Real-time Alarm Second Register
RTAM     EQU    $0F4               Real-time Alarm Minute Register
RTCC     EQU    $0F9               Real-time Clock Control

* Bit equate table
RIO      BIT    898                Serial port 0 receiver int flag
TIO      BIT    $99                Serial port 0 transmitter int flag
EX1      BIT    $0AA               External interrupt 1 enable bit
FO       BIT    $0D5               General-purpose flag

* LTC1392 I/O
?clk     BIT    p1.7               Clock bit
?dq      BIT    p1.6               Bidirectional data
?cs      BIT    P1.5               Chip select

* LT1180 control
EN232    BIT    p3.6               LT1180 enable control

* SIO completion flag
SENT     BIT    FO

* Baud rate constant for 9600 bps using timer 2 at 33 MHz
B9600    EQU    65536-107

* External RAM
BUFF     EQU    0                  Data collection buffer

* Interrupt vectors
         ORG    0                  Reset vector
         LJMP   START

         ORG    $13                External interrupt 1 vector
         LJMP   EXT_INT1

         ORG    $23                Serial interrupt 0 vector
         LJMP   SER_INT0

         ORG    $6B                Real-time clock interrupt vector
         LJMP   RTC_INT

         ORG    $100

* System initialization
START
         MOV    SP,#$80            Set stack pointer
         MOV    P1,#$FF
         MOV    P3,#$FF
         CLR    EN232              Disable LT1180
         MOV    PMR,#1             Enable on-chip RAM

* Set up alarm for hourly wakeup call
         MOV    RTAM,#0            Minute alarm
         MOV    RTAS,#0            Second alarm
         MOV    RTASS,#0           Subsecond alarm
         MOV    TA,#$0AA           Enable timed access
         MOV    TA,#$55
         ORL    RTCC,#$0C1         Enable min, s, and sub s compares

* Set up timer 2 and serial port 0
         MOV    T2CON,#30H         Auto-reload timer off
         MOV    RCAP2L,#B9600      Low baud reload
```

(continued)

Also, you should consider that full-speed operation is the appropriate choice when the increased bandwidth can be used fully. Running at 8 MIPS, the DS87C530 can accomplish a lot of work in a short period of time.

CHOICES

Choosing the appropriate power-management regime requires a thorough analysis of the system's operating conditions and is totally application dependent. The subject involves not only selecting the right power-management methodology, but also:

- how to handle high-priority tasks in degraded operational modes
- how to deal with timing interdependencies across mode changes
- when to switch modes
- which modes are most effective in a given situation

Needless to say, a dynamic power-management scheme can get fiendishly complicated when taken to extremes-which is exactly what many portable applications have come to demand.

Two basic classes of system operation can be defined, although most real-world implementations blur the distinction. The first category includes systems that must remain fully responsive to external events (e.g., asynchronous communications). These systems can be served by operating the processor (at periods of reduced throughput) in PMM1 or PMM2. Here, the clock source usually remains the crystal oscillator for maximum accuracy.

Unattended systems performing relatively infrequent I/O at specific intervals are ideal candidates for burst-mode operation. Here, the system spends most if its time in standby, emerging only for brief periods to perform a periodic operation before reentering the quiescent standby mode. The processor spends most of its time hibernating in Stop mode. This mode of operation is the essence of a data-logger application.

Rather than just touch on the many available power-reduction tricks a DS87C530-based system is capable of

Listing I-continued

```
          MOV     RCAP2H,#=B9600    High baud reload
          MOV     TL2,#B9600        Low baud multiplier
          MOV     TH2,#=B9600       High baud multiplier
          MOV     SCON,#%01010010 Mode 1, REN. TXRDY, RXEMPTY
          SETB    T2CON.2           Enable baud timer

* Set up interrupts
          MOV     IP,#$10           Serial port 0: high priority
          MOV     EIP,#$20          RTC: high priority
          MOV     EIE,#$20          Enable RTC interupt
          MOV     IE,#$94           Enable SIO 0 and EXT1 interrupts

**************************************************************
* MAIN waits for interrupts. When they are complete. it
* switches back to the ring oscillator and puts the
* part back into stop mode.
**************************************************************
MAIN
          CLR     EN232             Disable LT1180
          ANL     EXIF,#$0F7        Switch to ring oscillator
          ORL     EXIF,#2           Enable restart from ring
          ORL     PMR,#8            Disable crystal
          ORL     PCON,#2           Enter stop mode
          SJMP    MAIN

**************************************************************
* RTC_INT reads a value on the LTC1392 every 30 minutes, and
* logs data to the data buffer. When done, it returns to the
```

(continued)

performing, I'll concentrate on a burst-mode data-logging application. Although the application runs and performs something useful, keep in mind that I'm keeping it extremely simple in order not to obscure the power-management theme.

A DIFFERENT DATA LOGGER

A traditional data-logger architecture requires multiple switched power supplies, external event-monitoring circuitry, and discrete power-control logic. In burst-mode operation, the processor emerges from a fully powered-off state using dedicated power-control circuitry. This wakeup is usually controlled by some form of periodic interval timer. Although highly effective, this approach unavoidably increases component count and cost. The cost penalty precludes this type of instrument for general-purpose applications.

In contrast, the system I'm presenting attains equivalent burst-mode capability using only the DS87C530's inherent power-management features.

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

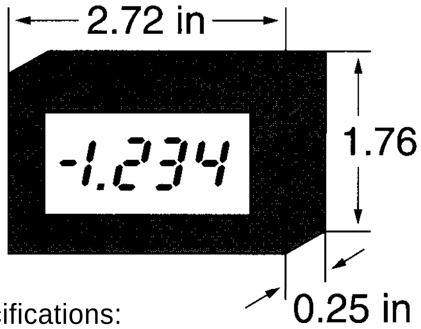
- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 805 1/803 1 or 8052/8032 CPUs*
- *Compatible with any RAM or ROM memory mapping*
- *Runs up to 50 times faster than the MCS BASIC-52 interpreter.*
- *Includes Binary Technology's SXA5 1 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51 $ 295.***

508-369-9556
FAX 508-369-9549

Binary Technology, Inc.
P.O. Box 541 . Carlisle, MA 0 1741

#126

3½-DIGIT LCD PANEL METER
-Available now at an unheard of price of **$15** plus s & h
New! Not surplus!



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

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

The trick to keeping power consumption down is to build a system entirely of micropower circuits.

To keep things simple, I added only one part to the basic DS87C530 system I presented last month. The addition of Linear Technology's LTC-1392 thermometer, differential A/D converter, and voltage monitor IC provides the front end of a data-collection system in an 8-pin circuit.

Fully micropower, the LTC1392 typically requires 200 nA of current in standby mode and 300 µA while operating. The processor interface via a synchronous serial link uses transmit and receive data lines, a clock, and chip select. Since communications are defined as half duplex, the data lines can be tied together resulting in a three-wire interface.

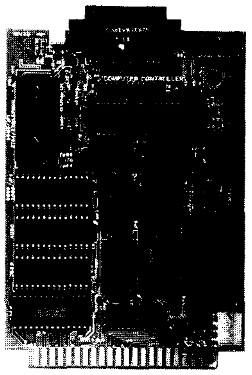
The data format is compatible with the Microwire and SPI interfaces. Low power consumption combined with serial operation offers the possibility of locating this device remotely from a processing unit if necessary. In many applications, it's advantageous

Listing l-continued

