

CIRCUIT CELLAR

INK®

THE COMPUTER APPLICATIONS JOURNAL

March 1994 — Issue #44

CROSS-DEVELOPMENT TOOLS

Choosing the Right Tool for the Job

Inexpensive 68HC11 Development

Double the Speed of your 8051 Circuit



\$3.95 U.S.
\$4.95 Canada

EDITOR'S INK

Double, Double, Toil, & Trouble



Have you ever noticed how code debugging more closely resembles a black art than a science? The religious wars over what technique is best to use under what circumstances rivals only what the best language is to use. Some will tell you that only a multithousand-dollar ICE will suffice when trying to track down your latest microbe, while others prefer to rely more on tricks, technique, and intuition. I'm in the latter camp.

I'm sure many will cringe, but my personal primary debugging technique is still the old crash and burn method. Write the code, try it, and watch it crash. Well...there is more to it than that.

A large part of my coding technique involves careful, step-by-step design, development, and testing. Thoroughly map out the algorithm in your head before writing the first line of code. Next, add only a small amount of code between tests and run that code in your mind at least three times to make sure it will do what you expect. Then test only that small portion. Finally, on a larger scale, make sure your new code hasn't broken the whole thing. Every time I try to make massive changes or add lots of code at once, I'm bitten by multiple bugs that seem to feed off each other. Take small, deliberate steps and you're most of the way there.

Some bugs do need more sophisticated techniques when time is of the essence, but don't overlook other tried-and-true methods when money is also important.

Speaking of debugging techniques, we start off this month with an overview of cross-development and debugging tools and some advice on what to use and when. There are many factors to take into account, so it's definitely not a cut-and-dry issue.

Next, we look at a low-cost, roll-your-own alternative for doing 68HC11 code development. As I've said, it doesn't have to be expensive.

When it comes to training tomorrow's engineers, nothing beats hands-on experience. With our colleges as money strapped as the rest of us, hands-on work is often a luxury rather than the norm. Take a look at how one college professor is giving his students experience without paying a king's ransom.

In our final feature, find out how to more than double the speed of your 8051-based controller without making a single hardware or software change to the board (other than replacing the processor). Do watch those software-based delay loops, though.

In our columns, Ed continues adding a large LCD panel to his embedded '386SX project by writing some support code. Jeff puts in a plug for the Institute for Interconnecting and Packaging Electronic Circuits and describes some of the benefits of joining your local Designers Council. Tom describes the latest in programmable parts that can be updated while in circuit. John starts the hardware of his embedded control system with the processor, backplane, and power control. Finally, Russ searches out patents related to cross-development tools.

CIRCUIT CELLAR **INK**®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcla

PUBLISHER
Daniel Rodrigues

EDITOR-IN-CHIEF
Ken Davidson

PUBLISHER'S ASSISTANT
Sue Hodge

TECHNICAL EDITOR
Michael Swartzendruber

CIRCULATION COORDINATOR
Rose Mansella

ASSOCIATE EDITOR
Rob Rojas

CIRCULATION ASSISTANT
Barbara Maleski

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

CIRCULATION CONSULTANT
Gregory Spitzfaden

WEST COAST EDITOR
Tom Cantrell

BUSINESS MANAGER
Jeannetre Walters

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

ADVERTISING COORDINATOR
Dan Gorsky

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

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) 875-2751. Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions \$21.95, Canada/Mexico \$31.95, all other countries \$49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 7694, Riverton, NJ 06077 or call (609) 786-0409. POSTMASTER: Please send address changes to The Computer Applications Journal, Circulation Dept., P.O. BOX 7694, Riverton, NJ 06077.

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

NORTHEAST
Jebra Andersen
(617) 769-8950
Fax: (617) 769-8982

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

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

MID-ATLANTIC
Barbara Best
(908) 741-7744
Fax: (908) 741-6823

MIDWEST
Nanette Traetow
(708) 789-3080
Fax: (708) 789-3082

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps, 8 bits, no parity, 1stop bit, (203) 871-1988; 2400/600 bps Courier HST, (203) 871-0549

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

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


Entire contents copyright © 1994 by Circuit Cellar Incorporated. All rights reserved. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.


1 4 Tools for Debugging Real-time Embedded
Computer Systems
by Noel Anderson, Dean Hoyt, & Ron Shaw


2 2 Inexpensive 68HC11 Cross-development
by Bruce Olney


3 0 Computer "Train"ing/Using commercial evaluation
boards in teaching
by Michael Smith

4 0 Improving the Performance of an 8051-based System
by Michael Alwais

48  **Firmware Furnace**
Testing the '386SX Project's Bitmapped
LCD Panel
Ed Nisley

5 6  From the Bench
Designers Spell Relief "IPC"
Jeff Bachiochi

6 0  Silicon Update
Update Your Design On the Fly
Tom Can trell

6 6  **Embedded Techniques**
Getting to the Embedded Core/The DS2250,
Backplane, and Power Control
John Dybowski

INSIDE ISSUE 44

2 Editor's INK
Ken Davidson
**Double, Double,
Toil, & Trouble**

6 Reader's INK
Letters to the Editor

8 New Product News
edited by Harv Weiner

75 Patent Talk
Russ Reiss

ConnectTime
Excerpts from
the Circuit Cellar BBS
conducted by
Ken Davidson

Steve's Own INK
Steve Ciarcia
Superhighway Limits

Advertiser's Index

82

96

81

READER'S INK

Termination Concerns

I love the magazine you put together and hold it in very high regard. Therefore, I was a bit disappointed about the article on high-speed PCB design in the January issue. I found it to contain omissions and factual errors.

The biggest error was the claim that an AC termination would cause "capacitive loading of the driver and distortion of the waveform due to the R/C time constant." This would point to a severe misunderstanding of the principle of AC termination or a gross miscalculation of component values! In fact, if the resistor correctly terminates the transmission line and the capacitor is sufficiently large, then in AC terms the capacitor creates a virtual ground and does not matter at all.

In this context, I also have to complain about the omission of active termination, where you terminate the line to a separately regulated voltage through a single resistor.

Moreover, the listing of the four termination methods as presented in the article could suggest that the methods were exclusive in nature. On the contrary, the fact that complete termination requires both ends of a transmission line to be terminated was conspicuously absent.

I would also have expected a more thorough coverage of the subject. Then again, I may be spoiled by the excellent quality of material you publish!

Peter G. Holzleitner

Vienna, Austria

peter@softwork.mhs.compuserve.com

Now, what was job #1?

I thoroughly enjoyed Jeff's "From the Bench" article about electronic caliper interfacing in the December issue. However, I do have a comment about his lead-in, where he says, "inspection is (or at least it should be) job #1."

It has taken the industry many years, but mostly they realize now that this is wrong. In the Detroit suburb where I work, we often say "quality is job #1," but quality is *not* the same as inspection. The common saying is, "You cannot inspect in quality, you must design it in." Perhaps an example is the simplest way to explain.

Say we have two computer companies. Company A has a 1% defect rate. They catch these in final inspection

and (hope to) ship 100% good units to the customer. Company B has a 50% defect rate. Half of their units don't work at final inspection. They catch those, and only ship good units to the customers. The customers get 100% good units, right? But if you were the customer, which company's product do you think would last longer and be more reliable? Equally important, which company do you think is making money, if their products are priced about the same?

The lesson is obvious. Real quality (and lower costs) comes from understanding and controlling the production process, not from inspection. Inspection is to catch the mistakes, and verify that the production process is working correctly. Inspection *is* important, but it *is not* job 1.

Whew. Sorry about the lecture. Hope I have explained my point without offending, or detracting from your otherwise excellent article.

Erik Kauppi

Ann Arbor, Mich.

(via the Circuit Cellar BBS)

Thanks for your comments. I don't think quality necessarily pertains just to inspection. As you've pointed out: garbage in, garbage out. Quality is important to all processes whether it be design, procurement, or manufacturing.

-Jeff

Excuse me, but I just could not resist the temptation to nitpick! I have had some change of mind in this area through the years and have started seeing the reasons for inspection. What I mean is the inspection of the components and materials that come in.

We used to just put them into stock and then do any checking when we used them. But that led to near disasters more often than what I am willing to think of. So, I now am fairly happy about the formal incoming inspection that we have where I work. I, of course, have to provide them with the specifications of everything and that is a pain in the neck, but seeing how often we get something other than what we ordered is really astonishing.

So, can I say that inspection is sometimes the first thing? At least it is the first thing in the material flow sense. Of course, there has been already plenty of engineering activity in selecting what we plan to use in the first place. And that comes before the inspection, I admit, but still..

—Pellervo Kaskinen

READER'S INK

CD-ROM, anyone?

Congratulations for your very interesting pages that I think are unique in the world. The main problem for the European reader is for the expensive charge of intercontinental call to your BBS. I suggest you collect the more interesting embedded software in an annual CD-ROM for the subscribers.

Ermanno Beber
Loreggia, Italy

Let me remind you that magazine-related software is also available via the Internet. If you have net access, it costs nothing extra to get our files from halfway around the world. The files are also available on floppy disk each month.

We've toyed with the idea of electronic publishing on CD-ROM. Right now, though, we just don't have the time or the staff to pull off such a feat. Perhaps it's something we'll try someday.

-Editors

Contacting Circuit Cellar

We at the *Computer Applications Journal* 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, The Computer Applications Journal, 4 Park St., Vernon, CT 06066.

Phone: Direct all subscription inquiries to (609) 786-0409. Contact our editorial offices at (203) 875-2199.

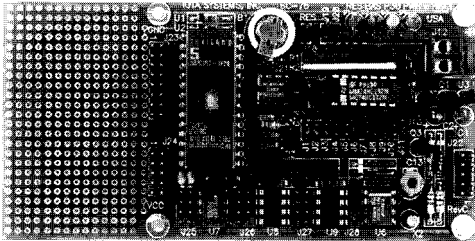
Fax: All faxes may be sent to (203) 872-2204.

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 Email to Jeff Bachiochi, address it to jeff.bachiochi@circellar.com. For more information, send Email to info@circellar.com.

NEW!

876752 CONTROLLERS



Iota continues its strategy of developing small powerful controllers with its new line using the 87C752, 87C751 and 87C750 microcontrollers. We have also developed our own BASIC-752™ which runs in the 87C752. Several software tools are available to ease the development process.

We offer:

- I²C-bus peripherals
- Complete Development Packages
- Powerful controller BASIC for the 87C752
- A BASIC-752 Simulator

Voice: 702.831.6302 • Fax: 702.831.4629

Iota Systems, Inc.

POB 8987 • Incline Village, NV 89452-8987

"Is the
TERROR
of BIG S-L-O-W software
stalking your job security?"

Take some of the bytes out of your bloated code!

Call Franklin Software, the first and last word in quality for

your 8051 and 80C166 software projects!

F. FRANKLIN
SOFTWARE, INC

Call for your free Release VI Evaluation Kit!

tel: (408) 296-8051 fax: (408) 296-8061

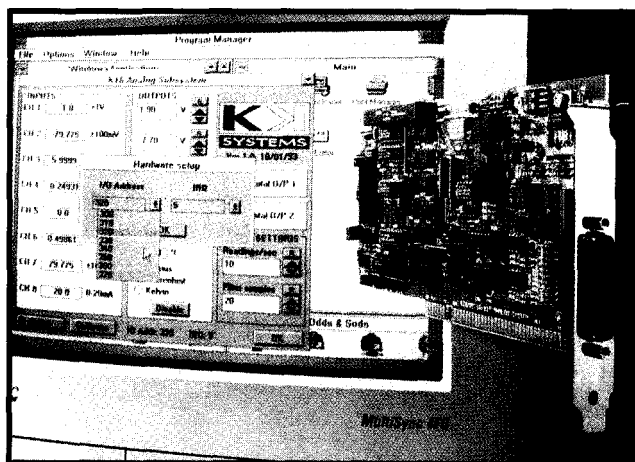
NEW PRODUCT NEWS

Edited by Harv Weiner

16-BIT DATA LOGGER FOR WINDOWS

A low-cost data acquisition system that provides a quick way to record 16-bit data with a PC and offers "plug-and-play" simplicity is being offered by the Saelig Company. The K16/PC offers multichannel isolated 16-bit data logging with very high conversion speeds.

The K16/PC is a unique type of analog interface card for the IBM PC and compatibles. Eight 16-bit inputs can be individually programmed for voltage or 4–20-mA current inputs. The voltage range is software selectable to ± 1 V or ± 100 mV. The K16/PC is autocalibrated to 0.02% accuracy using factory-loaded correction factors stored in an on-board EEPROM. An on-board temperature sensor, which can be used for temperature compensation in critical applications, is also provided.



A full-featured Windows 3.1 Virtual Multimeter program, written in Visual Basic 2, is included. It is supplied in .EXE form with the VB distributable runtimes and therefore does not require Visual Basic. The program supports logging a user-defined choice of A/D signals, digital input states, and card temperature into a disk file at intervals from 55 ms to 60,000 minutes. Each data group (stored under a user-defined name) can be prefixed with time, the log interval, or nothing. This constitutes the X-axis if imported into Excel for graphing, for example. The sophisticated multimeter software includes a programmable low-pass filter and supports up to eight simultaneously installed K16/PC cards. A complete Windows Help file is also provided. DDE communication is supported, which allows communication with other DDE-aware applications without having to import/export files.

All analog connections are in a 25-pin D-type connector and all connections are isolated from the connector shell. All inputs and outputs share a common analog ground. K16/PC options include two 16-bit D/A converters for control applications, TTL I/O, and a floating +10-V and +24-V power supply for transducer excitation. The K16/PC sells for \$399.

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

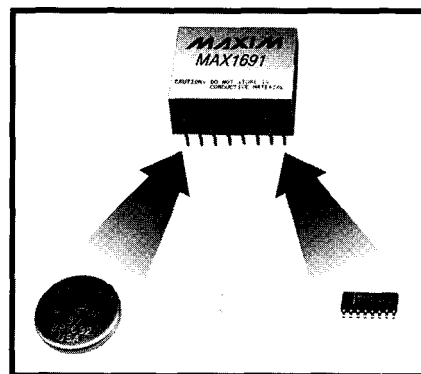
#500

MICROCOMPUTER SUPERVISOR MODULE

Combining an IC with a lithium battery, the Maxim MAX1691 microprocessor-supervisor module reduces complexity and saves board space in power supply monitoring and battery backup circuits. Applications include computers, controllers, and critical microprocessor power monitoring.

The MAX1691 includes backup battery switchover, memory write protection, and a watchdog function that monitors software execution by asserting a reset in the absence of normal digital activity on a selected I/O line. The MAX1691 switches automatically to the internal +3-V, 125-mA lithium battery when V_{cc} is below V_{bat} and below its own reset threshold. RESET and power-fail outputs assure that the controlling microprocessor assumes a known state during power up, power down, and brownout conditions. Quiescent operating current is 35 μ A, and standby current is 1 μ A maximum.

To protect CMOS RAM from erroneous write operations during power failures, the MAX1691 gates the RAM's chip-enable signal. It disables RAM by blocking CE when RESET is asserted, and delays CE no more than 10 ns during normal operation. The MAX1691 comes in a 16-pin plastic DIP and is tested for the commercial (0° C to $+70^{\circ}$ C) temperature range. Prices start at \$6.40 in quantity.



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

#501

NEW PRODUCT NEWS

144-LINE DIGITAL I/O CARD

A high-density, STD-bus digital I/O interface with interrupt event sensing has been announced by WinSystems. The LPM-I0144 supports 144 points of bidirectional digital I/O on a single card. The card can monitor for both rising and falling digital edges, latch them, and then signal the host processor that a change of input status has occurred. This feature allows the user to do real-time process monitoring in the background without polling the input status of each channel.

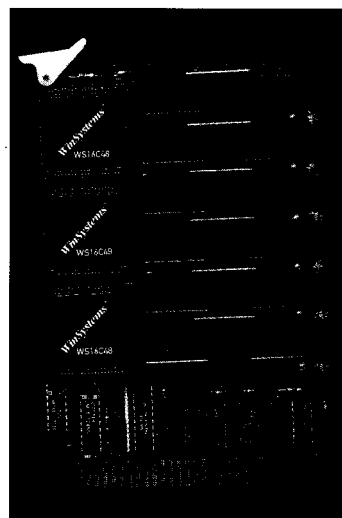
The high density of lines is the result of an ASIC device called the WS16C48. This ASIC increases single-card point density plus facilitates the sensing of event-related inputs without requiring the time-consuming procedure of polling. Up to three 16C48s are used per board, each capable of supporting 24 event sensing lines and generating an interrupt when an event occurs. Transition polarity is programmable and enabled on a bit-by-bit basis. Each line's transition is latched so even short-duration pulses will be recognized. An interrupt ID register is maintained for each line to allow more efficient ISRs.