```
*main loop where it enters stop mode again. This simple
* example assumes that the system is read before
* the data overflows RAM, so no error checking is included.
***************************X***********************
RTC_INT
        ANL    RTCC,#$0FD        Clear rtc interrupt flag
        PUSH   ACC
        PUSH   DPS               Save data pointer selector

        MOV    A,#0              Request temperature conversion
        LCALL  CONVERT           Acquire LTC1392 data

        MOV    DPS,#0            Switch to data pointer 0
        XCH    A,B               Get msb
        MOVX   @DPTR,A           Store to data buffer
        INC    DPTR              Point to next location
        MOV    A,B               Get lsb
        MOVX   @DPTR,A           Store to data buffer
        INC    DPTR              Point to next location
        XRL    RTAM,#$1E         Next interrupt in 30 minutes
        POP    DPS               Restore data pointer selector
        POP    ACC
        RET1

***************************************************************
* EXT_INT1 causes the processor to download all stored data.
* It switches to the crystal first. The SENT bit indicates
* when all the data has been sent by the SIO ISR. This
```

(continued)

BCC52
BASIC-52 COMPUTER/CONTROLLER



The BCC52 controller continues to be Micromint's best selling single-board computer. Its cost-effective architecture needs only a power supply and terminal to become a complete development system or single-board solution in an end-use system. The BCC52 is programmable in BASIC-52, (a fast, full floating point interpreted BASIC), or assembly language.

The BCC52 contains five RAM/ROM sockets, an "intelligent" 2764/128 EPROM programmer, three 8-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.

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

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

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

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

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

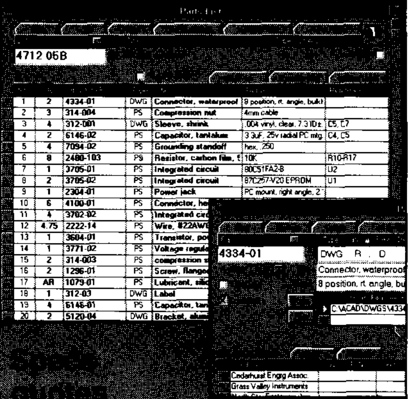
CALL FOR OEM PRICING

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

This Parts List Software is for Engineers / Designers

Not MRP: P&V actually assists development!

Windows™ program helps you stay organized



4712 06B

1	2	4334-01	DWG	Connector, waterproof	8 position, rt. angle, bulkh	
2	3	314-004	PS	Compression nut	4mm cable	
3	4	312-001	DWG	Sleeve, shrink	.004 vinyl, clear, 7.3 ID±	C5, C7
4	2	6146-02	PS	Capacitor, tantalum	3.3uF, 25v radial PC mtg	C4, C5
5	4	7094-02	PS	Grounding standoff	hex, .250	
6	8	2480-103	PS	Resistor, carbon film, 5	10K	R10-R17
7	1	3705-01	PS	Integrated circuit	80C51FA2-B	U2
8	2	3705-02	PS	Integrated circuit	87C257-V20 EPROM	U1
9	1	2304-01	PS	Power jack	PC mount, right angle, 2	
10	6	4100-01	PS	Connector, he		
11	4	3702-02	PS	Integrated circ		
12	4.75	2222-14	PS	Wire, #22AWG		
13	1	3604-01	PS	Transistor, po		
14	1	3771-02	PS	Voltage regula		
15	2	314-003	PS	compression s		
16	2	1296-01	PS	Screw, flange		
17	AR	1079-01	PS	Lubricant, sili		
18	1	312-03	DWG	Label		
19	4	6146-01	PS	Capacitor, tan		
20	2	5120-04	DWG	Bracket, alum		

4334-01 DWG R . D
Connector, waterproof
8 position, rt. angle, bu
C:\ACAD\DWGS\4334
Cedarhurst Engrg Assoc.
Grass Valley Instruments

Keep track of:
- Drawings & Specs
- Suppliers & quotes
- Product material costs
- Engineering stock

Parts&Vendor$™

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

$99 + shpg **Call 800.280.5176**

Requires 486, 8meg mtn. RAM, Windows™

Trilogy DESIGN™
voice 916.273.1985
fax 916.477.9106
P.O. Box 2270, Grass Valley, CA 95945

#119

Listing 1-continued

```
* prevents the software from exiting the and reentering stop
* mode before all the data has been transmitted.
*************************************************************
EXT_INT1
        SETB   EN232              Enable LT1180
        ANL    PMR,#$0F7          Enable crystal for serial activity
EI1_1
        MOV    A,STATUS           Wait until crystal has stabilized
        JNB    ACC.4,EI1_1
        ORL    EXIF,#8            Switch to the crystal

        CLR    SENT               Clear completion flag
        SETB   TIO                Data will be sent by SIO ISR
        JNB    SENT,*             Loop until all sent

        MOV    DPS,#1             Switch to dptrl and reset xmit ptr
        MOV    DPTR,#BUFF
        MOV    DPS ,#0            Switch back to dptr0 to log data
        RET1

*************************************************************
* SER_INT0 handles serial port 0 interrupts. Serial port
* interrupts are only possible when the system is "active"
* following the assertion of the \INT1 bit. The primary
* function of this interrupt is to transmit the next
* character in the table until all data has been sent.
*************************************************************
SER_INT0
        JB     TI0,XI0_1          Determine interrupt source

* Receive interrupt
RINTO
        CLR    RIO                Clear receive interrupt flag

        RET1

* Transmit interrupt
XI0_1
        CLR    TIO                Clear transmit interrupt flag
        MOV    A,DPL1             Check for end of data
        CJNE   A,DPL,XI0_2
        MOV    A,DPH1
        CJNE   A,DPH,XI0_2

        SETB   SENT               Indicate completion
        RET1

x10-2
        PUSH   DPS                Preserve current data pointer
        MOV    DPS,#1             Switch to dptrl to track data ptr
        MOVX   A,@DPTR            Still have data, xmit it, restore
        MOV    SBUF0,A            Data ptr, return to send next byte
        POP    DPS
        RET1

*************************************************************
* CONVERT data acquired using the LTC1392. The conversion
* command word is input in the accumulator. The two LSBs are
* significant:
*       OOB Temperature conversion.
*       01B Power supply measurement.
*       10B Differential, 1-V full scale.
*       11B Differential, 0.5-V full scale.
*************************************************************
CONVERT
* First, enable the LTC1392
        SETB   ?clk               Initial clock state
```

(continued,

to put the sensing circuitry where the action is.

Another front-end candidate worth mentioning is Maxim's MAX-186. This 8-channel (4 if operated differentially), 12-bit A/D converter features a built-in 4.096-V reference and has a Microwire/SPI-compatible interface. Designed for very low-power applications, the MAX186 has several power-saving modes. Full device shutdown can be invoked using either an external (\SHDN) pin or a command from the processor. In this mode, power consumption drops to about 2 µA.

This approach illustrates the virtue of incorporating all converter functions onto a single chip. Even a micropower band-gap reference can be a significant source of power consumption considering that the MAX-186 in fast shutdown mode (reference enabled) has a power consumption of 30 µA.

For the most part, this value represents the bandgap's consumption since all other device parameters are identical to full shutdown in this mode, except the bandgap is not disabled. When it becomes active, the MAX186 is still no slouch, coming in at just 1.5 mA.

You'd be hard pressed to find an A/D converter that pulls so many desirable features together. For applications needing an external reference or for ratiometric applications, the MAX188 provides identical features without the built-in reference.

OPERATIONAL OVERVIEW

My data-logging application operates under battery power, spending most if its time in standby (Stop) mode. This strategy brings power consumption down to minuscule levels. However, since the processor oscillator is shut down, some external method is needed to bring the system back online. Because the DS87C530's built-in, real-time clock runs from an independent crystal, this "external" wakeup is integrated right into the processor.

The real-time clock periodically generates an alarm interrupt which takes the processor out of standby

$149 4A C Compiler?

You heard right! A quality K&R C compiler designed for the 8051 microcontroller family, just $149, **including** the Intel compatible assembler and linker. A source level simulator is also available for just $169. A great companion to our fine Single Board Computers, like those below. **CALL NOW!**

552SBC

80C552 a '51 Compatible Micro
40 Bits of Digital I/O
8 Channels of **10** Bit A/D
3 Serial Ports (RS-232 or 422/485)
2 Pulse Width Modulation Outputs
6 Capture/Compare Inputs
1 Real Time Clock
64K bytes Static RAM
1+ UVPROM Socket
512 bytes of Serial EEPROM
1 Watchdog
1 Power Fail Interrupt
1 On-Board Power Regulation
1 Expansion Bus

Priced at just **$229** in 100 piece quantities. Call about our 552SBC C Development Kit, just $499 qty 1.

Hyperactive '51!

Our popular 8031SBC can now be shipped with Dallas Semi's hyperactive **DS80C320**, an 8051 on steroids. Averaging 3x faster than the standard 51, your project can really scream! **Call or ftp for pricing and brochures today!**

Other versions of the 8031SBC have processors with on-chip capture registers, EEPROM, IIC, A/D and more. Call or ftp for a list!

8031SBC as low as $49

Call for your custom product needs. Quick Response.

HTE
HiTech Equipment Corp.
9400 Activity Road
San Diego, CA 92126
[Fax: (619) 530-1458]
Since 1983
(619) 566-1892

VISA MasterCard

Internet e-mail: info@hte.com
Internet ftp: ftp.hte.com

#127

Listing 1—*continued*

```
            MOV     R1,#4
            DJNZ    R1,*
            CLR     ?cs                     Assert device select

*  Count appropriate delay
            ANL     A,#%11
            JNZ     ?1

            MOV     R0,#30
            DJNZ    R0,*                    Delay 80 µs for temp conversion
?1
            MOV     R0,#3
            DJNZ    R0,*                    Delay 10 µs for other conversions

*  Format command word
            MOV     C,ACC.0
            MOV     ACC.2,C                 Position argument bits
            SETB    ACC.0                   Start bit
            SETB    ACC.3                   MSB first mode
            SETB    ACC.4                   Line float for null bit receive

*  Transfer command to LTC1392 and accept null bit
            MOV     R0,#5                   4 xmit bits, 1 null receive bit
?2
            RRC     A
            MOV     ?dq,C                   Emit data bit
            CLR     ?clk
            SETB    ?clk
            DJNZ    R0,?2

*  Read 2 MSBs
            MOV     R0,#2                   2 bits to receive
?3
            CLR     ?clk
            MOV     C,?dq                   Input data bit
            RLC     A
            SETB    ?clk
            DJNZ    R0,?3
            MOV     B,A

*  Read 8 LSBs now
            MOV     R0,#8                   8 bits to receive
?4
            CLR     ?clk
            MOV     C,?dq                   Input data bit
            RLC     A
            SETB    ?clk
            DJNZ    R0,?4

            SETB    ?cs                     Release device select

            XCH     A,B
            ANL     A,#%11                  Only 2 LSBs in MSB are significant
            XCH     A,B