Each I/O line is individually programmable for input, output, or output with readback operation. The input lines are connected so the current status of its output port can be read from the corresponding input port. Each output channel is latched and capable of sinking 12 mA of current. This feature allows direct control of up to 144 optoisolated conditioning modules with a single LPM-I0144 card.

Three options are available. The LPM-I0144 has three 16C48s on board, supports 144 points, and sells for \$350. The LPM-1096 contains two 16C48s, supports 96 points, and sells for \$295. The LPM-1048 contains one 16C48, supports 48 points, and sells for \$240.

WinSystems, Inc. • 715 Stadium Dr., Ste. 100 • Arlington, TX 76011-6225
(817) 274-7553 • Fax: (817) 548-1358

#502



INTELLIGENT COMMUNICATIONS COPROCESSOR

The COMMintelligence System from Drumlin solves many PC communication problems by adding a second processor to handle low-level communication line activity. Communication throughput and ease of writing custom communication applications can be greatly enhanced with this system.

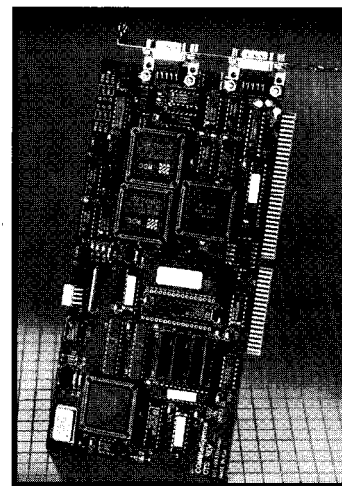
The CTS2 and CTS4 coprocessor boards have two or four PC-compatible COM ports, respectively. The CTS board is unique in that it can be programmed by a developer in Microsoft or Borland C/C++ using software normally used for PC COM ports. The on-board CPU communicates with the PC host processor through the ISA bus using dual-ported memory.

The CTS board includes firmware that provides ISA bus access to deeply buffered serial data held on the CTS board. A COMMintelligence application uses the supplied software to access CTS board COM data. A standard linkable C library is supplied for development of DOS applications; and a Dynamic Link Library (DLL) is supplied for Windows applications.

A COMMintelligence Developer's Kit is available to create front-end programs to execute on the CTS. Software supplied with the kit includes a message-based, event-driven main line handler, and a buffered, interrupt-driven COM library that supports full asynchronous operation and RS-485 multidrop connections. The on-board execution environment emulates DOS, providing BIOS and DOS interrupt services immediately upon power up. An .EXE file created with popular PC languages, when downloaded into the CTS, will execute as if it were running on a standard PC. Provisions are included for placing CTS programs in traditional EPROM, downloading them into on-board Flash memory, or downloading them into CTS RAM on each power up. The COMMintelligence system starts at \$395 with two PC-compatible COM ports. The COMMintelligence Developer's Kit sells for \$375.

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

#503



NEW PRODUCT NEWS

IN-CIRCUIT EMULATOR

Microamps has introduced an in-circuit emulator that is capable of supporting almost all 8051 derivatives. Using a new technique that combines the best features of an EPROM emulator with a full-function in-circuit emulator, the **BICEPS51** replaces the EPROM in the circuit under test with 64K bytes of emulation RAM that can be partitioned in 4K-byte blocks.

The BICEPS51 puts no restrictions on memory space, registers, ports, or interrupts. Real-time supervisor logic maintains control of the processor during program execution. Several breakpoint options are available to halt program execution. The processor status and memory can be investigated during CPU halt.

The EPROM adapter contains a socket for the original EPROM from the test board. The adapter can then use the EPROM with these advantages: the EPROM can be used to start the processor, the EPROM can be copied into emulation memory, and the EPROM can be mapped in 4K-byte blocks.

The BICEPS uses a single adapter to emulate almost all 8051 derivatives. The package comes with high-level debugging, hardware breakpoint capability, MSC51 family cross-assembler, and a real-time trace buffer that can be interrogated "on the fly" without interrupting processor execution. The unit connects to the serial port of an IBM PC or compatible. The BICEPS51 sells for \$995 and comes complete with software, power supply, cables, and manual.

Microamps Ltd.
66 Smithbrook Kilns
Cranleigh, Surrey
GU6 8JJ United Kingdom
t44 (0)483 268999
Fax: t44 (0)483 268397

#504

ATTEND THE SECOND ANNUAL



JUNE 9-10, 1994
SANTA CLARA
CONVENTION CENTER
SANTA CLARA,
CALIFORNIA

Hardware and Software Solutions for Embedded Computer Applications

Who Should Attend Hardware and software engineers who put computing power to work in embedded applications.

Technical Program "32 bits or better" The demand for higher performance, smaller size, lower cost and faster time to market, is forcing the move to more powerful and advanced solutions such as 32- or 64-bit RISC or DSP for processing; ASICs and multichip modules; high-performance backplane buses and more...

You'll find two and a half days packed with a combination of theoretical, practical, and application orientated sessions on incorporating high-performance embedded computers and developing embedded realtime software for complete systems and subsystems.

Exhibit Hall A comprehensive mix of leading vendors will talk with you about solutions to your specific embedded and dedicated applications.

See The Latest In: ☐ microprocessors ☐ single-board computers ☐ peripheral boards ☐ operating systems ☐ realtime kernels and compilers ☐ development systems ☐ CASE and debugging tools ☐ embedded PCs ☐ and more...

To receive detailed information on seminars and attending ECC, please complete and return the coupon below. FAX TO: (203) 838-1447, OR CALL (203) 831-9444



MAIL COUPON TO: ECC / BAV Expositions, Inc.,
85 Washington Street, Norwalk, CT 06854

☐ Send Me Information On Attending ☐ Send Me Information On Exhibiting

Name

Title

Company

of Employees

Address

City

State

Zip

Phone

Fax

Sponsoring publications: Computer Design, EE Times, Military & Aerospace Electronics, VMEbus Systems. Associations: MMG, PC/104 Consortium, SIG32, STDMG.

CPJ

#105

NEW PRODUCT NEWS

HALF-SIZE 486SLC CPU BOARD

A PC-compatible single-board computer which integrates the functionality of a standard motherboard, peripherals, and watchdog timer on a half-size card has been announced by MCSI. The ICH-486SLC is powered by the Cyrix 486SLC microprocessor, which contains an internal 1K cache, a RISC-like execution unit, and hardware multiplication logic. The card provides a Landmark V2.0 rating of 108 and is ideally suited for use in embedded applications.

The ICH-486SLC board also includes a math coprocessor socket, two serial ports, a parallel port, dual floppy disk port, an IDE hard disk port, a PS/2 keyboard port, up to 16M bytes of DRAM, and an on-board speaker. In addition, each board contains a standard PC/104 expansion port for adding optional boards such as EPROM/SRAM, VGA controller, and digital I/O. Since the ICH-486SLC was designed for embedded applications, the BIOS will allow booting without either a keyboard or monitor.

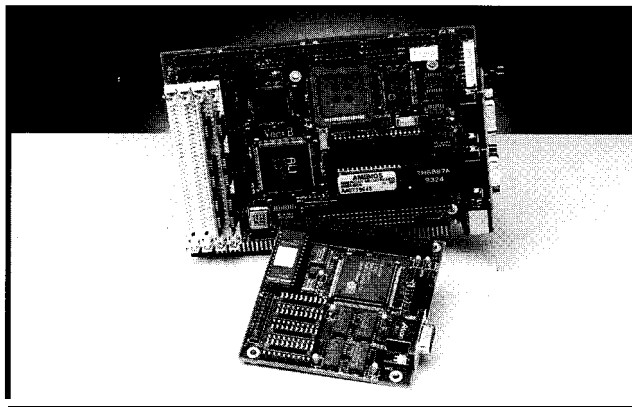
The watchdog timer makes the board ideally suited for controlling critical processes where unattended operation is essential. In the event of an I/O timeout

delay or external failure, the watchdog can be programmed to generate a nonmaskable interrupt or reset.

Power consumed by a fully populated board operating at 33 MHz is only 7 W. The ICH-486SLC single-board computer comes complete with user's manual and carries a 2-year warranty. Pricing starts at \$495.

Micro Computer Specialists, Inc.
2598G Fortune Way • Vista, CA 92083
(619) 598-2177 • Fax: (619) 598-2450

#505



ACSBUSS Modular Control

80386
68306

OUR CPU's HAVE
"CONNECTIONS" !

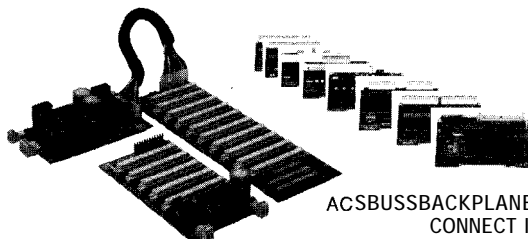


ACSBUSS "CONNECTIONS"
bring your development
& production costs
under CONTROL!

YOU'RE IN CONTROL with our

ACSBUSS "CONNECTIONS" for:

CONTACT INPUT, A.C. SENSE, CONTACT OUTPUT, TRANSDUCER,
TRIAC OUTPUT, PRECISION MOTION CONTROLLER, USER INTERFACE,
RAM EXPANSION, VIDEO DISPLAY, DIGITAL AUDIO, MODEM & FAX,
REMOTE PROGRAM LOADING. *and many more every day!*



ACSBUSS BACKPLANES
CONNECT IT ALL!

ACS

Call or Write for more info!

On The Cutting Edge of Technological Evolution

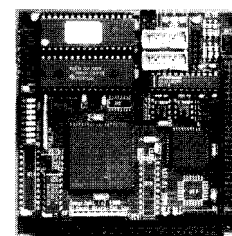
4276 Lago Way • Sarasota, FL 34241
Ph.(813)377-5775 • FAX(813)378-4226



We're Small, We're Powerful, And We're Cheaper.

MMT-188 EB

- 2 serial I/O ports
- 3 programmable parallel I/O ports
- 1 Meg RAM/ROM capable
- powerfail detect interrupt and reset
- counter- timers
- watch dog timer
- expansion connector



Only \$191⁰⁰ (Qty 100)

ALSO AVAILABLE: MMT-Z180, MMT-196, MMT-HC11, MMT-EXP

In fact, you'll get the best product for about half the price. If you're interested in getting the most out of your project, put the most into it. For the least amount of money.

Call us today for complete data sheets, CPU options, prices and availability.

Custom Work

Welcome. Call or fax for complete data sheets

2308 East Sixth Street

Brookings, SD 57006

Phone (605) 697-8521

Fax (605) 697-8109



WE'RE SMALL BUT WE'RE POWERFUL.

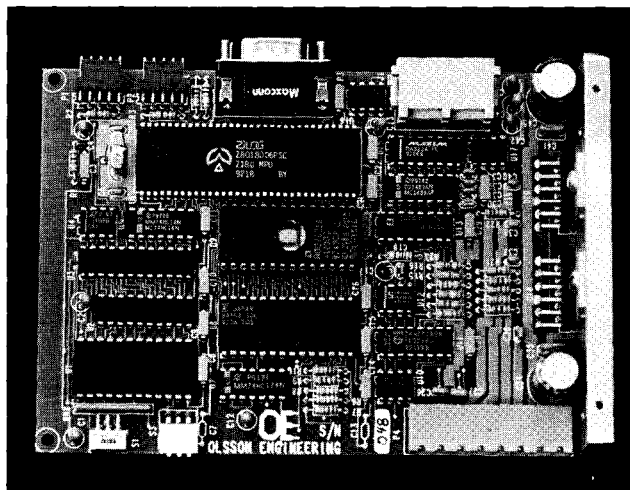
NEW PRODUCT NEWS

MICROSTEPPING STEPPER MOTOR CONTROLLER

Olsson Research has introduced a "smart" microstepping motor controller that provides a single axis of motion control for a wide range of industry-standard stepping motors. The MC600 is easily integrated into robotic and instrument applications.

The MC600 integrates motion-control parameters and pulse-width-modulated waveforms of 3 A/phase into a single device. The MC600 operates on 12-48 VDC and is fully programmable to interface to a wide range of nominal motor voltages (2.0-24.0 VDC). Microstep resolution is factory set at 256 microsteps per step. Advanced waveform algorithms optimize torque and accuracy over the entire operational range (1800 RPM max. for 1.8" motors). Positional information is maintained to 32 bits and limit sensor inputs are available. The unit is controlled from any host with a serial port and includes a powerful ASCII command language.

Power conditioning and motor waveform generation are handled by an on-board microcontroller. The MC600 measures 4"x6"x1.5" and features short circuit and over



temperature protection. The MC600 Microstepping Stepper Motor Controller sells for \$319 (100s).

Olsson Research, Inc.
561 Pine St. • Edmonds, WA 98020
(206) 778-9480 • Fax: (206) 771-3994

#506

EMBEDDABLE PC

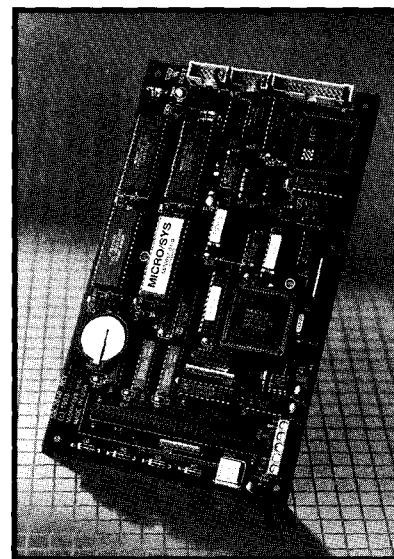
The SBC2040 Embeddable Computer from Micro/sys is a PC-compatible single-board computer optimized for embedded, diskless systems. A designer can develop and prototype software on a PC for eventual execution on an SBC2040 embedded into a larger system.

The SBC2040 features three serial ports, a parallel port, and a watchdog timer. A PC/104 expansion connector allows the user to add more functionality with industry-standard PC/104 plug-in modules. On-board memory consists of four 32-pin JEDEC sockets that will accept a mix of SRAM, battery-backed SRAM, EPROM, and Flash memory.

An optional on-board ADC adds six channels of 12-bit analog input to the SBC2040. Input voltage range is 0-5 V, with linearity of $\pm 1/2$ LSB. A 14-pin connector provides access to the analog inputs.

The firmware available on the SBC2040 reduces the development time required to embed a PC. Any PC language compiler can be used. The SBC2040 firmware includes built-in support for Borland's Turbo Debugger running in remote mode. By merely connecting a serial cable between the SBC2040 and a PC during development, an SBC2040 program can be downloaded and debugged with full source code, single-stepping, and breakpointing.

An available Flash memory package provides simplified field updates of the application program. By plugging in the special download cable, the SBC2040 firmware enters an XMODEM download mode instead of executing the installed application program. Using any PC terminal program, a new application program can be downloaded into the SBC2040. Removing the cable and reapplying power will run the revised program. This feature allows field service personnel to carry a laptop computer to an installed system, plug in a download cable, and update the internal operating software without ever opening a panel or removing an EPROM. The SBC2040 basic computer board sells for \$275; the ADC option costs \$75.



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

#507

NEW PRODUCT NEWS

IN-CIRCUIT EMULATOR FOR Z8 MICROCONTROLLER

A real-time emulator that supports the Zilog Z8 microcontroller family has been announced by Emulation Technology. The ET-iCZ8 is a high-performance in-circuit emulator with an optimized, integrated environment that provides more efficient development of embedded systems.

The ET-iCZ8 supports the complete range of standard Z8 microcontrollers in NMOS and CMOS. It also offers full symbolic debugging so that the same symbols can be used in different modules. The emulator has a 40-pin DIP socket connector and a processor-specific probe which plugs into the target system and connects to the emulator unit with a flexible ribbon cable. The static overlay RAM can be mapped to either the target or emulator, in sizes ranging from 2K to 32K bytes.

The ET-iCZ8 development environment includes a standard SAA interface, a source-level debugger, a multifile editor, and a powerful, integrated project management tool. The simple, easy-to-use interface offers pull-down menus, mouse support, context-sensitive on-line help, and function key and keyboard input. The ET-iCZ8 connects to the host PC at 57.6k bps over COM1 or COM2 ports.

The 32K real-time breakpoints can be set for three conditions: Breakpoints can stop the real-time execution of a program on a specific address. They can be used to set the trigger of the emulator only, without breaking the real-time execution of the program. Finally, breakpoints can be set to stop a program after a number of iterations specified by the loop counter, which can be set from 1 to 255.

The ET-iCZ8 can perform full-speed real-time emulation from 100 kHz up to 16 MHz and eliminates wait states or intrusion on I/O or interrupt pins. The emulator comes complete with base unit, external power supply, and a PC serial interface cable. It can be connected to any PC-compatible computer. The software with source-level debugger and manual are also included. Pricing for the ET-iCZ8 starts at \$2272.

Emulation Technology, Inc.

2344 Walsh Ave., Bldg. F • Santa Clara, CA 95051 • (408) 982-0660 • Fax: (408) 982-0664

#508

REMOTE POWER CARD!

3 VERSIONS:

RESET

WATCHDOG RESETS PC IF IT HANGS FOR HARDWARE OR SOFTWARE REASONS

PHONE

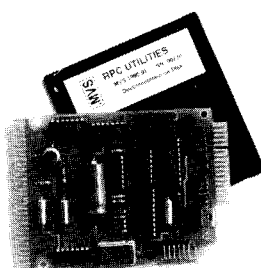
TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

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

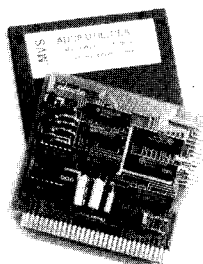
DATA ACQUISITION, SERVO CTL, AUDIO 8-BIT RESOLUTION 22KHZ SAMPLE RATE SHARP CUTOFF ANTI-ALIAS FILTER CREATE STEREO BLASTER (.VOX) FILES

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**

SU-MAR ENTERPRISES

Ja Mar Distributing

ORDERS (800)477-4181 ORDERS(410)437-2324

INFO (410)437-4181

VISA * MASTERCARD * AMEX 1292 MONTCLAIR DRIVE

FAX (410)437-3757

FEDERAL EXPRESS NEXT DAY AVAILABLE PASADENA, MD 21122

MARCH SPECIALS

PANASONIC HYBRID PHONE SYSTEM

KX-T7050	SYSTEM PHONE	\$111.82	KX-T30180	CONTROL UNIT 3-LINE 8-EXT.	\$502.92
KX-T7020	SPEAKERPHONE	135.70	KX-T61610	CONTROL UNIT 6-LINE 16-EXT	825.82
KX-T30865	DOOR INTERCOM	27.40	KX-T7030	SPEAKERPHONE W/LCD	176.40



HOME AUTOMATION

X-10 LEVITON STANLEY ENERLOGIC

SKYLINE CONTROL	SOFTWARE UPGRADE FOR X-10 CP290P	\$49.99
EVENT CONTROL SYSTEM SOFTWARE		\$250
APPROACHING HOME AUTOMATION	BOOK ON X-10 & MORE	\$18.99

HOME ENTERTAINMENT

SSI	DOLBY PRO LOGIC SURROUND SOUND DECODERS, AMPS & SPEAKERS	
AR POWERED PARTNERS	COMPUTER SELF-POWERED SPEAKERS	
FOSGATE	THX & IN-WALL SPEAKERS	
PHILIPS	CDI 220 COMPACT DISC INTERACTIVE PLAYER	\$459
COMPTON'S	ENCYCLOPEDIA INCLUDED FREE.	
MOVIE ADAPTER FOR CD1200	MOVIES & GAMES IN STOCK	\$219

CALL OR WRITE FOR CATALOG

DEALERS PLEASE WRITE OR FAX ON COMPANY LETTERHEAD

FEATURES

14

Tools for Debugging
Real-time Embedded
Computer Systems

22

Inexpensive 68HC11
Cross-development

30

Computer "Train"ing

40

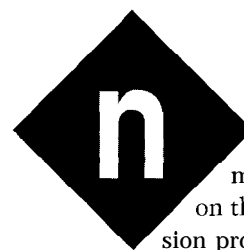
Improving the Performance
of an 8051 -based System

FEATURE ARTICLE

Noel Anderson
Dean Hoyt
Ron Shaw

Tools for Debugging Real-time Embedded Computer Systems

When all you have is a hammer, everything looks like a nail. It's important to use the right tool for the job at hand, and code debugging is no exception. Explore the tools available to the embedded designer.



Homer Abram is a master carpenter on the public television programs "This Old House" and "The New Yankee Workshop." It seems he always has the right tool to make the cuts and bring pieces of wood together into fine furniture. The components fit snug and secure the first time.

Computer hardware and software, even when produced by master engineers, seldom comes together so well. Bugs create gaps in I/O streams, freeze task rotation, and cause applications to fall apart with the first gust of wind. Freshly integrated systems often resemble woodworking creations by Homer Simpson.

Many compilers now come with integrated debugging tools for finding software problems. The tools work well when the software does not have to interact with hardware outside the computer, but are often inadequate for real-time embedded computer systems (RTECS).

RTECS debugging requires monitoring the hardware/software interface *where* the program is running, *while* the program is running. The programmer needs to observe software responses to external events.

The target environment can be physically harsh and also the target system could be too pressed for I/O and CPU resources to support the debugging overhead.

In this article, we'll set the stage for RTECS debugging with a model. Then we'll consider six types of tools: simulators, in-circuit emulators, logic analyzers, oscilloscopes, software monitors, and background debug monitors.

DEBUGGING MODEL

A generic software lifecycle is shown in Figure 1. Once any program is written, its operation needs to be verified against the original specification. Errors are iteratively located and corrected until "all" have been eliminated. After the software has been released, additional errors may be identified and need correction. Revised software replaces the old until the system is retired.

General-purpose and real-time computer systems differ in physical depth of the debugging. However, both may deal with application-specific abstractions such as objects, modules, functions, and procedures. They may also deal with machine-independent, high-level language (HLL) abstractions such as statements and data structures (Figure 2). Ideally this is where most of the functional software verification and debugging occurs because the programmer requires minimal knowledge of the underlying target hardware and the debugging can be performed on a resource-rich host system.

The next level of debugging occurs at the register transfer level and is processor dependent. It may be done on an assembly language instruction basis or by the bus activities which constitute the assembly language instructions. Errors that can be found this way include incorrect task handling and interrupt handling, mismatched parameters on calls, incorrect addressing of peripherals devices, and other software/hardware interaction problems.

The lowest level of debugging is at the physical level. It is also circuit dependent and may be done with logical quantities like 0, 1, pulse, and

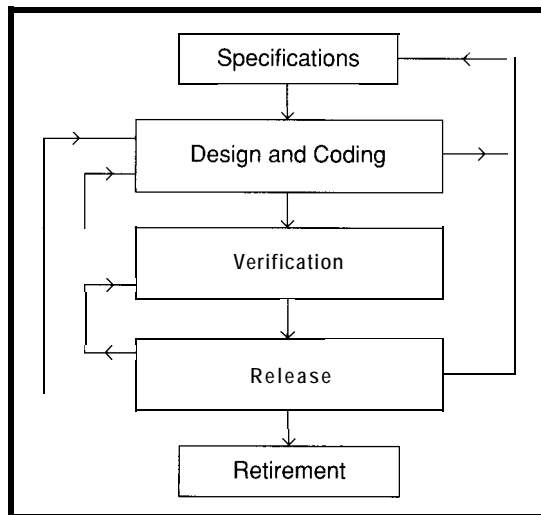


Figure 1—Most software goes through a common cycle of events during its lifetime.

edge. In some situations, the analog signal values may need to be considered. Errors related to circuit loading, incorrect peripheral initialization, software/hardware timing problems, noise, and other hardware-related problems can be found this way.

An ideal debugging tool would simultaneously observe program execution at all six levels in the target environment. It would not intrude in the system by electrically loading the system, by physically occupying space, by temporarily slowing the system, or by changing system behavior in any other way. This is so critical in RTECSs because errors are often related to electrical signal and logical timing problems. Such errors can disappear and others appear in their place if the system is changed by the observation instrument.

All the debugging levels in Figure 2 can be studied with a *simulator*. A simulator is a program written to mimic the behavior of another system. They can be written to execute HLL statements, to execute assembly language instructions, or to switch individual transistors. Since a single HLL statement may represent execution of dozens of assembly language statements involving thousands of transistors, simulation sizes and times grow rapidly with increasing simulation detail. Each simulated system action requires execution of tens to thousands of instructions on the system hosting the simulation.

The general inability of simulators to operate in the same time frame as target systems is a major disadvantage. Another is the difficulty in simulating interaction with system I/O, interrupts, and the target environment. For this reason, we won't consider simulators in the rest of this article. However, they are useful for functional testing of code that does not interact with the external world, particularly if the target hardware is not yet available or is very expensive.

The remaining tools all intrude upon the system by taking space or system resources, by electrically loading the system, or by altering timing. They differ in how they impact the system under study, in which debug levels they support, and in what they cost.

Figure 3 shows the tools in terms of how they view the RTECS and how they intrude upon the target system. The oscilloscope (CRO), logic analyzer (LA), and in-circuit emulator (ICE) measure, and consequently affect, the electrical signals of the system. The system runs continuously while the tools are in use.

When a *trigger condition* specified by the programmer is met, the tools record signal values for a period of time. This sequential record of system activity is called a *trace*. The programmer then examines traces off-line. Trigger conditions can be used to mark the start of a trace, the end of a trace, or an offset into the trace.

The *monitor* (MON) and *background debugger* (BDM) do not impact the target system electrically, but do interrupt program execution so the programmer can examine, as well as modify, register and memory values. They also usurp system resources such as memory and I/O. These tools are activated when a condition called a *breakpoint* is met. Breakpoints are frequently implemented by replacing the opcode at the breakpoint location with a software interrupt (SWI) instruction. This will not work if the breakpoint location is in ROM, as much RTECS code is. Most ICEs also support breakpoints.

These tools will now be discussed individually in more detail. Figure 2 shows their most typical use. They will be discussed in that context, but in practice, a skilled individual can use a tool to gain insight in program behavior at other levels. This advanced use is highly dependent on the reasoning ability and experience of the individual, the nature of the bug being tracked, and the exact features of the tool(s). Normal cross-domain *measurements* will be discussed later.

IN-CIRCUIT EMULATORS

An *in-circuit emulator* replaces the target system processor with an *emulator probe* attached by a short cable to a card cage. The user typically communicates with the ICE using a PC or workstation connected to the card cage by a serial data link. The size of the probe and its need to be near the larger card cage can limit the ICE to lab use. Heat produced by the probe can also cause problems for some target systems. Another disadvantage of emulators is their high cost—they are typically the highest of all the tools we'll discuss here.

A well-designed probe will have the same logical, electrical, and temporal behavior as the processor. It also provides several important debugging services. These include the ability to set breakpoints and trace triggers, to examine and modify processor registers and memory, and to time intervals between events.

Emulators implement breakpoints and trace triggers by watching for addresses and data values, rather than by modifying code. This supports use with software already programmed into ROM. Advanced emulators, like advanced logic analyzers, allow triggers and breakpoints to be expressions involving logical and sequential relations. For example, execution can be traced if an interrupt service routine is executed during a particular subroutine. This would be specified as "trigger on the starting address of the interrupt service routine after encountering the starting address of the particular subroutine but before encountering address of the last (return) instruction of the subroutine."

ABSTRACTION Languagedependent	Application	Function, model, procedures	ICE, BDM, MON
	High Level Language	Statements, data structures, variables.	ICE, BDM, MON
REGISTER TRANSFER Processor dependent	Assembly Language	ASM Instruction, operands	ICE, BMD, LA, MON
	BUS State	Read, Write, Addresses	LA (state)
PHYSICAL Circuit dependent	Logical	0, 1, glitch, edges	LA (timing) CRO
	Signals	Analog Voltages	CRO

Figure 2—The choice of debugging equipment can be made easier by considering the characteristics of the system under test.

ICEs often include memory beyond what is present on the target system. One use of the emulation memory is to provide memory for incomplete target hardware. An ICE with memory only needs system I/O working for software testing to begin. Another use of the memory is to host diagnostic and hardware test routines without using target system memory.

These memory features can be quite useful in developing software for microcontroller units (MCUs) such as the Motorola M68HC 11. MCUs, typically have a few hundred bytes of RAM for program variables, several thousand bytes of ROM for storage of executable code, and I/O ports all on one chip. An ICE can be used to thoroughly test MCU software before it is committed to ROM.

Emulation memory may be dual-ported, which means both the probe and card-cage controller can access memory in a normal target system memory cycle. The programmer can monitor variable values without having to stop the system. This facilitates nonintrusive use of the tool. Values are also easily written to memory. This can be used to test error recovery routines.

A major ability of, the ICE is to measure time between events. This time could be the time between accesses to specified memory or I/O locations, or could be the time it takes for an instruction or loop to execute.

While ICEs deal with the target processor on the register transfer level,

they can readily support HLL debugging. Software can be written and compiled on the workstation, downloaded to the target system, and executed by the emulator. The workstation and card cage can map programmer commands and queries to their machine language and absolute memory location equivalents for action by the emulator.

When multiple processors are used within a system, the usefulness of the emulator for system integration may be limited. Most RTECSs that have multiple processors require that they communicate within specific time limits. If the emulator stops execution, the remaining processors may request information or update the stopped processor. This may cause synchronization problems within the system. Careful planning is required to prevent this type of error from causing additional problems during program verification. New ICE models are providing increased support for multiprocessor systems, but this often requires one ICE per processor—a very expensive proposition.

LOGIC ANALYZERS

Debugging at the physical level can be done with logic analyzers and *oscilloscopes*. A basic logic analyzer consists of a display, a keypad for user input, a trace memory which can be up to 160 bits wide and several thousand words deep, and input lines which can be connected to target system buses and miscellaneous signal lines. Some

LAs are ISA bus cards that use the PC screen and keyboard to supply the user interface. Logic analyzers take "snap shots" of system activity in two modes: *state analysis* and *timing analysis*.

In state analysis mode, an external signal clocks the sampling of the input lines once a trigger specification is met. Using the target system clock, an LA can capture the logic levels present on the address, data, and control buses of the target system. This raw data can be displayed on the LA in numeric format or as digital waveforms.

Two accessories can aid in the analysis of the raw data. A *disassembler* or *preprocessor* can take the 0s and 1s observed on the buses and display the corresponding instruction mnemonics and operand values. This works well for code written in assembly language, but no product we know of supports logic analyzer data translation back to source HLL code. The second helpful accessory is a *software performance analyzer*. This tool indicates the percentage of time spent executing code in user-specified address ranges or counts of variable accesses.

This can be further supported by embedding special code sequences in routines so the logic analyzers can be triggered via internal processor control. An example of such a triggering method would be a write to a specified memory location with the identification number of the procedure or interrupt being executed.

In timing analysis mode, inputs are sampled at user-specified intervals once the trigger condition is met. The sampling time-base is internal to the LA. Data is displayed in numeric format or as digital waveforms.

This mode also detects and displays glitches, pulses which are as short as some fraction of the shortest sampling interval. Glitches are often spurious and can cause major problems for RTECSs, especially on external interrupt signal lines. LAs typically permit triggering on glitches, capturing low-level program execution caused by them. A key role played by the LA is in identifying external causes of RTECS software problems.



THE INCREDIBLE NEW PE-8351 FX IN-CIRCUIT **EMULATOR** **ONLY \$1451.00**

- Supports 8X51/31, 8X52/32, 8X51FA/FB/FC
- Real-time and Nonintrusive
- 64K Program Memory
64K Data Memory
- 128K Hardware Breakpoints
- 16K Frame Trace Buffer
- Transparent Trace
(View Trace and Execute Simultaneously)
- Fast, Easy Installation to RS232 Port of any Dos PC, even Laptops!
- Built-In Self-Test
- Symbolic Debug
- Source-Level Debug

- Other Fine Emulators
- 8051 From \$1400*
 - 68HC11 From \$1600*
 - 68HC05 From \$1700*
 - COP8 From \$625*
 - More than 100 devices supported through interchangeable probe cards

- New Debugger Enhancements
- Full support for structures, unions, arrays, pointers
 - Data structure browser/editor
 - True expression in watch window-detect bad pointers

Nobody Matches the Value of the PE Family

PE Product	Max Req.	MetaLink List Price*	NOHAU List Price*+
8351 FX	16	\$1451	\$4820
837511752	16	851	4300
8032-24	24	851	4945
8032-42	42	999	7195

*U.S. Retail price.

+U.S. Price List dated 3/192, verbal quote 8/93.

METALINK INVENTED LOW-COST EMULATION IN 1984.

Our new AET Emulator architecture (Advanced Emulator Technology, Pat. Pending) provides unmatched value. MetaLink also delivers leading edge customer service, including a 30-day money back guarantee, 10 day trial periods, rental plans and free technical support. Call today for FREE demo diskette.



(800) 638-2423

MetaLink
Corporation

MetaLink Corporation
325 E. Elliot Road, Chandler, AZ 85225
Phone: (602) 926-0797 Fax: (602) 926-1198

MetaLink Europe GmbH
Westring 2, 8011<85614>Kirchseeon-Egharting
Telefon (08091) 2046, Telefax (08091) 2386

Since LAs observe processor signals rather than replace the processor, they tend to be less expensive than ICEs. They are also more portable than ICEs because of their smaller size. In fact, an LA can be incorporated into a prototype system by adding a small printed circuit board which is eliminated when the system goes into production. Handheld units are becoming popular for field use. LAs have several disadvantages. As mentioned earlier, they do not support automatic translation from raw data to high-level language constructs. The programmer is required to manually map instruction and data addresses back to HLL statements and variables.