            RET
```

mode. Once operational, the system takes a temperature reading, stores the value to RAM, and concludes by reentering standby mode. A crude data feature can be invoked on demand by manually asserting an external interrupt pin.

SOFT MACHINE

The program in Listing 1 is totally dedicated to this simple data-collection task. As such, it adequately illustrates a purely software-based approach to power management. Admittedly, it is too simplistic for any real application. For your reference, Table 1 shows some of the clock control and status bits relevant to the program description. Let me begin by describing functions of the various program blocks.

The program starts by defining a number of processor and application resources. Since I am using Dunfield's generic DS80C320 assembler, I first inform it of the new DS87C530 special function registers (SFRs). Subsequently, the application-specific I/O bits and variables are defined.

Following the reset and (expanded) interrupt-vector block, the executable program begins setting up the stack pointer, initializing processor I/O, and enabling the 1 -KB embedded RAM. The LT1180 RS-232 transceiver chip is disabled to conserve power since it is only needed for communications.

Now, the real-time clock is set up for use as a periodic interrupt source. The RTC alarm-compare registers— RTAM (minutes), RTAS (seconds), and RTSS (subseconds)-are loaded to generate an interrupt on the hour. Timed access is required to enable the RTC alarm capability through RTCC (real-time clock control). The previously initialized alarm compare registers are set up to participate in the alarm compare. Note that RTAH (hours) is disregarded since it would put the alarm interval beyond the bounds of an hour.

Prior to entry into the main loop, serial port 0 is initialized to run using timer 2 as the baud clock. Interrupt priority is given to the SIOO and RTC so they can interrupt EXT1. Interrupts are unmasked through the standard Interrupt Enable and Extended Interrupt Enable (EIE) SFRs.

The main program loop switches the clock source to the ring oscillator and allows restart from Stop via the ring oscillator through EXIF. On interrupt, the processor immediately begins executing under control of the ring oscillator before the crystal oscillator even starts up.

The crystal oscillator is disabled using PMR, and Stop mode is invoked using the familiar PCON SFR. The system is now totally inoperative until an interrupt occurs. As the ultimate interrupt-driven system, it is quite dead for the moment.

The system can now be enabled via the real-time-clock interrupt or through external interrupt 1. Normal logging operations are handled in response to an RTC interrupt. On seizing control of the processor, the RTC ISR proceeds by first clearing the RTC interrupt flag. After pushing the accumulator and data pointer select register, the LTC1392 is accessed via its serial support routine. When conversion is complete, the appropriate data pointer is selected and the 2-byte (10 significant bits) temperature conversion is stored.

Unmodified, the RTC causes the next interrupt to occur on the hour since a compare of RTAM (alarm minutes) registers a match with RTCM (real-time minutes) once an hour. However, in this case, the desired sample interval is 30 minutes. To accomplish this, setting RTAM to 30 produces the desired result since minutes are the qualifying parameter.

The simplest way to program a recurrent, 30-minute interrupt is to set RTAM to 1Eh (the RTC counts in binary) when it is 0 and back to 0 when it is 1Eh. Setting the interrupt can be accomplished with the single instruction `XRL RTAM,$1E`.

The ISR concludes by restoring the pushed registers and returns to the main loop, which places the system back into Stop mode. The ring oscillator provides the clock source for the brief duration of the data-collection sequence. The process concludes before the crystal oscillator even has time to warm up.

Asserting external interrupt 1 invokes the data-dump function. The EXT1 ISR begins by enabling the LT1180 RS-232 transceiver. This enabling is done early to allow the chip's charge pump to ramp up the positive and negative RS-232 rails. Immediately following the charging, the crystal amplifier is enabled since serial communications are not possible while using the ring oscillator. Further processing is suspended until the STATUS SFR indicates that the crystal oscillator has stabilized and is avail-

BIG THINGS COME

DOMINO™

Microcontroller

Starting at $79

Micromint's Domino-52 microcontroller is a "supercomputer" in less than 0.75 cubic inches. We've packed the most essential elements into one tiny package. Domino is a plug-and-go module, just attach +5V and a terminal or network. A simple keyed sequence saves an autostarting program in nonvolatile memory.

SPECIAL FEATURES

- 80C52 with ROM-resident, full floating-point BASIC
- 32K bytes SRAM and 32K bytes EEPROM
- Two PWM outputs, I^2C bus
- Serial I/O: (up to 19,200 bps) RS-422, RS-485 & RS-232A
- Two interrupts and three timers
- Parallel I/O: 12 bits, 3 shared with ADC and I^2C
- Power: +5V @ 15 mA; Size: 1.75"×1.062"×0.4" potted
- A/D converter: 2 channels, 12 bits, 10k samples/sec.
- Connections: via 2×10, 0.1" dual-row header
- -20°C to 75°C operating temperature
- Industrial temperature available

MICRO MINT

4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(203) 871-6170 • Fax: (203) 872-2204

#119

Bit name	Location	Function	Reset	Write access
XT/RG	EXIF.3	Crystal-Ring Clock Source Select 0 = Select ring oscillator as clock source 1= Select crystal or external clock as clock source	1	0 anytime; 1 when XTUP = 1 and XTOFF = 0
RGMD	EXIF.2	Ring/Oscillator Mode Status 0 = Crystal or external clock is current clock source 1 = Ring oscillator is current clock source	0	None
RGSL	EXIF.I	Ring Oscillator Select, Stop Mode 0 = Crystal or external clock will be the clock source when resuming from Stop mode 1 = Ring oscillator will be the clock source when resuming from Stop mode Note: Upon completion of crystal warm up period, device will switch to clock source designated by XT/RG bit	Unchanged except after power-on reset, when it is cleared to 0	Unrestricted
XTOFF	PMR.3	Crystal Oscillator Disable 0 = Crystal oscillator is enabled 1 = Crystal oscillator is disabled. Device is operating from ring oscillator	0	0 anytime; 1 when XT/% = 0
XTUP	STATUS.4	Crystal Oscillator Warm Up Status 0 = Oscillator warm up still in progress 1 = Oscillator warm up complete	1	None

Table **1-A look at the** *DS87C530's* **new** *SFRs* **reveals power** *management* **is** *built* **info the fundamental** *architecture.*

able as the system's primary clock source.

At this time, the code falls through and switches from the ring oscillator to the more accurate crystal time base. Directly setting serial port O's TIO interrupt flag transfers control to the SIOO ISR. The code now stalls while continually polling the Sent bit which functions as the SIOO completion flag while the SIO ISR transmits the stored data. It's important not to allow the EXT1 service routine to terminate until the interrupt-driven SIO activity completes since a return to the main loop immediately takes the system out of service.