Second, LAs can only observe signals external to the target processor. When the processor has a data or instruction cache, internal processor activity can be hidden. Some processors, like the Motorola 68332, optionally force all bus activity to be visible on external pins to facilitate debugging with LAs.

Third, LAs load the system electrically. The probe capacitance is usually not significant. In a marginal system, however, a spurious signal that caused a problem can be filtered away by the addition of the probes.

OSCILLOSCOPE

The **oscilloscope** is the most useful tool in dealing with analog signals. It consists of one or more analog signal inputs, a display, and switches for specifying display parameters. Like LAs, scopes have some electrical effect on the system and do not stop program execution when capturing system behavior. Scopes are triggered by the voltage level and slope of the input signal. Slope is positive when the voltage is increasing with time and negative when the voltage is decreasing with time.

The oscilloscope is a very versatile RTECS debugging tool in skilled

hands. Here we will focus only on a few conventional uses of the tool related to software/hardware interaction. When used alone, the scope readings must be manually correlated to source code. Traditional oscilloscopes only work with DC or periodic signals. **Storage oscilloscopes** have memory and can be used to capture aperiodic analog events.

Scopes are useful in verifying hardware response to software. Wave-

showing the actual signal received so the programmer can verify the software action taken for the error.

Some signal noise is continuous because the source is continuous, such as a nearby radio station. Other noise occurs only from brief instants under certain circumstances. An example would be noise from starting an electric compressor motor which crashes a computer when the piston is in a certain position. The storage scope can be used to find and characterize the noise prior to implementing a hardware or software remedy.

Scopes are relatively inexpensive and commonly available in technician shops and hardware laboratories. The versatility of the tool is countered by the limited number of input channels, typically two, and the fact that only the raw

analog display is available.

CROSS-DOMAIN DEBUGGING

So far we've focused on the individual use of the hardware-intensive tools. We've highlighted their advantages and disadvantages. These tools can be used together for cross-domain measurements, meaning that an event detected by one tool triggers an observation on another tool as well as on itself.

On the most basic level, each tool has an external trigger output which generates a pulse when a trigger condition has been met. Each tool also has an external trigger input which will initiate a measurement when a pulse is received. Thus an analog voltage level on a scope can trigger an assembly language instruction trace on a logic analyzer, or a glitch detected by a logic analyzer can trigger an instruction trace by an emulator which is then mapped back to HLL source code.

Integrated tools are commercially available. Common pairings are a digital scope with a logic analyzer and a logic analyzer with an in-circuit emulator. These products have tighter

	Hardware Intensive			Software Intensive	
	CRO	LA	ICE	BDM	Monitor
Measurement Stimulus	Trigger	Trigger	BRK point/Trigger	BRK point	BRK point
System Execution	Continue	Continue	Stop/Continue	stop	stop
Intrusion	Physical	Physical	Temporal/Physical	Temporal	Temporal
Perspective	External	External	Internal/External	Internal/External	Internal/External

Figure 3—Almost all the debugging tool options intrude upon the target in some manner, whether it be at the hardware or the software level.

forms resulting from writes to D/A converters can be checked for correct shape, amplitude, and period. These checks are also required for pulse-width modulated (PWM) signals controlled by software. MCUs, such as the M68HC11, have on-chip PWM capabilities which are configured by writing values to control registers.

Scopes are useful in identifying problems with interrupts from external sources. Continuous noise on signal lines can be measured. So can the period of periodic interrupts. If the interrupt is from a source internal to the processor, the period can be measured by toggling an output pin upon entry and exit to the corresponding interrupt service routine.

Two applications of the storage scope are testing communications software, and finding problems caused by transient noise. Processor serial communications hardware, such as the SCI on the Motorola M68HC11, reports events such as frame overrun and noisy data to applications software through flag bits in a status register. A storage scope can capture the analog signal for corrupted test transmissions,

integration than their discrete counterparts resulting in easier use, lower cost, and reduced space.

MONITORS

A *monitor* is code residing on the target system, normally in ROM, that supports interaction between the programmer and the target system. A program runs at full speed until it reaches a breakpoint. At that time, the program stops and the monitor is executed. If the program skips over the breakpoint, the monitor never regains control. The advantage of always having the monitor present in the system must be balanced by the possibility of it not being usable when needed most. If the target hardware fails, the monitor may also become unavailable. Monitor functions are kept minimal to limit their use of target system memory. Typical functions include examination and

modification of registers and memory, downloading and execution of programs, setting and clearing breakpoints, and simple instruction assembly and disassembly. Stepping through a program, a machine instruction at a time, is usually supported. When single-stepping through a program, the user can intervene between any two instructions. With a suitable compiler and HLL debugger, these primitive operations can be used to support source language debugging from a software development workstation.

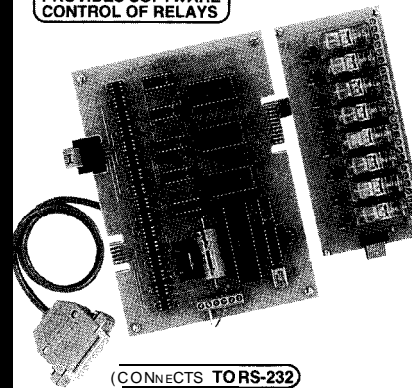
Monitors add no extra cost to an RTECS other than their development time and the memory they occupy in the system, typically less than 32K. This memory requirement may be significant for smaller embedded systems such as microcontrollers. Monitors require a means of communicating with the programmer,

Cross Development Feature	Monitor (MON)	Emulator (ICE)	Background Debugger (BDM)	Logic Analyzer (LA)	Scope (CRO)
HLL Support	yes	yes	yes	no	no
ASM Support	yes	yes	yes	yes	no
Load program into memory	yes	yes	yes	no	no
Start program execution	yes	yes	yes	no	no
Single step program	yes	yes	yes	no	no
Real-time instr. trace	no	yes	no	yes	no
Timing of instructions	no	yes	no	yes	no
Triggers on address and data values	no	yes	no	yes	no
Set BRK point	yes	yes	yes	no	no
Modify memory and registers	yes	yes	yes	no	no
Cost	low	high	low	medium	low
Software debugging	good	excellent	good	fair	poor
Hardware debugging	poor	excellent	poor	good	good

Figure 4—Comparison of debugging tool features.

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

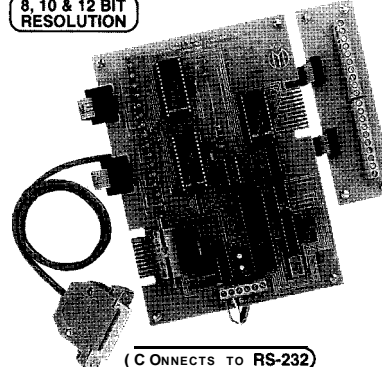


(CONNECTS TO RS-232)

AR-16 RELAY INTERFACE (16 channel) \$ 89.95
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)

A DC-16 A/D CONVERTER (16 channel/8 bit).... \$ 99.95
A DC-8G A/D CONVERTER (8 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 A/D configurations and 12 bit c onverters (terminal block and cable sold separately).
A DC-8E TEMPERATURE INTERFACE (8 ch).... \$ 139.95
Includes term. block & 8 temp. sensors (-40' to 146' F).
STA-8 DIGITAL INTERFACE (8 channel)..... \$ 99.95
Input on/off status of relays, switches, HVAC equipment, security devices, smoke detectors, and other devices.
STA-8D TOUCH TONE INTERFACE..... \$ 134.95
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-485 (RS-232 to RS-422/RS-485 converter).... \$ 44.95

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

* **FULL TECHNICAL SUPPORT**...provided over the telephone by our staff Technical reference & disk including test software & programming examples in 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 (614) 464-4470.

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

International & Domestic FAX (614) 464-9656
Use for information, technical support & orders.

ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 604
Columbus, Ohio 43215-5438

typically through a target system serial port. This can be a problem if all serial ports are being used by the application.

RTECSs with critical timing requirements and interrupts may not operate properly with simple monitors. This results from the processor cycles used by the monitor when invoked and violation of design assumptions when the program is not running at normal speed. For example, suppose a breakpoint is placed in a reentrant interrupt service routine. The interrupt occurs every 10 ms and the service routine takes 100 μ s to execute. Under normal execution, there is no problem. If a breakpoint is placed in the service routine, the routine will not be completed by the time the next interrupt occurs. Repeated interrupts will eventually cause a stack overflow-or worse.

One solution to the interrupt problem is to run the monitor as a separate, low-priority task. Programmer activity does not delay the service of time-critical interrupts. This requires the presence of a debugged task manager on the target system and

its associated memory and time overhead. The approach is not appropriate for every situation, particularly if the task manager is part of the software under development.

BACKGROUND DEBUG MONITOR

A *background debug monitor*, borrows features from both the in-circuit emulator and the monitor. It acts like a very simple monitor except that it is integrated into the processor and requires no memory or system I/O when operating. Communication with the user is done through dedicated pins on the processor chip. Breakpoint addresses may be held on-chip in *breakpoint registers* or require minimal external hardware. Tracing is not supported. Thus the BDM is temporally intrusive because it and the application program cannot be executing simultaneously.

The BDM in the Motorola CPU32 core processor for the 683xx family, is implemented in microcode and uses five pins on the chip for user interaction. By being available on every chip, it easily supports field debugging.

Fourteen simple commands permit the reading and writing of memory and registers, program execution, and I/O reinitialization. These commands can be used by debug software on a host development workstation to support full-featured HLL debugging while only requiring four bytes of the system memory. The monitor can be viewed as an extension to the processor instruction set, except that the instructions are fetched from an external source rather than memory.

IN CONCLUSION

We've described six tools for RTECS software/hardware debugging. An ideal tool would support debugging from the application abstraction all the way down to the physical signal voltages. In practice, as summarized in Figure 4, this is not the case.

Secondly, an ideal tool does not intrude upon the system being observed. Real tools do impact system behavior electrically, temporally, and spatially. There are also great differences in the costs of the tools.

Finally, a tool is only as good as the hand that wields it. The user must select the right tool for the job and use it properly. Tool selection depends on the perceived problems to be solved, the exact features of tools available for the system software and hardware, the money available, the time available, and the background of the person(s) doing the debugging. May your integration efforts be worthy of Norm and not Homer. 📌

Noel Anderson is an assistant professor in the Electrical Engineering Department at North Dakota State University and a computer engineer with Concord Inc., both in Fargo, N.Dak. He may be reached at nanders@plains.nodak.edu.

Dean Hoyt is a computer engineer with 3M in St. Paul, Minn.

Ron Shaw is a computer engineer with E.F. Johnson in Waseca, Minn.

Real-Time Multitasking with DOS for Microsoft C, Borland C, Borland/Turbo Pascal

Develop Real-Time Multitasking Applications under MS-DOS with **RTKernel!**

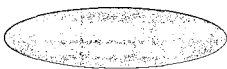
RTKernel is a professional, high-performance real-time multitasking kernel. It runs under MS-DOS or in ROM and supports Microsoft C, Borland C, Borland/Turbo Pascal, and Stony Brook Pascal+. **RTKernel** is a library you can link to your application. It lets you run several C functions or Pascal procedures as parallel tasks. **RTKernel** offers the following advanced features:

- preemptive, event/interrupt-driven scheduling
- number of tasks only limited by available RAM
- task-switch time of approx. 6 μ secs (33-MHz 486)
- performance is independent of the number of tasks
- use up to 64 priorities to control your tasks
- priorities changeable at run-time
- time-slicing can be activated
- programmable timer interrupt rate (0.1 to 55 ms)
- high-resolution timer for time measurement (1 μ sec)
- activate or suspend tasks out of interrupt handlers
- programmable interrupt priorities
- supports math coprocessor and emulator
- semaphores, mailboxes, and message-passing
- keyboard, hard disk, and floppy disk idle times usable by other tasks
- interrupt handlers for keyboard, COM ports, and network interrupts included with source code
- supports up to 36 COM ports (DigiBoard, Hostess boards)
- full support of NS16550 UART chip
- fast, inter-network communication using Novell's IPX
- runs under MS-DOS 3.0 to 6.0, DR-DOS, LANs, or without operating system
- DOS calls from several tasks without re-entrance problems
- supports resident multi-tasking applications (TSRs)
- runs Windows or DOS Extenders as a task
- supports CodeView and Turbo Debugger
- ROMable
- full source code available
- no run-time royalties
- free technical support by phone or fax

RTKernel-C (MSC 6.0/7.0/8.0, BC++ 1.0/2.0/3.x) \$495 (Source Code: add \$445)

RTKernel-Pascal (TP/BP 5.x/6.0/7.0, SBP 6.x) \$445 (Source Code: add \$375)

For international orders, add \$30 for shipping and handling. Mastercard, Visa, check, bank transfer, or COD accepted.



OnTime
MARKETING

Professional Programming Tools

In North America, please contact:

LEL Computer Systems
20 Canterbury Court
Setauket, NY 11733 • USA
Phone (516) 473-8119 • Fax (516) 331-0706

Outside North America, please contact:

On Time Marketing
Karolinenstrasse 32 • 20357 Hamburg • GERMANY
Phone +49 - 40 - 43 74 72 • Fax +49 - 40 - 43 51 96
CompuServe 100140.633



I R S

- 401 Very Useful
- 402 Moderately Useful
- 403 Not Useful

Inexpensive 68HC11 Cross- development

Think you need all kinds of fancy cross-development tools, debuggers, and elaborate boards to create 68HC11 code? There are cheaper alternatives, and Bruce is only too happy to share one with you.

FEATURE ARTICLE

Bruce L. Olney



I am an associate and I work together designing embedded control projects. Recently we finished a cross-development system for Motorola M68HC11MCUs. We use it as the launching pad for almost all of the embedded control designs that we create. Being interested in code reuse, we decided to extend the concept to hardware and make it available to others.

The development system consists of two parts: a PCA that is not unlike the microcontroller evaluation boards available from Motorola, and a set of ROM routines that work with PC software. This development system aids in developing and debugging firmware and hardware quickly and inexpensively.

We chose the Motorola MC68HC711D3 MCU, but the ROM routines and software could very easily be adapted to other processors. It will be particularly easy to adapt them to work on other M68HC11MCUs.

PCA FEATURES

Our development platform uses a Motorola MC68HC711D3 processor and can use 8K or 32K SRAMs mapped either high or low. For I/O, the system offers eight decoded strobes that are also mappable either high or low. This platform can also function as an MC68HC711D3 EPROM programmer. It was designed to allow the user to quickly prototype expansion hardware by providing a processor interface at an expansion connector. The debugger and download utilities we provide are designed to help get your firmware up and running quickly. The supply

requirements call for a simple +5- or +12-volt operation. (+12 V is required for EPROM programming)

There are several reasons why we chose the MC68HC711D3 processor for this project. This MCU is one of the least expensive members of Motorola's OTP family, and they are relatively easy to get. For those of you who don't have anywhere else to turn, I have made arrangements to make a quantity of these chips available via mail order. Look for ordering information at the end of the article.

Another reason for choosing the MC68HC711D3 is the availability of low-cost development tools. Motorola provides free cross-assemblers on the Motorola Freeware BBS (512/891-3733). Every OTP MCU has its own EPROM programmer built in, so there is no need to spend money on development tools and EPROM programmers for these devices.

The final reason is the versatility of the device. It can be used for a broad range of projects, from tiny to large. For a small design, it would be tempting to use some sort of M68HC05 MCU, but the MC68HC711D3 could be chosen as well. Since it is less expensive than many M68HC05s, why not use it instead? For larger designs that require an expanded bus or have complex software needs, the MC68HC711D3 works great. Standardizing on one MCU where possible keeps one from constantly rewriting library routines due to a different instruction set or hardware.

For projects where the MC68HC711D3 doesn't quite fit, there are hundreds of other Motorola MCUs that contain similar peripherals. If you know the MC68HC711D3, the learning curve is greatly reduced.

We created this board to fulfill three purposes. First, it serves as a platform to quickly build up new hardware prototypes. Second, it was designed to be a software development system that is easy to use. Third, the board functions as a low-cost, but effective and efficient, EPROM programmer for MC68HC711D3 parts.

Before we created this tool, getting a project going required several days' work simply re-creating things that are

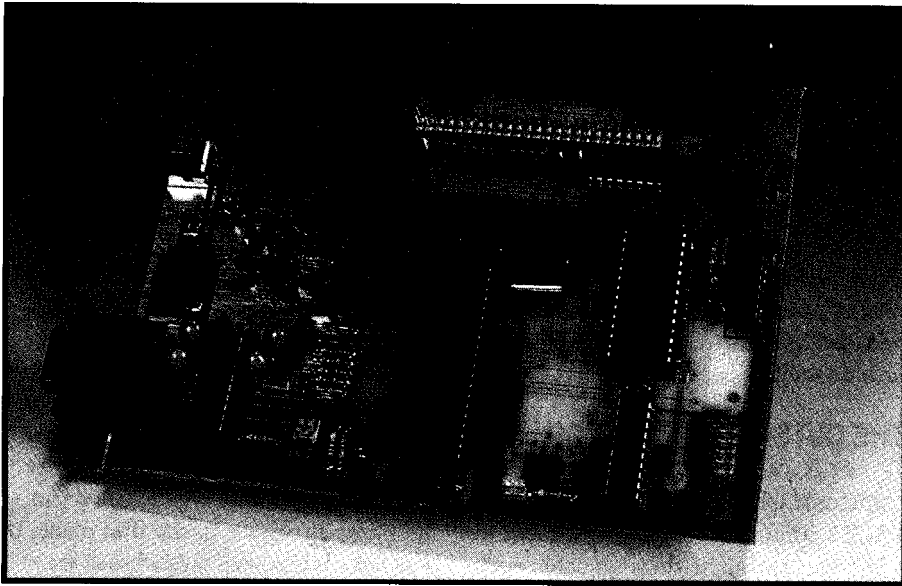


Photo 1—The MC68HC11 development system's cross-development capability will help any engineer save time when working on a project.

now on this board. We started wondering why we were spending so much time reinventing the wheel. Most of the things we needed for any given project are included in this design. To use this design for a new project, just hook up the I/O that's unique and write the firmware for it.

To fulfill the function of a software development tool, the board and accompanying software provide various methods to download code to the board and debug it. Each has a particular purpose.

Most often I use RUN11.EXE to download a program to the external RAM and execute it. If I am having trouble with a bug that I can't figure out, I use DEBUG11.EXE, which is a window-oriented debugger like Codeview or Turbo Debugger, but doesn't have as many features. You can use it to load a program, single step through it, set break points, inspect variables, and so forth.

After designing and debugging the code using RUN11.EXE and DEBUG11.EXE, the finished, debugged code can be programmed into the EPROM of a MC68HC711D3 using the RUN 11. EXE utility. The EPROM programming feature of RUN 11. EXE requires that the MODA/MODB jumpers be set to bootstrap mode or it will not work. DEBUG11.EXE and the normal download mode of RUN 11. EXE require them to be set to single-chip mode.

DEVELOPMENT BOARD CONNECTORS, OPTIONS, AND INDICATORS

J1 is a standard coaxial connector used for power in many applications. Power may be either regulated +5 V or unregulated +8–18 V. Set JP4 appropriately for the voltage you are supplying to the board. Keep in mind that in order to use the board as an EPROM programmer, you must supply at least +12 V. A standard wall transformer available from Digi-Key can be used. In fact, the Digi-Key T506-ND+12V will work nicely. The Digi-Key part number for the coaxial connector mating with the connector on the board is CP-003A-ND. You may wish to use regulated +5 volts instead of +8–18 V. If you power the board with a regulated +5-V source, the LM7805 regulator is not required. JP4 must be moved to the +5 position, which bypasses the regulator.

LED D2 is the V_{pp} programming power indicator and will be turned on by the bootstrap mode firmware when V_{pp} is applied to the MCU. Q1 and Q2 act as switches and allow V_{pp} to be applied under the bootstrap mode firmware control. When bit 5 of port D is set by the firmware, Q1 and Q2 supply V_{pp} to the MCU. A zener diode (D1) keeps V_{pp} from going over +12 volts when using unregulated DC.

A 7.3728MHz crystal was selected because it allows a 115,200-

bps serial communications link. If an application requires a different crystal frequency, you will have to change the baud rate setting in the bootstrap firmware. Without changing the firmware, the bit rate for an 8-MHz crystal will be 125,000 bps, which is unsupported by PC serial ports.

RUN11.EXE and DEBUG11.EXE support nearly any bit rate available on the PC. The run-line option switch (invoked by -baud:xxx) can be used with both to change the bit rate.

U4 holds the MC68HC711D3 processor. This is also the programming socket when using the board as a MC68HC711D3 programmer.

JP3 sets the state of the MODA pin. There should almost always be a jumper in this position. Removing the jumper provides an expanded multiplexed mode or a special test mode. Occasionally you may need to set the board into expanded multiplexed mode (e.g., when you have to add an external EPROM). I doubt you will ever use special test mode since it is for Motorola's manufacturing processes.

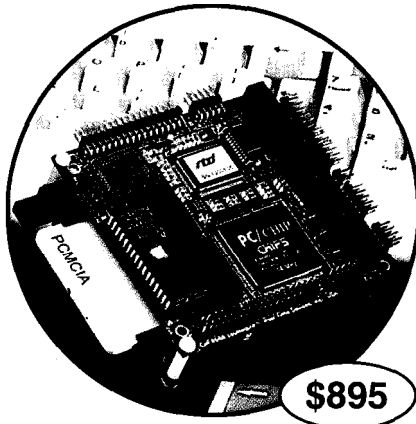
JP2 sets the state of the MODB pin. A jumper shorts MODB to ground; removing the jumper allows MODB to be pulled up to +5 V by a pull-up resistor. Most of the time the jumper will be removed to put the board in single-chip mode, thereby running the debugger firmware in the internal EPROM. Replacing the jumper puts the board in HC11 bootstrap mode, which must be selected when using the development system as an MC68HC711D3 EPROM programmer.

U2 may contain either an 8Kx8 or a 32Kx8 SRAM. JP1 selects whether RAM will reside above or below \$8000 in the memory map. The I/O page will go in the opposite area. Specifically, if RAM is set high, then I/O will be set low and visa versa. If an 8K SRAM is used, it will appear at \$2000–\$3FFF if RAM is set low and at \$A000–\$BFFF if RAM is set high. Also, when using a 32K SRAM, if it overlaps an on-chip peripheral within the HC 11, the peripheral will have priority and will be accessed instead of the SRAM.

The 74HC138 at U5 provides eight decoded I/O chip selects for external devices. If JP1 is set for RAM high, the

The New Shape of Embedded PCs

The amazing CMF8680
cpuModule™ is the first complete
100% PC-compatible
PC/104 single board computer
measuring only 3.6" by 3.8"!



- 16-bit, 14 MHz PC/Chip™
- CGA/LCD controller
- 2M DRAM
- ROM-DOS kernel
- bootable 1 M solid-state disk
- configuration EEPROM
- 1 B-bit IDE controller & floppy interface
- PCMCIA interface
- two RS-232, one RS-485 & parallel port
- XT keyboard & speaker port
- watchdog timer
- +5 volts only operation

*Designed for low power applications, the CMF8680 draws one watt of power, which drops to 350 milliwatts in sleep mode, 125 milliwatts in suspend mode. Free **utility** software lets your application boot from ROM!*

RTD also offers a complete line of PC/104 peripherals for expansion:

- 1.8" hard drive & PCMCIA carriers
- 12- & 14-bit data acquisition modules
- opto-22 & digital I/O modules
- VGA CRT/LCD interface

*For more information:
call, write or fax us today!*

*Place your order now and receive a
CM102 PCMCIA carrier module
FREE!*



Real Time Devices, Inc.

P. O. Box 906

State College, PA 16804

(814) 234-0887 ■ Fax: (814) 234-5218

decoded addresses will be \$2000, \$2400, \$2800, \$2C00, \$3000, \$3400, \$3800, and \$3C00. Setting JP1 for RAM low provides decoded addresses at \$A000, \$A400, \$A800, \$AC00, \$B000, \$B400, \$B800, and \$BC00.

All of the I/O pins from the MC68HC711D3 appear on connectors JP5 and JP6. In addition, the demultiplexed address/data bus and the eight decoded I/O chip selects are provided here as well.

THE OUTSIDE CONNECTION

P1 is a standard 9-pin D-type connector which is set up to support KS-232 communication. To connect to a PC, use a straight-through cable such as Radio Shack catalog number 26-117.

The RTS line is used to reset the HC11 board. A lot of serial communication libraries for C and Pascal wiggle the RTS line when initializing the serial ports. If you download some software to the HC11 board using RUN 11. EXE and then wish to communicate with that software using a program written with one of these communication libraries, the communication library will wiggle the RTS line and reset the board. The software you downloaded will no longer be running. To avoid this, either use a communication library that doesn't wiggle the RTS line, or download the software to the board within your own program rather than using RUN 11. EXE. It'll cover the communication protocol used to do this later. It is the same protocol that RUN 11. EXE uses. Downloading the firmware within the application that communicates with the board saves you from using two different utilities.

THE DEBUGGER FIRMWARE

The debugger firmware for this project deserves a separate article entirely. A lot of the embedded projects I design are connected to a PC via a serial port. Some portion of the software system runs the PC and communicates with the embedded firmware through the serial port. This naturally leads to a system where the application on the PC would download firmware to RAM on the slave. This minimizes the need to replace

EPROMs in the slave when adding or changing features in the system. Using this method, we upgrade portions of the firmware by releasing a new application for the PC rather than burning new EPROMs.

The next logical step is to have no EPROMs at all in the slave and have it run entirely with downloaded firmware. At first we tried using the bootstrap mode of the MC68HC11 to do this, which worked adequately, but had some serious drawbacks such as a low bit rate, and the limited protocol was prone to error. Minimizing those drawbacks was why we designed the bootstrap firmware in the first place. It was later when we realized that we could create an excellent embedded debugger by adding a few additional commands. I feel this is an excellent example of how to implement a debugger through a serial link. This is a very useful technique for more than just embedded control projects.

The design philosophy used for the firmware is "less is more." It is better to do most processing on the PC side and keep it simple on the HC11 side. Any downloading or debugger task can be implemented with the following seven basic operations:

- Reset the system to get it into a known state.
- *Write data to memory.
- Read the contents of memory.
- *Read the state of the registers.
- Set the registers.
- Cause the system to begin executing the program.
- *Provide some method to notify the PC when it has stopped executing, usually because a breakpoint was reached.

Our design task was to create bootstrap loader firmware that supplied these essential operations, but it had to fulfill one other requirement: it had to be as fast as possible. One of the drawbacks of the built-in bootstrap mode of the HC11 is that the bit rate and communication protocol caused it to be somewhat slow.

We solved speed requirement in two ways. First, we selected a crystal frequency that allowed the HC11's

serial hardware to operate at 115,200 bps, the highest bit rate available on a PC. And second, we defined a communication protocol that would allow operation at this speed.

THE COMMUNICATION PROTOCOL

The communication protocol used operates using a master/slave relationship. Only the master may initiate a packet. After sending a packet to a slave, the master becomes a temporary slave and must wait for the "slave" to send an acknowledgment before the "master" can send another. After reset, the HC11 comes up as a slave and the PC is normally the master.

To initiate a packet, the master sends a single byte (a length byte) to the slave. This byte signifies the number of bytes remaining to be sent in the packet. If the length byte is zero, it signifies that 256 bytes will be sent.

When the slave has everything ready to receive the data bytes (it may need to stop what it is doing-disable interrupts, etc.-to keep from dropping

bytes) it will echo the length byte back to the master, signifying that it is ready for the packet burst. After receiving the echo, the master is free to burst the rest of the bytes at full speed. The slave is required to be ready for them at full speed.

That is all there is to it. Packets are defined by the firmware to provide the essential functions that were listed above. Refer to the source code for more detailed information. Other applications could define other packets to perform operations as needed.

SOURCE FILE SPELUNKING

The source files are too long to be printed in the pages of the magazine, so you'll need to download them to follow this discussion. See the end of this article for more information on how to download these files.

The source code for the debugger project is divided into four distinct modules: **BIGMAMA**, **DEBUG11**, **RX-PACKET**, and **VECTORS**. Each module consists of an include file (with the extension **.INC**) and an assembly

language source file (with the extension **.ASM**). The include files specify the items exported from the module and the assembly files define the implementation of the exported items. Additional support files are **R E G S - 11D3.INC** and **LOWCOST.ASM**.

A word about the **B I G M A M A** filename. My partner wrote **RUN 11 . EXE** and **DEBUG11 . EXE**. I wrote the **B I G M A M A** firmware. I had to explain to him how the firmware worked. I also explained the differences between the enhanced bootstrap firmware that I designed and the special bootstrap mode built into the MC68HC711D3 silicon. In the explanation I called the special bootstrap mode of the MC-68HC711D3 "little mama." Naturally then, I called the enhanced firmware "big mama," and the name stuck. It connotes that the firmware is really just an enhanced bootstrap mode. More correctly, it is analogous to the Background Debug Mode (BDM) of the Motorola HC 16 and CPU32 MCUs. If BDM were similarly implemented in the HC11 family, this project wouldn't be necessary.

The **B I G M A M A** module contains the entry point and the main routine for the program. There are three items exported from the module. **Entry Point** is exported so the **VECTORS** module can make it the target of the power on reset vector. **Bi gMama** is exported so that the **DEBUG 11** module can jump to it after processing a breakpoint or illegal interrupt.

The main routine of the **B I G M A M A** module uses the **RX PAC KET** module to communicate with the PC. It interprets the meaning of the packets received and responds accordingly.

The **VECTORS** module exports 24 items, which may sound like a lot, but it isn't really. There are only three kinds of things exported.

First is the constant **J M P _ E X T E N D E D**. This symbolic constant represents the value of the extended jump opcode, which is needed if a user program wants to redirect any of the interrupt vectors. In this project, all of the interrupt vectors have to be defined in the EPROM. Since the EPROM cannot be changed, in order to allow a user program to use interrupts,

AVOCET SYSTEMS, INC.

C and Pascal Compilers
Simulators
Assemblers
In-Circuit Emulators
EPROM Programmers

Intel
8051
8048
8085
8096

Hitachi
64180
H8

Western
65816

Rockwell
6500
6502

RCA
1802

Motorola
680x0
68300
68HC11
6800
6801
6802
6803
6804
6805
6809

Zilog
Z8
Z80
Z180

TI
32010
32020

Source For Quality Embedded Systems
100 Main Street, P.O. Box 490, Rockport, ME 04856
For outside U.S., call (207) 236-9055
FAX: (207) 236-6713 • FAX: (207) 236-6713

Call today:
1-800-448-8500

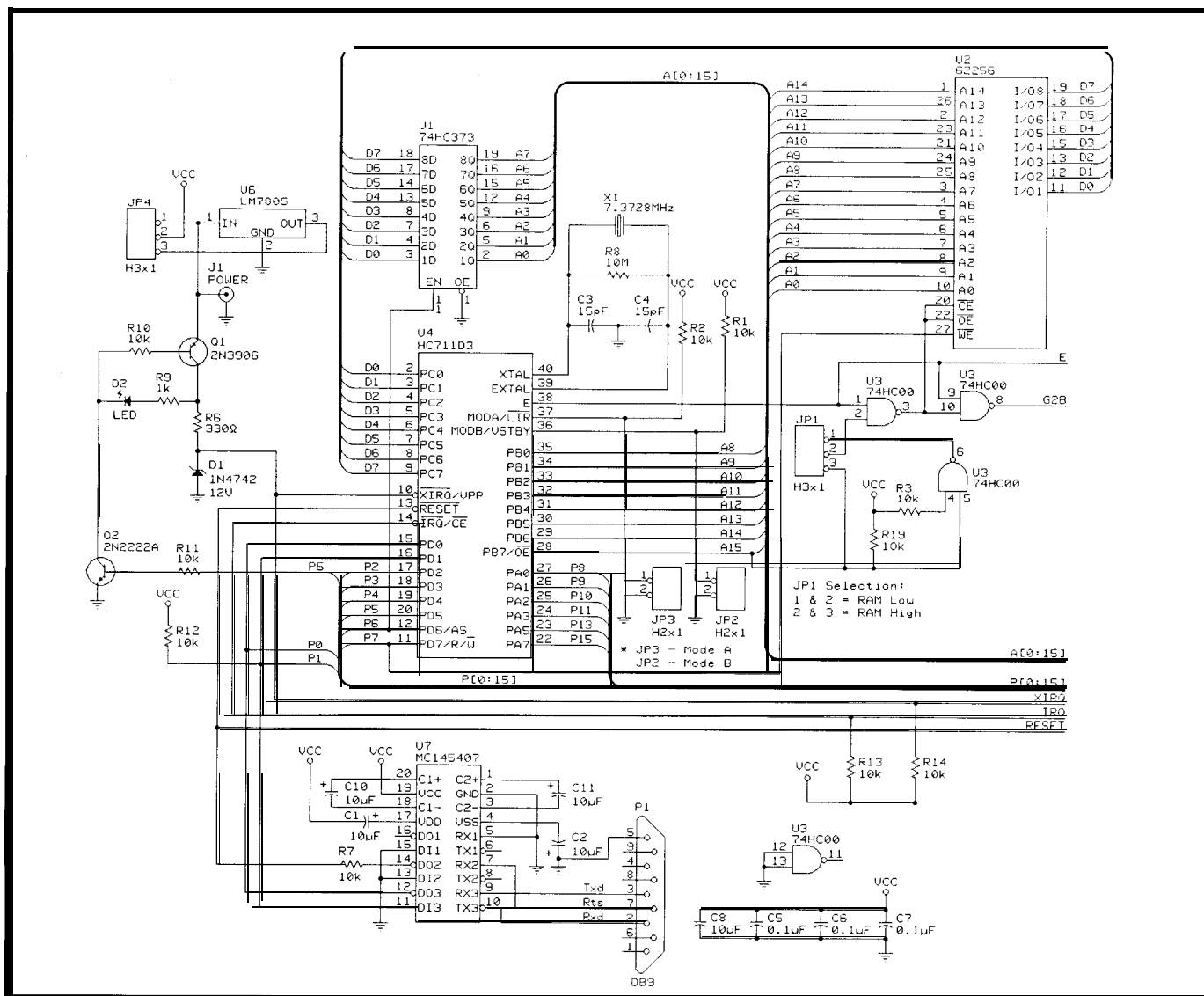


Figure 1a--The MC68HC11 development system is based on the MC68HC711D3 EPROM-programmable processor.

the interrupt vectors are pointed into the internal RAM. The user program that needs to use an interrupt stores the value of `JMP_EXTENDED` into the appropriate location in internal RAM followed by the address of the interrupt service routine to be called when the interrupt occurs. This is exactly what the `DEBUG1 1` module does.