On completion of the communications activity, the program drops through and initializes the storage pointer, effectively purging all stored data in preparation for a new collection cycle.

AND THAT'S NOT ALL

Effective power management dictates fitting the right power-control techniques to a given application's specific requirements. The burst-mode data collection I described is arguably the simplest of all such applications since it typically involves no real-time processing whatsoever.

Things can and do get much more complicated when you have to cut power consumption and keep up with real-world events at the same time. Here, the wrong processor could easily leave you out of options and out of luck.

However, the DS87C530, known primarily for its high throughput, won't let you down when it comes time to reduce the electric bill. ❑

John Dybowski is an engineer involved in the design and manufacture of embedded controllers and communications equipment with a special focus on portable and battery-operated instruments..

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" for downloading and ordering information.

I R S

431 Very Useful
432 Moderately Useful
433 Not Useful

IN SMALL PACKAGES

BLACKJACK™ TELECONTROLLER

Starting at $139

Micromint's BlackJack-552 is a true "telecontroller" incorporating both microcomputer and FCC-certified Xecom modem in a single package. BlackJack comes preprogrammed with firmware utilities which are optimized for assembly language, C, and BASIC programs. Like Domino, BlackJack is easy to use. Attach power and a terminal, then upload, store, and execute your program.

SPECIAL FEATURES
- 80C552 processor with firmware monitor, 14.7456 MHz
- Serial I/O; (up to 38,400 bps); full-duplex TTL, I^{2}C bus
- Connection: 1.2" dual-row 48-pin DIP header
- Parallel I/O: 20 bits, 4 bits shared with RTC
- Two interrupts: three timers, watchdog timer, two PWM outputs
- Hardware real-time clock
- A/D converter; 8 channels, 10 bits, 20k samples/sec.
- 2400 bps data modem (data/fax available)
- Power: +5V @ 55 mA; Size: 2.75"×1.375"×0.5" potted
- 32K bytes SRAM, 32K bytes ROM and 128K bytes EEPROM
- Industrial temperature available

MICRO MINT

4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(203) 871-6170 • Fax: (203) 872-2204

#119

CONNECTIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) 871-l 988-Four incoming lines
Internet E-mail: sysop@circellar.com

Based on feedback I've received from readers, we have quite a few ham radio operators out there (including our own Ed Nisley, Russ Reiss, Harv Weiner, and John Gorsky). This month, we start off with a topic that may be familiar to many of them: using PIN diodes for doing RF switching.

Next, we've had a number of articles on motor speed control, but here we look at dealing with higher voltages and AC versus DC. If's not as scary as you might think.

Finally, we fake a quick look at the keyboard lock found on almost all of today's IBM PC-compatible machines. Is if really a true hardware-based lock?

PIN diodes as Tx/Rx switch

Msg#: 5906
From: David Gwillim To: All Users

I have recently seen advertisements in ham radio magazines for equipment using PIN diodes as transmit/receive switches in linear amplifiers and finals for transceivers. The PIN diode circuit apparently replaces a mechanical relay to switch in and out the receiver connection to the antenna. Does anyone have any circuits that perform this function and any explanation of how they work?

I would also be interested in supply houses that sell parts (especially a line of PIN diodes that will function in this type of circuit). All I know at present is that the acronym PIN stands for Positive-Indium-Negative, referring to the kind of junction in the diode, but I have no practical (or theoretical for that matter!) experience with the device.

I was wanting to construct a Tx/Rx relay using a PIN diode for use in a ham band (1.8–30.0 MHz) linear amplifier operating at above 180 watts RF output.

The idea is quite simple, David. The diode exhibits a relatively high impedance to an AC signal (if its voltage is sufficiently below its turn-on voltage). When the diode [illegible] is forward biased, it looks like a relatively *low* impedance to the AC component. This property can be used to form a Tx/Rx switch, as you mention.

You can do this with any diode, though PIN diodes perform better (don't ask me why). PINs used to be available from HP and others. I built a 2-m switch as you describe and operated it at about 100 W using nothing more than 1N4148 switching diodes, as I recall! The nice thing about VHF/UHF, though, is that you can use quarter-wave lines as impedance transformers, which allows you to place the diodes at the optimum voltage/current peak/valley.

Often, more than one diode is used, some in series, some in parallel. Some maybe driven on while the others are driven off. With quarter-wave transformers, you can create either an open or a short condition, as you wish.

I suggest you check the ***Radio Amateur's Handbook*** from ARRL (Newington, CT) first. I'd guess they must have some of these circuits. Problem is, the contents change yearly, so it depends a great deal on which issue you happen to have access to. Also, there must be ***tons*** of references and practical designs of these switches in QST, CQ, 73, and ***Ham Radio*** (now defunct) magazines if you have access to a library for back issues.

Msg#: 5955
From: David Gwillim To: Russ Reiss

Thanks for some insight into the theory. It makes sense that that is how they operate. In all the years I did hardware repair, though, I always found it was a long way from a theoretical understanding of something to having a practical, efficient working circuit that doesn't produce any nasty, unwanted side effects! I will research the subject at the library if I can get some back issues of ham mags there.

Msg#: 6114
From: Pellervo Kaskinen To: David Gwillim

I may not have the latest catalogs on the areas of little or no interest to my job. Microwave is one of those areas. And microwave catalogs are the ones I would need to take a look at for your PIN-diode-based Tx/Rx switch question.

The best I have available is an HP catalog from 1990. There, the highest power offered is 120 W. Consequently, then, it looks like just a couple of those diodes might do the 180 W you are asking for. But wait a moment!

The frequency range does not meet your requirements. Not at the low end. The stated minimum frequencies for most of the devices are either 30 or 50 MHz. The upper end goes typically to 10,000 MHz. Maybe you should move up to the 2-meter band?

CONNECTIME

Just kidding-like I said, I have not followed the trends on this arena. Maybe newer devices really have improved more than what is evident in my data sources.

I have another reason for my reply. PIN diodes are also used in the optoelectronics (light waves). They are used in fiber-optic signal receivers due to their fast response, low noise, and excellent stability, All this is due to the "I" layer that reduces the leakage current as well as the capacitance.