The next two symbols (`RAM VectorsStart` and `FreeRamStart`) mark the locations in internal memory where the interrupt jump table starts and ends. Internal RAM beginning at `FreeRamStart` is available for downloaded programs to use. The memory locations used by an interrupt that is disabled may also be used by downloaded programs.

The last 21 symbols are the locations in internal memory where

the corresponding interrupt vector will jump to if the interrupt is generated.

The `RX PACKET` module provides a software interface to the communication protocol. A perfect implementation would have this module know nothing about the contents of the packets. The meaning of the packets received and sent would only be understood and handled by the module(s) using the services of the `RX PACKET` module. In practice though, in order to keep the packet buffer small (13 bytes), the `RXPACKET` module looks at the contents of the packets being received and sent to see if it is a `SET-MEMORY` or `GET-MEMORY` packet. If it is one of these packets, then the data portion of the packet will be taken from (or written to) the address specified by the packet.

Without this, the user of the module would have to have a packet buffer of 256 bytes. This is too large to fit in the internal RAM of the MC68HC711D3. Also, when a `SET-MEMORY` or `GET-MEMORY` packet is sent or received, the user of the `RX PACKET` module would have to copy data around in memory.

Downloaded user programs may make use of the services provided by the `RX PACKET` module by calling the interface routines and taking over the packet buffer. They can even create new packet types and assume new uses for the existing ones except for `SET-MEMORY` and `GET-MEMORY` that are handled by the `RXPACKET` module.

The `DEBUG1 1` module handles the debugger features of the firmware. It exports a couple of items. The first item is the variable called `process`

processorState. This variable is a data structure that contains the processor state of the code being debugged. The assembler we used doesn't have any special directives to directly support data structures, so the next items in the `DEBUG11.INC` file are the offsets to the individual fields within the processorState datastructure. A routine can access the individual members of this structure by loading an index register with the address of processorState and use the indexed addressing mode. For example, to load the A register with the contents of the ACCA member of the processorState data structure, the following code fragment could be used:

```
LDX #processorState
LDAA X, ACCA
```

The assembler doesn't offer full structure support, but the HC11's machine language does it quite well.

The final item exported by the `DEBUG11` module is the address of the `InitializeDebugger` routine. One of the first items performed by the `BIGMAMA` module on power-up is to call the initialization function exported by the `DEBUG11` module.

The `InitializeDebugger` function captures all the interrupt vectors and points them to default handlers. The most important interrupt vector is the SWI interrupt vector. The SWI instruction is used as a breakpoint instruction and is mapped to the debugger code. If an SWI instruction is executed within the user's program, the instruction causes the processor state to be pushed onto the stack and control transfers to the nonexported `Debugger` routine. This routine copies the processor state from the stack to the processorState structure and it jumps to the `BigMama` routine (in `BIGMAMA.ASM`). `BigMama` will communicate to the PC that a breakpoint has been reached and execution has stopped.

Any other interrupt is an error condition and causes the debugger to be entered and the status reported to the PC by `BigMama`.

The `REGS11D3.INC` include file isn't a header file for a software

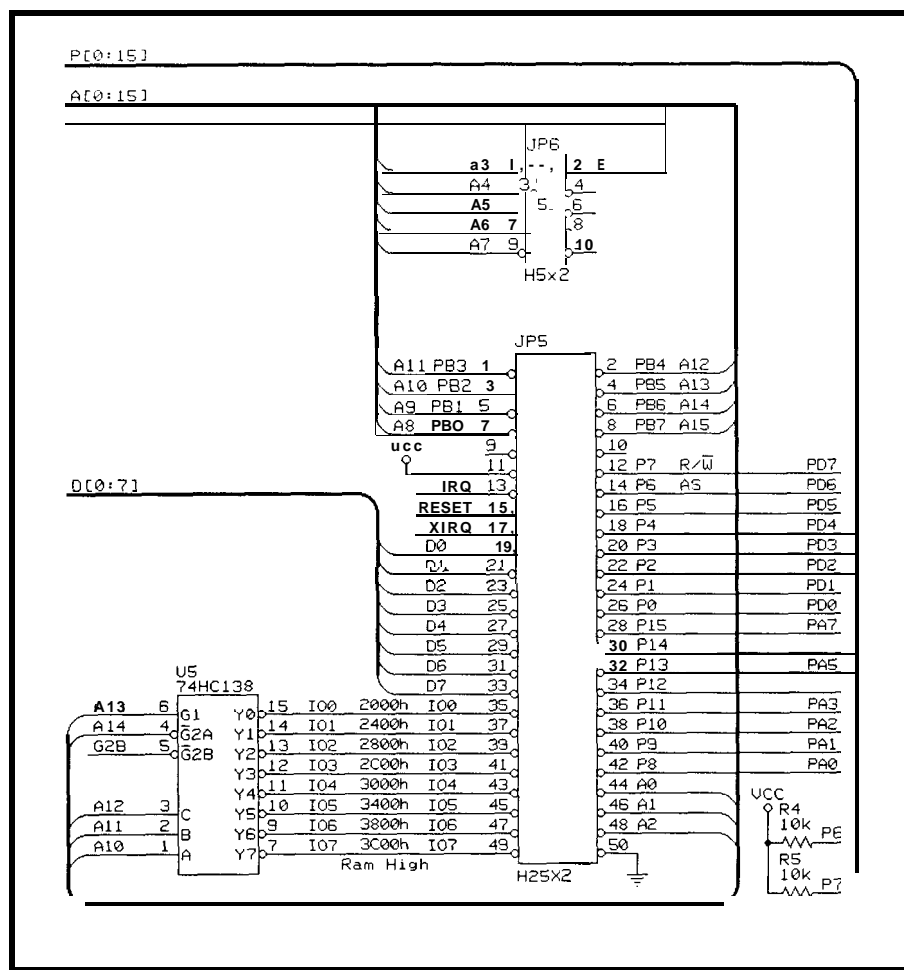


Figure 1b—The MC68HC11 development system also includes an expansion header to make customizing it to your project easy.

module, rather it defines symbols to interface with all the hardware registers, bits, and so forth, for the MC68HC711D3. Any program I write for the MC68HC711D3 can include this file and then access the registers and bits within the MC68HC711D3 using the symbols defined in the data sheet. Note that register names are in lower case and bit names are in upper case. My symbol naming convention is that upper-case symbols are reserved for symbolic constants. Symbols using a mix of upper and lower case with the first character of the symbol being lower case are reserved for variables. I consider registers to be variables and bit names to be constants. Symbols using a mix of upper and lower case with the first character of the symbol being upper case are reserved for code address labels.

This firmware project was written to work with either of two assemblers: the EveryWare Professional assembler

or the EveryWare Low-cost assembler. The EveryWare Professional assembler is a relocatable, linkable assembler that creates and supports multiple compiled modules. The EveryWare Low-cost assembler doesn't support multiple modules other than through the `INCLUDE` directive.

The `RUN11.EXE` and `DEBUG11.EXE` utilities are available from the Circuit Cellar BBS, the Motorola Freeware BBS, and from EveryWare.

CONCLUSION

Reusability is a powerful tool that can improve engineering productivity. I think this cross-development system is an outstanding example of multiple layers of reusability. It has saved me countless hours over the course of many projects. I am able to finish more projects in less time, which means I can take on and finish more projects. It is my hope that this article will help you do the same. □

Bruce Olney has been involved with embedded systems since 1984. He is employed by Smart Disk Security Corporation as a technical consultant specializing in embedded systems design. He holds a B.S. in Computer Science from Weber State University in Ogden, Utah. He may be reached at bruce.olney@emc2-tao.fisc.com.

SOURCE

The source code and software described in this article are available on the Circuit Cellar BBS or by sending \$5.00 to:

EveryWare
229 W. Clark Street
P.O. Box 784
Grantsville, UT 84029
(801) 884-5050

Specify disk size when ordering. The following additional items are also available from EveryWare:

Blank PCB-\$19.95
Kit of all parts needed (less the bare PCB). Includes 32K SRAM and one MC68HC711D3S programmed with BIGMAMA firmware-\$59.95
MC68HC711D3S blank or programmed with BIGMAMA firmware-\$19.95
MC68HC711D3 databook-\$3.00
M68HC11 reference manual—\$8.00
EveryWare Low cost cross assembler for HC11—\$49.95
12-volt wall transformer-\$9.00
The Works (assembled and tested PCA, 12-volt wall transformer, software, M68HC11 data, etc.) is \$99.95

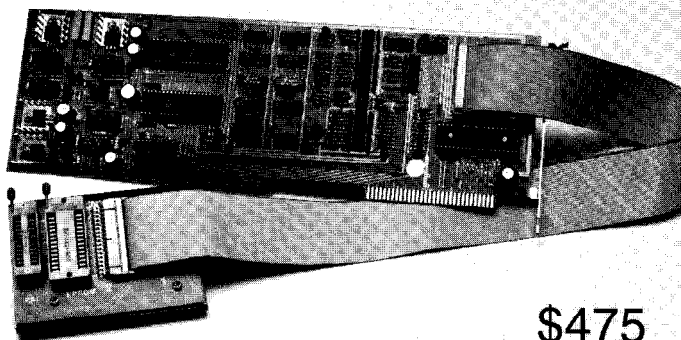
EveryWare carries many other items; call or write the address listed above for a current price list. Add \$4.95 to all orders for shipping and handling. Credit cards accepted.

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
87C51
PIC
93C46
XC1 736
PSD 3xx
5ns PALs

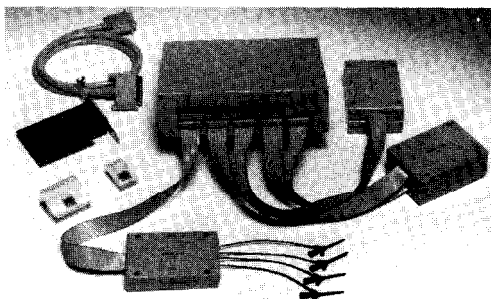


\$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

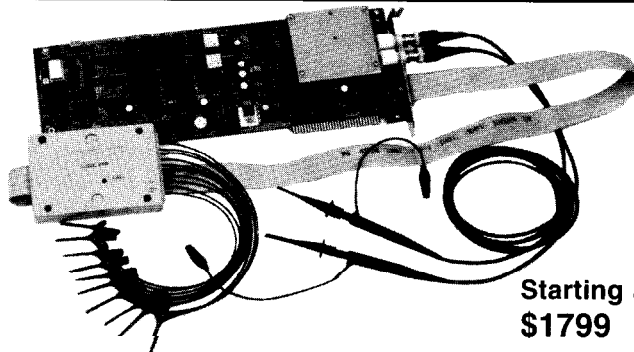
- up to 128 Channels
- up to 400 MHz
- up to 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



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

Price is Complete
Pods and Software
included

200 MSa/s DIGITAL OSCILLOSCOPE



Starting at \$1799 With Pods & Software

- 200 MSa/s sampling Rate
- 2 Analog Channels (2 ch. Digital Oscilloscope)
- 8 Digital Channels (8 ch. Logic Analyzer)
- 125 MHz Single Shot Bandwidth
- up to 128K Samples/Channel

Call (201) 808-8990



Link Instruments

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

Computer "Train"ing

FEATURE ARTICLE

Michael Smith

Using commercial evaluation boards in teaching

Manufacturer evaluation boards aren't restricted to the domain of the working engineer. They also make inexpensive trainers for up-and-coming engineers. Check out one board and how it is being used.

O teaching a project-oriented laboratory course to third- and fourth-year students in an electrical and computer engineering program has many pleasant aspects. By this point in their program, most students realize that the "cookbook" approach to laboratories is not going to give them the edge when they apply for jobs in industry. Take the students off the leash, give them the go-ahead to tackle an interfacing project of their own choosing, and just watch the enthusiasm and wide range of topics that they undertake.

However, there is an old saying: *Good judgement comes from experience. Experience comes from bad judgement.*

This gets reflected in what happens in the laboratory. The students know enough to conceive sophisticated problems, yet it is important to temper their enthusiasm since they lack experience. In addition, there is a limited amount of time available, although instructors like me prefer to think that ours is the only course the students are taking.

My laboratory projects are made more difficult because they are being used to give students assembly language programming experience.

Since the conception of the computer engineering minor program at the University of Calgary in 1981, interfacing has been done with custom peripherals connected to a 2-MHz Motorola M6809 processor in a student workstation. This generic (or perhaps geriatric) complex instruction set (CISC) microprocessor may simplify some of the problems with teaching assembly language programming. However, it does not provide the students with experience with the more recent processor advances such as register windows on reduced instruction set (RISC) processors. Nor does it provide experience with the pipelining found in current RISC and CISC processors.

Keeping up with technology in the microprocessor area is a continual problem. University budgets are decreasing and the funds remaining must be spread over many courses. Very little hands-on experience is gained when there are more than two students sharing each workstation. This means each upgrade must be purchased for each station in the laboratory, and not for just a few.

In an attempt to upgrade at low cost, we have been investigating the use of the commercial Motorola M68332EVS CISC and Advanced Micro Devices Am29200 RISC microcontroller evaluation boards. These boards are intended to provide an inexpensive method for people buying a manufacturer's software/hardware to try out their ideas on a ready-to-run board. However, the boards look ideal for keeping the experimenter and university laboratory up-to-date. There is additional savings when these systems are purchased without the full "official" C compiler and assembler.

In addition to low cost, these two boards offer a number of other advantages. The MC68332 and Am29200 microcontrollers are very good for laboratory projects as they include many features internally that require additional external devices on other processors. For example on the Am29200 microcontroller, there is a serial port, a printer "video" port, a JTAG interface, and peripheral

interface adapter (PIA) strobe lines for attaching devices to the processor buses. The major advantage of these boards, however, is the inclusion of the easily manipulated 16-pin programmable input/output (PIO) port which can be used as a "slow-as-you-like" bus, very suitable for the experimenter and student project.

The purpose of this article is to give an overview of the low-end Advanced Micro Devices SA-29200 evaluation board (Photo 1) in the context of an actual embedded computer "train"ing application. This project was initially introduced more as an interest booster for my students than with any real "computing" in mind. However when examined in earnest, the possible depth and advantages of the project became very apparent.

Suitable interfacing projects for the experimenter/student engineer are normally boring or expensive. By contrast, an electric train (Photo 2) comes in at around \$50 and provides a very visual feedback on the correctness of programming. Control of just a single train requires manipulation of steady currents (train speed), transient currents (track turnouts/switches), digital switching (signals), and position sensors. This provides control over virtually all of the signals that your typical embedded processor uses.

The currents involved can rise up to several amperes. Isolate a number of track sections from each other, and it is possible to control several trains by one processor and it requires development of an operating system with semaphores and the like. Control each train with a different processor and you open opportunities for multiprocessor communications.

A further advantage of the SA-29200 system is that it exposes students to a RISC processor, something with which many are unfamiliar. However, many of the comments made with respect to the SA-29200 system, and the projects suggested, are equally applicable to other evaluation boards.