Ordinary PN junction devices with otherwise similar geometries have a much higher capacitance unless they are operated at very high reverse biases. And then they leak too much, which means they are noisy. If the capacitance is high, then the response speed is low. Basically, the same issue you are dealing with in switch applications.

Msg#: 6157
From: David Gwillim To: Pellervo Kaskinen

Thanks for your input. Is there anything you don't involve yourself with? I see a lot of answers to people's questions coming from you. What business are you in?

Msg#: 6162
From: Pellervo Kaskinen To: David Gwillim

I'm an electrical engineer, as in one trained for power transmission and use of electricity. But all my professional life I have been involved with electronics, control systems, and measurements. It all started in manufacturing cables, or actually making machinery for the cable manufacturing.

Next place I got into was a paper mill (of the same company that owned the cable manufacturing). Besides a paper mill, they had a pulp mill, a saw mill, a rubber products factory, and plastics factory on more or the less same premises. Actually, we had a joint R&D center for the three branches and I supported the measurement activities for all.

Then I got myself into a real adventure-it was as an inspector abroad for a Finnish engineering company to inspect the quality of installations for a paper mill, pulp mill, plywood mill, wood handling, steam power plant complex. We had at peak 17 inspectors. The job lasted for 9 months.

After that I came to the U.S. and started learning arc welding. I still am trying to learn some of that after 16 years. Actually, I'm designing control circuits for welders and accessories.

Msg#: 5926
From: Chuck Olson To: David Gwillim

I thought PIN stood for P-doped, Intrinsic, N-doped. I don't have any first-hand experience with PINs, but I understand they will act as RF resistors with forward DC bias— more current, less resistance.

There was an article in the December 1994 QST which compared PIN and PN diodes for distortion characteristics as switches for transceiver front-end filters. One of the diodes tested was the 1N40071-kV rectifier which has a PIN structure. The 1N4007 might be your cheapest, easiest to get part. It should also be hard to kill.

I assume you want the PIN switch for full break-in. I'd get hold of an ARRL handbook. My 1987 issue has plans for a PIN Tx/Rx switch for a tube final transmitter. You could probably modify the design for a solid-state final.

Personally, I always used a separate antenna or threw a switch manually, but I'm not a CW demon!

Msg#: 5954
From: David Gwillim To: Chuck Olson

I am just getting into ham radio and don't yet have my license (am taking the General exam soon). I had a lot of interest in ham back in the early '70s but was at sea then and couldn't easily get or use a ticket. My theory is pretty good; just have to get my ear used to the dit-dahs again.

While I am getting ready for the license exam, I am purchasing some ham gear. I bought an Index Labs QRP Plus since small, neat packages really appeal to me, but I wanted to hedge my bet with a few more watts of power. So I bought a CCI linear parts kit for a 5-140-W, 1.830-MHz unit (their AN762). It doesn't have a Tx/Rx relay, hence my interest in PIN.

You are probably right about the Intrinsic instead on Indium-it's been a while since I did electronics seriously (now have my own computer consulting business and do financial software maintenance for a large bank).

Msg#: 5979
From: Chuck Olson To: David Gwillim

Good luck on the test. I got my General about 20 years ago. I still remember those "sweaty palms" at the Federal building in Milwaukee.

I'd like to hear how your CCI amp works out. I've always meant to put some shoes on my Ten-Tee Argonaut. Five watts is fun, but sometimes it's nice to have the extra power.

I've never heard of Index Labs. Is the QRP Plus a kit?

Msg#: 5992
From: David Gwillim To: Chuck Olson

Index Labs is a small outfit in Washington state run by a ham called Bruce Franklin. I get the feeling he has been in the equipment design business a long time since the innards of the QRP Plus are beautiful to behold! It isn't a kit, it's a finished digital-synthesis QRP rig with full break-in CW and SSB covering all HF ham bands. It has a built-in iambic keyer, too.

Index Labs advertises in CQ magazine and they had a review of the rig in the October or November 1994 issue

CONNECTIME

(can't remember which). The little rig is really compact (maybe 5" on a side in a cube) and pulls about 120 mA on receive and about 1.5 A on transmit for 5-W output.

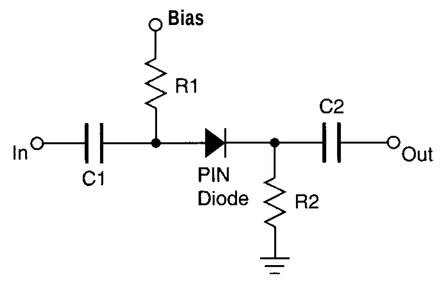
Since I don't have my ticket yet, I had a friend try it out for me and he was really impressed. A nice feature is that Bruce built in a super-narrow CPU-controlled SCAF (switched-capacitor audio filter) that gives you a brick-wall bandwidth of anywhere from 100 Hz to 2400 Hz. You can really peel apart CW with that.

If you are interested in seeing what makes the QRP Plus tick, give Bruce a call at (206) 851-5725 and he'll send you an owner's manual with circuit description and schematics. You have to see the construction quality to believe it, however.

Msg#: 5941
From: James Meyer To: David Gwillim

PIN stands for Positive, Intrinsic, Negative. The P and N are exactly the same as the P and N used in NPN and PNP when you refer to transistors. Intrinsic refers to a semiconductor without any doping that would otherwise have made it into a P-type or N-type.

Here's a schematic of a PIN diode switch:



The RF flows from In to Out. A bias voltage is applied so that when the bias is positive, the diode is forward biased and RF flows from In to Out. When the bias is negative, the diode is reverse biased and the RF is cut off.

The bias voltage needs to be larger than the applied RF so that the diode stays either forward or reverse biased.

Switching lots of power isn't very practical with PIN diode switches. The resistors will eat up quite a bit of the RF power. PIN diode switches are used at low power.

Simple, ain't it?

Msg#: 5956
From: David Gwillim To: James Meyer

Thanks for the acronymic correction and the schematic. It looks as though you need some added protection circuitry in case the diode fails or there is a high enough VSWR to turn it back on despite the bias (unless I am missing something). I am curious if you know why a PIN diode is so special-maybe it's got something to do with its reverse recovery speed or inter-electrode capacitance.

Msg#: 6011
From: James Meyer To: David Gwillim

My reply was purposely simplified to illustrate just the basic idea of a diode switch. A real, useful switch would require more parts depending on its application. For instance, PIN diodes are often used as very high speed photodetectors as well as RF switches.

A PIN diode is special because it's got very low leakage in the reverse-biased condition along with low capacitance. Some power rectifiers are made with a PIN structure. As far as reverse recovery goes, PIN diodes are slower than equally sized ordinary signal diodes. An HP PIN diode for RF use will have a reverse recovery of around 100 ns as compared to a similar-sized signal diode's 5-ns reverse recovery.