RISC EVALUATION BOARDS

Students are well aware of the Intel 80x86 or the Motorola 680x0 CISC processor present in their ISA (Industry Standard Architecture) or Mac PCs. However, many students perceive assembly language programming of CISC processors as difficult because the wide variety of addressing modes and processor characteristics causes confusion. Although not without their own programming quirks, the simple LOAD/STORE

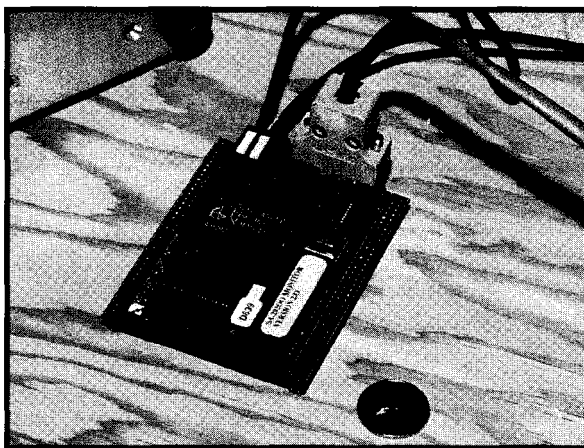


Photo 1—The Advanced Micro Devices SA-29200 board requires a 5--V power supply and a serial connection to a PC/workstation.

architecture and the more regularized instruction set associated with RISC processors makes for an easier starting point for assembly language programming. The RISC instruction set simplicity is also reflected in the internal processor architecture, making it straightforward to explain concepts and problems associated with pipelining, register forwarding, and the like. It was for this reason that I began to look for an inexpensive RISC system. Combined with the processor in the student's PC, it was then possible to provide assembly language programming experience first on the simpler RISC systems and then compare this with the CISC processors in their PCs.

One of the courses taken in the computer engineering minor program in our department is on comparative architecture. This compares the characteristics of various CISC, RISC, and DSP processors. It was through this course that we have become familiar with the Advanced Micro Devices (AMD) 29k RISC processors.

Through finagling, and considerable cooperation from AMD, we were able to equip several laboratory stations with STEB evaluation boards, which were upgraded to the highly pipelined floating-point Am29050 processor capable of 40-MHz single-cycle performance. With home-built interfaces, we have used these boards in laboratories to show that RISC processors are capable of handling high-speed DSP applications. However, even when these boards were avail-

able, the price of the boards and chips was not conducive to upgrading all our laboratory stations and providing hands-on experience with RISC processors in our general assembly language teaching.

The dearth of articles and books associated with RISC programming echoes the misconception that RISC means high-end processors—expensive and therefore unsuitable for use in general teaching. While this might be true for the Intel i860 and AMD Am29050 microprocessor

floating-point chips, it certainly is not the case for the low-end RISC microcontrollers such as the Am29200 microcontroller.

THE SA-29200 RISC EVALUATION BOARD

When reading this overview, keep my expectations in mind. I am looking for an opportunity to upgrade a number of laboratory workstations to a high-speed, up-to-date RISC processor at a fairly low price. In addition, today's students do much of their work, particularly software development, at home on their own PC. This is because the allotted laboratory time is either insufficient or not flexible enough. In our department, we try to have a flexible policy to signing out equipment. However when using the boards, hardware alone is not sufficient, as inexpensive software is required in order to be able to provide students individual copies and avoid "pirating" issues. Since the student, or experimenter, is not developing a large commercial application involving a

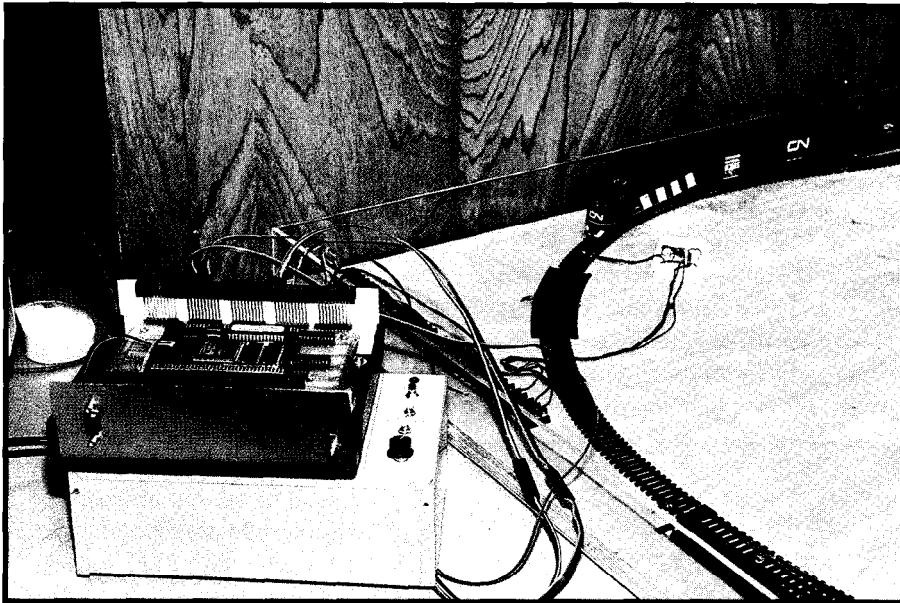


Photo P-Computer "train"ing opens up opportunities for steady and transient power control, digital switching, and sensing along with multiprocessor and operating system development.

long project development time, a few compromises to keep the "cheap" aspect of the software can be made.

HARDWARE

The AMD SA-29200 evaluation board fits many of the requirements discussed in the previous paragraph. It only requires a low-power 5-V power supply and can be attached directly to the serial port on the student's home or laboratory PC or other workstation. Cost is kept down as board I/O uses the workstation screen and keyboard together with a communication link established through host interface calls to an on-board operating system. The board is complete with boot ROM, debug monitor, and 1 megabyte of RAM that is 32 bits wide. It has a respectable speed as it runs at 16 MHz with single-cycle performance.

Although the Am29200 microcontroller does not have the Harvard architecture of the other 29k processors, it is pipelined and has 192 registers, of which 128 are configured for use with register windows. Floating-point operations are supported through instruction traps to the SA-29200 operating system.

We have used the SA-29200 board for projects during the winter 1993 term. Photo 3 shows it hooked up to several D/A converters connected to an oscilloscope in X/Y mode to provide

the elements of a vector-scan display device. Other demonstrations included radio-controlled cars, LCD screens, and security systems—all requiring typical embedded processor tasks that the student can expect to experience in industry.

As can be seen from Photo 1, the SA-29200 board is a small-sized system, measuring only 3"x5". However, we have mounted it in a larger box for a number of reasons. First, it meant we could power it using the connections on our student prototyping station, shown in Photo 3. We also made it easier to connect to student projects using multiline cables made by companies such as Molex (C-GRID modular interconnection system) and Amp (AMPMODU interconnection system). The Am-29200 microcontroller pins are all brought out to SA-29200 connectors, allowing access to the PIO, PIA, JTAG, video, address, and data lines. However, the newer versions of the board have the "long" end of the pins to the rear of the board, inconvenient for the way we want to use them. AMD does provide a multilayer prototyping board to use with the SA-29200 board. This board has a parallel port and slots for additional RAM/ROM. We did not find this expansion board cost-effective for our implementation, but that may not be the situation for other applications.

In addition to providing easier signal access for the students, the box enables us to provide replaceable buffers for the board's I/O lines. These are probably not necessary since the board is robust. However, students can be harder on a system than industry. Also the boxes act like the plastic cases found around audio tapes in the stores; making the boards less likely to walk away permanently.

SOFTWARE

Evaluation boards are intended by a manufacturer to be used as a test bed for their high-powered company software. We were able to get 29k processor software under AMD's support program for use in the University laboratory. However, that is not the route to take if you want the student to have access to suitable software for projects developed on their home PCs.

The board can be attached to the serial port on either an MS-DOS or Sun workstation. It comes with a diskette with the `mondf` program, which supports full debugging and simple assembly language programming from the keyboard or an ASCII file, and the `montip` program, which handles the communications between the board and `mondf`. The diskette also includes a soft copy of the manual that could be useful if the SA-29200 package is to be used in a stand-alone configuration with no other software. I would recommend using the ROM file provided to activate a high-speed serial link between the workstation/PC and the board.

A 29k C compiler can be obtained by turning on the appropriate option of the `gcc` compiler available from the network. We have the compiler installed on Sun workstations networked to PCs via Ethernet or modem. The code generated from `gcc` can be downloaded directly to the board using the `mondf` as long as you invoke it with the `-Cfi 1 ename` option and appropriate path settings.

The simplicity of RISC syntax means that a "free" assembler, able to handle documentation and variable names, can be made with `mondf` in conjunction with a preprocessor pass

through Turbo C or g c c. The code below demonstrates the simple form of definition file required for the preprocessor which allows the use of macros and other aids.

```
/* 29k local register--window */
#define number, 1r2

/*29k CPU general register*/
#define rtnvalue, gr96

/* Show use of macros */
/* No explicit 29k CLEAR inst. */
#define CLEAR(reg) srl reg, reg, 0
```

Although not necessary, it only takes a few additional lines in C, in addition to the preprocessor pass, to handle labels and recognize syntax errors before `mon d f e` does. Such an assembler is useful to reduce the effective cost of any marketed evaluation board. We turn a "bug" into a "feature" by getting students to develop such a program to give them some familiarity with assembler development.

The ethics associated with acquiring legal copies of software still, unfortunately, needs to be continually stressed, not only to the students but also to many academic staff members. The availability of these software items provided with the board or from the network, means that it is cost effective to get an up-to-date RISC processor for home projects without the students deciding to pirate anything.

DISADVANTAGES

There are few disadvantages to using the board. The SA-29200 user guide describes the board and its interface, but provides no information on the processor itself. However, the Am29200 microcontroller user manual is actually a useful teaching tool, which is a little unusual for a manual. If you want to examine pipelined floating-point operations, the Am29050 processor manual covers most aspects.

The `mon d f e` program is simple to use and comes with

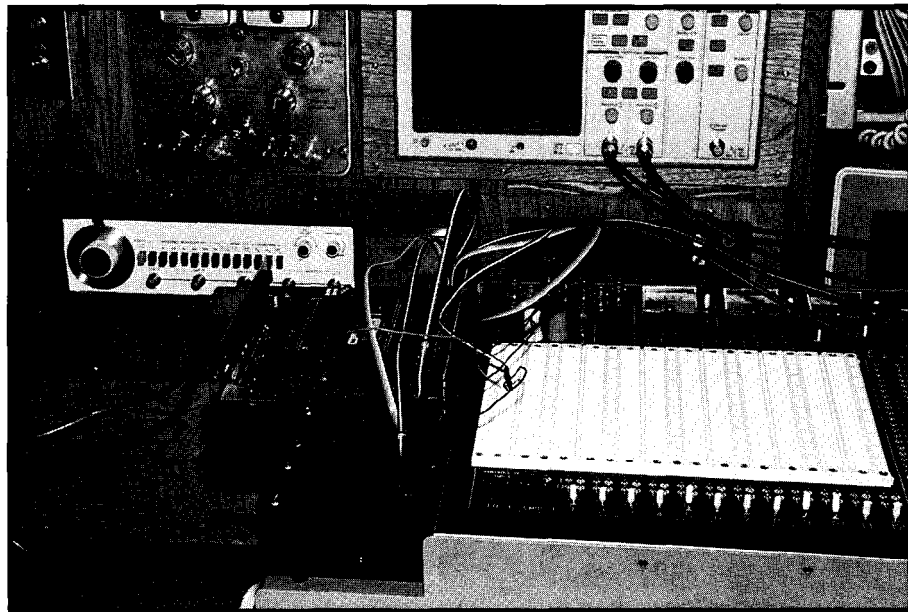


Photo 3-The SA-29200 board and an oscilloscope in X/Y mode can be used to make a vector-scan video controller.

a series of help screens and manual pages. The host interface (HIF) on-board operating system allows basic communication between the user's program and the workstation screen/keyboard/file system. The RISC syntax is also straightforward, especially if you adopt a convention of putting `NO P` instructions in all the delay slots behind the `J U M P` and `CAL L` instructions rather than having students optimize the code by hand.

The demonstration programs on the diskette, however, seem to be more appropriate for other 29k processors. In my opinion, they might as well have not been there. However, I am familiar with the 29k processors and perhaps the programs might be more useful to the first-time user than to me.

As mentioned earlier, I am looking for an opportunity to upgrade a number of laboratory workstations to a high-speed, up-to-date RISC processor at a fairly low price. In this context, my only major criticisms are with the assembler that is built into the `mon d f e` program. The Motorola M68332EVK board, albeit at a higher price, comes with a full "freeware" assembler. It may be straightforward to build a RISC assembler as discussed in an earlier section. However, if AMD wants to encourage the use of the SA-29200 board as an inexpensive teaching tool, and expose more students to their products, they need to do something similar.

Downloading ASCII code to be assembled by the monitor is a satisfactory approach for simple, and small, projects, but the board is straightforward enough that there is little need to place this kind of restriction on its potential application. This may be an unfair comment as I am not using the board with the "official" compiler/assembler the way the manufacturer expects me to. However, if they were not expecting that some people would use the board in a stand-alone configuration, why bother adding an assembler capable of handling

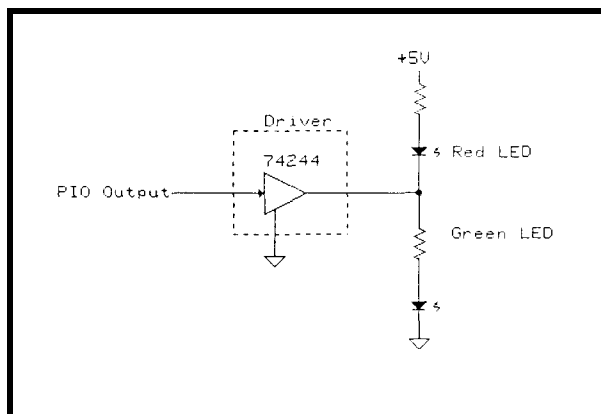


Figure 1-Basic interfacing project to control LEDs.

disk-based files to **mondf** in the first place!

The thing that annoyed me the most was the fact that the SA-29200 board is able to quickly download common object file format (COFF) files generated by the "official" assembler. However, the **mondf** program provides no simple way of generating these files. It would have even been satisfactory if **mondf** had assembled programs to the board memory and then provided an option to upload/download a section of memory in a binary format, or even S records as does the MC68332 board.

This situation could have been used to advantage for both data and instruction transfer in many projects. It does not take long to use the board's operating system calls to generate a program that would do this, but, since it is so simple, why not provide the necessary code on the diskette or as part of the board debug monitor? I have made a number of suggestions directly to AMD, and it remains to be seen whether any of these are taken up in their next release of **mandfe** (**mondf** 3.3, released since I submitted this article, has some useful new commands).

COMPUTER "TRAIN"ING

The usefulness of computer "train"ing very much depends on your point of view. Kalmbach Publishing put out a series of books on model trains. One of their authors, P.J. Thorne, an obvious train buff, makes the following comment:

"...The capabilities now exist to develop a completely computer-controlled model railroad. Given sufficient detection and feedback gadgetry, you could run several trains. In all honesty, though, having built such a layout, how much fun would it be!..."

My response to him is that programming and hardware design are fun in their own right. In addition, the layout development

provides good practice for the computer manipulation of any production line.

This project was originally intended to be done in a nonserious manner. However, when examined in depth, there are many embedded processor related topics covered in a semi- or fully automated HO train system, such as:

- Steady-state power control (engine speed).
- *Transient power control (the track switches require instantaneous power current levels of several amps).
- Digitized switching (control of signal lights).
- *Motion and position sensing (position of powered [engines] and unpowered [carriages] devices).
- *Interrupt handling (even identical trains do not run at the same speed).
- *Local-area networks (having multiple processors controlling different engines on different tracks).
- *Operating system development. (A single train is simple. Many trains in a general system are difficult. We are working on a "train" compiler.)
- Many other aspects of interfacing, project management, and interper-

sonnel relationships (associated with multistudent projects).

All this for around \$100 and an evaluation board. In fact, we intend to get both RISC and CISC experience out of the project. The RISC experience comes from the SA-29200 board. The CISC experience comes by using the processor in the student's PC, and the PC's parallel port as the equivalent of the PIO interface.

We have developed all the hardware for the "train"ing project and are now working our way towards completing a book providing simplified details of the software and hardware. However, the following sections give some idea of the problems we have handled in the context of what the projects show about embedded processor applications.