For RF use, one of the things that makes PIN diodes useful is their characteristic resistance. At RF frequencies, you can make a PIN diode's resistance look like anything from near infinity to near zero just by changing the DC bias current through it. Makes remote-controlled attenuators easy to build.

For simple on/off switching circuits, ordinary diodes often are a better choice.

Msg#: 5958
From: Russ Reiss To: James Meyer

The "heating of the resistors" can be eliminated by using chokes instead. This is also where quarter-wave lines can come in handy as long as the design only has to work over a narrow bandwidth (like my 2-m unit). An "AC short circuit" at the bias source end transforms into an open-circuit at the RF switching point, and the losses become negligible.

As for being a "low-power" device, not true. They have been used up in the kilowatt range with proper design. I was amazed how well the unit I constructed (from some kind of ordinary diodes) worked at around 100 W. The trick is in turning the diodes on during transmit. Then, residual rectification doesn't happen since the receive signal is never great enough to forward bias the diodes. You just have to ensure that you pump enough bias current through the diodes that any AC/RF component doesn't succeed in turning them back off again while transmitting. But if the diode is slow enough (at the frequency employed), it has a hard time doing this anyway.

Jim's circuit is but a starting point using a single series diode as a switch. You can add more series diodes for greater isolation and diodes which shunt various paths to ground for further protection. And, as I keep saying, you can do real magic with diodes and quarter-wave transformers!

CONNECTIME

Msg#: 6012
From: James Meyer To: Russ Reiss

> The "heating of the resistors" can be eliminated by
> using chokes instead.

I realized this mere microseconds *after* I posted my message. 8-)

Not to defend myself for saying that PIN-diode switches are only for low power (they *can* switch kilowatt levels), but in most cases they only have to actually pass small amounts of power. Take the typical ham's "CW break-in" Tx/Rx switch, for example. The diode is in there to isolate the receiver input from the transmitter output to prevent overloading. The transmitter's output normally doesn't pass directly through the diode part of the switch.

Motor speed control

Msg#: 5656
From: Dave Friedeck To: All Users

I am working on a project that requires a microprocessor to control the speed of a 220-VAC universal (one with brushes) motor. I need to reach full motor RPM, so an SCR will not work, but some triac circuit should do the trick. I hope to cycle an output pin off the micro at various duty cycles to do the trick. Now the problem: I hate high voltages and blowing up good microprocessors. Any ideas?

Msg#: 5775
From: Pellervo Kaskinen To: Dave Friedeck

Sounds like you are talking about a single SCR? If that is true, then of course you would get the effect of half-wave rectifying and lose much of the speed range unless you add four diodes in a bridge. Then you could feed your motor either with DC or AC as you prefer and still do with a single SCR.

What I'm talking about is a connection of a single SCR from the common cathodes (positive out terminal) of the bridge to the common anodes (negative output terminal), while connecting the AC load in series with the AC supply and one of the two AC terminals of the bridge. It works if you take care of the stray inductances and transient suppression. Otherwise, the SCR may fail to turn off.

The same problem tends to plague the DC-load version. There, the load is simply in series with the SCR inside the diode bridge. I have once battled it and found that I had to put a 6.8-µF capacitor across the 24-V transformer secondary before the SCR started turning off.

Due to the commutation issues, I prefer to use at least two SCRs (together with two diodes or with a center-tapped transformer). Then I have a choice of adding an extra freewheeling diode or omitting it. I also could get the freewheeling effect if I connect the two diodes in series and the two SCRs in series to form my bridge. But, I generally like the idea of connecting the cathodes of the SCRs together so the common-cathode point becomes the control-circuit common.

Whatever bridge configuration you choose, the important issue is providing a good gate drive at the correct times. That translates to the following rules in my book:

- The trigger circuits have to be synchronized to the line.
- The gate drive circuits in most cases have to be isolated from the power circuits. In case of the two SCR cathodes tied together, you may be able to avoid this.
- The gate pulses have to be sturdy enough to guarantee proper firing of the SCR. This can be done with a UJT or a PUT plus a transformer. I prefer a blocking oscillator that produces a 70-µs pulse of about 7 V, 1 A capability.
- The control circuit has to cover the range of O-180" of electrical as close as possible, without losing the synchronizing.
- The control circuit most likely should contain either feedback or some linearization to make the output motor speed a reasonably linear function of the input voltage.
- You always need a good snubber network over each SCR. They moderate the inductive kick that comes when an SCR switches either on or off. They also may provide the necessary initial current to keep an SCR turned on before the current can start flowing through an inductive load, although that purpose is better served with a longer gate pulse. For most SCRs, a **15-25-p** pulse does fine with a resistive load, while 70 or more microseconds is required for a partly inductive [motor) load.

Come to think of it, most of these rules are equally valid for a triac use. The only difference may be that you are likely to use the triac circuit for an AC motor.

There have been reports of poor results when trying to control a ceiling fan speed with a triac circuit. One of the main problems generally is that the triac trigger circuits are not guaranteed to produce symmetrical positive and negative half cycles. In that respect, I'm still in favor of the two- or four-element SCR bridge.

PC keyboard lock

Msg#: 5601
From: Calvin Krusen To: All Users

I would like to know how the keyboard lock [the elevator style cylinder lock on the CPU box) is integrated into the operating system. Does the POST use the state of this to prevent the boot-up sequence, or is it something that can work any time the PC is powered up?

What I would like to do is "Lock" the keyboard (of one of our servers) so that "accidental" key strokes don't cause problems.

I was doing this for a while, but found that some times after unlocking the keyboard, certain keys were remapped. For example, the Num Lock key functioned as the Esc key (Num Lock LED didn't toggle when Num Lock key was pressed]. Also, none of the cursor keys would work. I used to do a cold boot (power off) to remedy this problem, but later found that I only needed to momentarily disconnect the keyboard connector.

Does the keyboard interface on the PC truly recognize the keyswitch? *Or* is it that the keyswitch just breaks several signals between the controller and the connector?

Msg#: 5629
From: Ed Nisley To: Calvin Krusen

It's worse than you think: the lock switch goes to the keyboard controller. The controller is supposed to suppress keystrokes from the keyboard and ignore (most) commands from the system when the lock is active. There's a command bit that disables the lock function, which lets the system send some commands and receive some responses even when the keyboard is "locked" at the front panel.

It sounds like the BIOS and your keyboard controller aren't getting along well at all. My guess is that the controller is ignoring a command that the BIOS thinks ought to work fine; I haven't a clue as to how you'd track that down, though.

As to how the keys get remapped, that's a =real= puzzle!

Msg#: 5779
From: Calvin Krusen To: Ed Nisley

Is the above mentioned keyboard controller the one in the PC or the one in the keyboard?

Also, I often find that a similar problem occurs after running Central Point Backup (run on a daily basis). I noticed that when CP Backup is running Ctrl-Alt-Del does not work (had to try the old three finger solute many times). How does the running program disable the Ctrl-Alt-Del? Is it an operating system hook or direct mashing of hardware registers (I'm guessing the keyboard controller has some specific I/O for this function)?

Anyway, in a "working" system would it be normally acceptable to disable the keyboard while either a program is running, or just sitting at the DOS command prompt?

Msg#: 5788
From: Ed Nisley To: Calvin Krusen

Curiouser and curiouser!

I was thinking of the keyboard controller on the PC system board, but it could also be the one out in the plank if you've got a particularly bizarre clone. No way to tell.

The canonical way to "disable" the three-finger salute is by hooking Int 8 and discarding the "third" keystroke. There's an official, documented, really-truly BIOS hook that gives you access to all keystrokes after the BIOS processes them and before it does anything, but I don't know if that function gives access to Ctrl-Alt-Del sequences.

In any event, there's nothing specific in the keyboard that makes Ctrl-Alt-Del do what it does. The BIOS defines that function, which is why it doesn't work when you're running, oh, say, some offbeat protected-mode code such as you'll see in a few months.. .

Methinks disabling the keyboard is rude, if only because it's hard to do anything else after the program crashes!

(Not much of an answer in there, I'll admit. Sounds like the only way to track it down is the relentless application of logic, accompanied by swapping all the hardware in sight. Phooey!)

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 (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps. For information on obtaining article software through the Internet, send E-mail to info@circellar.com.*

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. 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 (203) 875-2199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

I R S

434 Very Useful 435 Moderately Useful 436 Not Useful

STEVE'S OWN INK

Welcome to Gambleticut

Virtually every region of the country has its never-ending news story. Every night, for what seems like forever, the news always starts with the same old saga. Here in Connecticut, the big story for the last couple years has been Indian Gambling (i.e., Foxwoods Resort Casino).

The only reason I mention this is because writing a serious editorial is being inhibited by fits of laughter about an earlier computer adventure. Very few people know that one of my first "commercial" experiences with computers was gambling!

Back in 1977, while I was working for Control Data Corp., my fellow worker Ralph conned me into letting him use my computer. While we worked for a computer company, private access to one was another story. My full-blown CP/M Digital Group Z80 system became the prime target of his scheme to run a betting-odds program and beat the bank at jai alai.

Eventually, I weakened (ran out of excuses) to the constant hustle and agreed. For the next two weeks, my life was a disaster. I thought I was the only pawn until the six members of Ralph's SWAT (Special Wagering and Tactics) team arrived at my house after work each day. Even though I loved to cook, two weeks of gourmet entertaining (Ralph liked to eat) made me feel like a cruise director. When I wasn't playing chef, I was the computer's field-service engineer.

Getting rid of these guys meant agreeing to move the computer to a motel next to the jai alai fronton. I'll forgo the sordid details associated with seven guys checking into a single motel room.

In any case, because the eight jai alai players and their statistics weren't available until shortly before a game, the betting program couldn't be run until then either. Also, without the benefit of cellular faxes and phones, runners got the statistics and betting calls back and forth between the fronton and the computer operator at the motel.

I'm not a gambler. I don't even buy lottery tickets. The fact that I was involved in this at all was embarrassing enough. I volunteered to stay in the motel while the others played runners. Ralph sat and played money man, of course.

Since I've always professed myself to be a hardware guy at heart, I never really thought too much about their betting algorithm. I guess I just wanted to see if the hardware was up to it. At the appointed times, I entered the data and printed the computer's predictions for the order in which the eight players would end up. Ralph took the first three predictions and bet on them as a boxed (any two of the six possible combinations) win, place, and show trifecta bet. Because this was still experimental, the bet was only $18 each time.

Each time I gave the runner a new printout, he'd give me the results of the previous set. Ralph had won $83 on the first game but had lost the next six in a row. As I passed the predictions to the runner for the eighth game, I studied the results of the previous seven and nearly fell off the chair.

While the win, place, and show numbers had only come up once in seven games, within the first three numbers in the computer's prediction, they came up five times in the fourth, fifth, and sixth predicted positions! When the runner came back with the eighth game results, the trend held true again.

Seeing that three out of four games were being forecast correctly (albeit not directly) was too much. I instantly volunteered to be the new fronton runner. Of course, I stopped at the trifecta betting window on each pass.

To make a long story fit one page, let me just say that by the end of the 12-game evening, Ralph had lost $97. The good news was that I had won $110 on the three games I played! Even so, I looked at it as an adventure, not a new profession. To this day, I have never been back to jai alai again, nor have I ventured to Foxwoods. The businesses I'm already in involves about as much gambling as I can take.

Steve

P.S.: I'll be at the West Coast Embedded Systems Conference on September 12 and 13.