WARNING

These trains are more tricky to handle than you think. We have tried out the following circuits and, since we have yet to see smoke, they seem suitable at least in the short run. However, we are train neophytes and perhaps there is something that will be more obviously wrong as we expand the complexity of the board. (That's

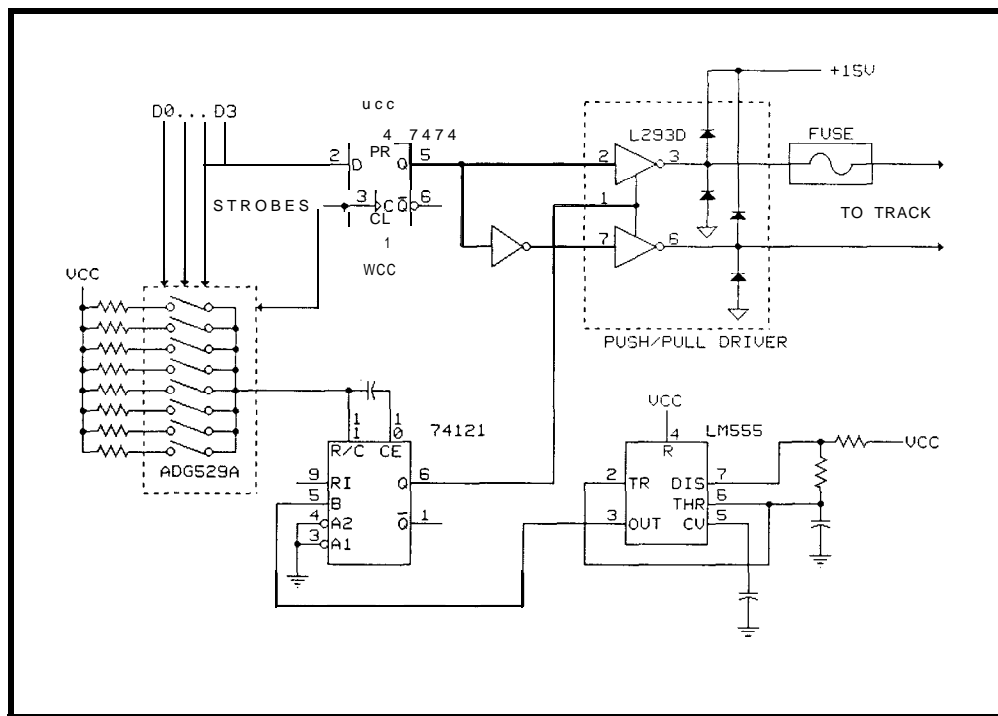


Figure 2—The computer-controlled switching train power supply creates a power waveform that resembles that generated by the train's stock power supply.

why we have asked R. Krouse, a train buff of many years, to help us with the book; we need his experience!)

SIMPLE PROJECT

An important part of any project is to give the student a straightforward task to get the necessary background hardware/software experience of using a processor. Figure 1 shows a series of LEDs driven by a PIO line. This project is not intended to be difficult and allows the student to test basic interfacing techniques under the context of providing signal lights for the train layout.

MORE COMPLEX PROJECTS

One of the intentions of using the evaluation board is to expose the student to various interfacing techniques. The projects are therefore directed that way. The train set we finally intend to control requires track sections and turnouts (switches), each with a different power supply, together with lights, sensors, and other animated track-side activities. This cannot be handled using the 16 lines of the microcontroller's PIO port. The various signals could be generated by a number of PIA-controlled devices connected directly to the microcontroller bus. However, with a high-speed processor, this unnecessarily complicates student projects. Because of time restrictions, this approach will probably lead back to "cookbook" laboratories.

Instead, we chose to use the PIO port as a private data bus by driving the PIO pins and then capturing that information on an appropriate latch using the microcontroller PIA strobe signals. We are also trying to use the printer "video" port of the Am29200 microcontroller to handle local-area network traffic between a number of processors to demonstrate interrupt handling and DMA activity. Interrupt priorities can be useful when monitoring sensors activated by "local" and "fast" train movement.

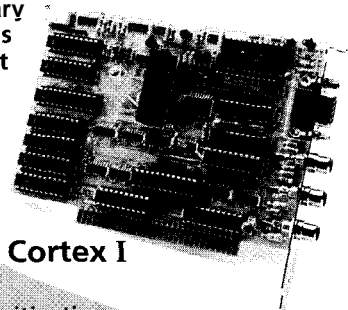
The schematic for the train power controls is shown in Figure 2. The narrow clock pulses of the 555 timer are stretched using the 74123 one-shot. The widths of the one-shot pulses are

Video Frame Grabber

- \$495 Including Software with "C" Library
- Half Slot Card for Compact Applications
- Real Time Imaging with Display Output
- 8 Bit (256 Gray Levels)

FEATURES:

- Single 512 X 484 Image or Four 256 X 242 Images
- External Trigger
- Input Look Up Table
- Low Power Option Available
- Elegant Software Interface
- <10 nsec Pixel Jitter Means Accurate Digitization
- EISA (PC) Bus and STD Bus Products Available
- RS-170 and CCIR Video Formats Available
- Binary and TIFF File Formats



Cortex I

The Cortex I is used in machine vision, industrial control, medical, security and scientific applications around the world.

ImageNation strives to delight customers with quality products and personal service at a competitive price.

Call today for volume pricing or to discuss your application.

ImageNation Corporation

Providing Imaging Solutions

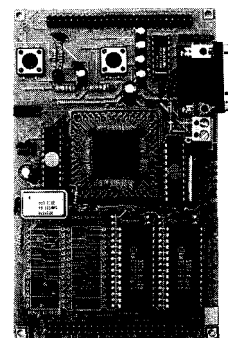
P.O. Box 276
Beaverton, OR 97075
(503) 641-7408

FAX (503) 643-2458 • (800) 366-9131

#120

NOW 16-BIT POWER FOR UNDER \$200 !!

Rigel Corporation introduces the RMB-166, an evaluation board for the Siemens 80C166 16-bit high-performance microcontroller. The board is configured to run at 40Mhz with no wait states. A set of option headers, decoded by PAL devices make the RMB-166 a flexible hardware platform. All system address, data, and control lines as well as ports 2, 3, and 5 are terminated at two 50-post headers. Both serial ports are available to the user at W-232 levels through DB-9 sockets, and a 6-post connector.



The RMB-166 package includes:

HARDWARE: Flexible and embeddable (4 by 6 inches). Two 16-bit general-purpose input/output ports, 10 bits of analog or digital inputs. An internal IO-bit analog-to-digital converter. Designed for 64K of EPROM, and 64K or 256K of RAM (only 64K RAM installed).

SOFTWARE: READS166 - PC host software: Runs in MS-Windows 3.1, allows downloading and running application programs, comes with demonstration programs, **RMON166**-80C166 monitor program: Supports basic memory and port functions. Works with READS166 to download and run application programs.

EVALUATION SOFTWARE PACKAGES INCLUDED: From **BSO/Tasking**, **CMX**, **Hill Country Research**, and **Embedded System Products**, (formerly A.T. Barrett and Assoc.)

DOCUMENTATION: RMB-166 User's Guide with information on: the bootstrapping feature, the minimal monitor, and RMON166 monitor. Step-by-step tutorial, PAL equations, Circuit diagrams. Commented source code includes: bootstrap code, the minimal monitor, and RMON166.

BULLETIN BOARD: Now available for tech support, software updates, free software, and users group. Supports both the 8031 8-bit microcontrollers, and Siemens' 16-bit microcontrollers. **BBS Number (904) 377-4435 Phone Number (904) 373-4629**

RIGEL CORPORATION, PO BOX 90040, GAINESVILLE FL, 32607

#121

At only \$21.95*
for twelve issues...

...the Computer Applications Journal

could be the best investment
you make this year.

*Domestic price. Canada and Mexico add \$10, all other foreign countries add \$28. U.S. funds drawn on U.S. banks only.

With your subscription to the *Computer Applications Journal* you will receive twelve great issues plus valuable discounts on the Circuit Cellar Project File book series. With each issue you can count on:

- ➡ the wit and wisdom of Steve Ciarcia
- ➡ high-level tutorials
- ➡ fully tested projects
- ➡ hardware design and selection tips
- ➡ clearly explained firmware design
- ➡ documented application software

a n d m o r e !

Don't miss a single issue! Mail in the subscription card on page 17 of this issue or FAST FAX it to:

The Computer
Applications Journal
attn: Subscriber Services
Fax: (203) 872-2204

ACCLAIMED AS THE
TECHNOLOGY RESOURCE
OF THE '90s!

controlled by switching a series of resistors through an analog mux. Speed and train direction is handled through the H-bridge in the L293D push-pull driver and four PIO lines. The 15-V pulses are then supplied to one of the tracks to control an engine. The turnout switch operates in an equivalent way.

The power supply must handle the engineering compromises that the model train manufacturer uses to keep the trains inexpensive. The switched power supply offers the opportunity to slowly speed the train, simulating train momentum and keeping train buffs like Mr. Thorne happier. In addition, the pulses shake the engine, freeing up cheap bearings and making the train able to move more smoothly and at lower power levels.

The fuse in the circuit is a compromise to circumvent (possible) problems in programming. With a train being passed between controllers and different track sections, it is possible to have two adjacent tracks powered with opposite polarities. This means the bogies on the train can short out different power supplies. I teach my students systematic assembly language programming and debug techniques, so a slow-blow fuse is more than adequate. However, if the students are really ham-fisted, you can always turn a bug into a feature and have them develop some current-limiting sensors!

CONCLUSION

In this article, I gave a brief overview of the Advanced Micro Devices SA-29200 RISC evaluation board and discussed the advantages of using RISC processors in the teaching of assembly language programming. Evaluation boards, CISC or RISC, are an inexpensive way to keep a laboratory up-to-date with current technology advances. Costs are kept down as the boards use the keyboard/screen of the a PC or workstation for input and output through calls to a host interface built into the board ROM. Microcontrollers are particularly suited to projects because of their built-in serial, video, and programmable input/output ports. Although not without their own

problems, RISC processors make an easy starting point for assembler language programming than do the CISC processors with their rich variety of addressing modes.

The success of using this commercial SA-29200 RISC evaluation board has lead us to undertake further work with other DSP and CISC boards. This approach provides an inexpensive path to keep our laboratory up-to-date with the rapid changes in processor technology.

ACKNOWLEDGMENTS

I would like to thank the University of Calgary for financial support during the development of these teaching projects. In addition, I have had considerable cooperation from Advanced Micro Devices through their University Support program. Technicians from my teaching laboratory, R. Thompson, W. Flaman, and E. Evanik, were responsible for the implementation of the control devices based on my original concepts.

I would also like to thank students T. Chamberlian and E. Ellefson for taking my code and the hardware at the beginning of the course and demonstrating that it would actually work. I must have taught them structured programming techniques well: the slow-blow fuses only show a few molten balls, signs associated with current overloads. ☐

Michael Smith is a professor of Electrical and Computer Engineering at the University of Calgary teaching assembly language programming, comparative processor architecture, and computer graphics. He may be reached at smith@enel.ucalgary.ca.

REFERENCE

- P. J. Thorne, 34 New Electronic Projects for Model Railroaders, Kalmbach Books, Waukesha, 1990.

I R S

- 407 Very Useful
408 Moderately Useful
409 Not Useful

FEATURE ARTICLE

Michael Alwais

Improving the Performance of an 8051-based System

For most, seeing their kids grow up and finally move out of the house is a hard thing to do. For some, the same holds true for their first microcontroller. One thing is certain, they will always be there for you.

Wide acceptance of the 8051 architecture during the past 15 years has created a large knowledge base and existing software library for the part. The incentive to use what is familiar allows the 8051 to remain one of the most commonly used microcontrollers. Yet during the past 15 years, few innovations have allowed for a real improvement in the performance of the 8051.

Until recently, performance was limited to the standard 16-MHz version which performed an instruction cycle in 750 ns. Manufacturers of the standard 8051 shrunk and recharacterized the old design to allow higher-frequency operation. The only way to increase performance was to increase the crystal speed. This technique brought an increase in power consumption, radiated emissions, and the complexity of the off-chip interface which frequently required a redesign of the hardware. When new versions appeared, most included a range of new peripherals, but few architecture improvements.

The introduction of the DS80C320 from Dallas Semiconductor opens a new realm of performance for the 8051 user. This first part, based on a new high-performance 8051 core, uses only 4 clocks per machine cycle instead of the traditional 12 used in a standard 8051. Thus, 8051 code gets a 3x improvement in peak performance. The DS80C320 pinout is a direct replacement for the standard 80C32 (or 80C31). All of the internal resources of the 80C32 such as RAM and timers are duplicated. In several cases, they have been improved. For example, the inclusion of a second data pointer (DPTR) simplifies the process of a block move of data. Figure 1 shows a direct comparison of the DS80C320 to the 80C32.

The DS80C320 uses an identical instruction set to the standard 8051 and is object code compatible. A timing comparison on an instruction-for-instruction basis is included in Table 1. An average across all instructions gives a relative performance improvement of 2.5x. That is, for identical crystal speed, software using all of the instructions would be executed 2.5 times faster on a DS80C320 than on a standard 8051. Since the instructions are identical, the user can simply drop in the DS80C320 in place of the standard 80C32 and measure the actual improvement of throughput.

Dallas constructed a comparison test set as shown in Figure 2 for the Embedded Systems Conference. The standard 80C32 and DS80C320 were given identical EPROMs with four example programs. They were run from a common 25-MHz clock and

DS80C320	80C32
4-clock machine cycle	12-clock machine cycle
256 bytes direct RAM	256 bytes direct RAM
3 16-bit timers	3 16-bit timers
2 on-chip UARTs	1 on-chip UART
2 DPTRs	1 DPTR
6 external interrupts	2 external interrupts
Programmable MOVX timing	Fixed MOVX timing
Power Monitor-Reset	
Brownout detection	
Programmable Watchdog Timer	

Figure 1—Direct feature comparison *between the DS80C320 and the standard 80C32.*

synchronized to begin simultaneously. An arbitrator measured the elapsed time from start to completion. The results were displayed on an LCD. Since the DS80C320 is instruction set compatible with the 8051, the standard Franklin C compiler was used to generate the benchmark programs. No optimizing was performed. Also, the Dual Data Pointer of the DS80C320 was not used, though it is supported by Franklin. The following is a description of the four programs that were used and the results of the comparison.

SHELL SORT

The shell sort initializes an array with 4096 random unsigned integers. The random number generator is initialized with the `Srand` function so the same sequence is executed every time. The shell sort uses a diminishing increment of three. When the C programs were run by both the DS80C320 and 80C32, the DS80C320 finished 2.35 times faster than the 80C32.

PI

The pi calculation finds pi to 116 places. This algorithm uses 32-bit integers and the software's absolute performance is sensitive to the 8-bit handling code of the compiler. However, in comparing the two micros, this was irrelevant as the software is identical. The DS80C320 ran the pi code 2.26 times faster than the 80C32.

PRIME

In the prime benchmark, the first 6495 prime numbers are found using the sieve of Eratosthenes. This happens to be all of the primes less than 65025. The number 65025 fits into an unsigned 16-bit number and produces a square root of 255. The DS80C320 ran this code 2.25 times faster than the 80C32.

DHRYSTONE

This is the standard Dhrystone benchmark algorithm. It was written by Reinhold P. Weiacker, CACM vol. 27, No. 10 10/84 pg. 1013, and translated from Ada by Rick Richardson. Normal benchmarks list the number of dhrystones per second. In this test, the DS80C320 and 80C32 were given

Instruction Category			Quantity	Speed Advantage
Total Instructions:	One Cycle	One Byte	37	3.0
Total Instructions:	Two Cycle	One Byte	4	3.0
Total Instructions:	Two Cycle	Two Byte x1.5	27	.5
Total Instructions:	Two Cycle	Two Byte x3.0	11	3.0
Total Instructions:	Three Cycle	One Byte	4	2.0
Total Instructions:	Three Cycle	Two Bytes	8	2.0
Total Instructions:	Three Cycle	Three Bytes	7	2.0
Total Instructions:	Four Cycle	One Byte	2	1.5
Total Instructions:	Four Cycle	Three Bytes	9	1.5
Total Instructions:	Five Cycle	One Byte	2	2.4
Average across all instructions			111	2.3

Instruction Category			Quantity	Speed Advantage
Total Opcodes:	One Cycle	One Byte	126	3.0
Total Opcodes:	Two Cycle	One Byte	6	3.0
Total Opcodes:	Two Cycle	Two Byte x1.5	35	1.5
Total Opcodes:	Two Cycle	Two Byte x3.0	27	3.0
Total Opcodes:	Three Cycle	One Byte	4	2.0
Total Opcodes:	Three Cycle	Two Bytes	29	2.0
Total Opcodes:	Three Cycle	Three Bytes	7	2.0
Total Opcodes:	Four Cycle	One Byte	2	1.5
Total opcodes:	Four Cycle	Three Bytes	17	1.5
Total opcodes:	Five Cycle	One Byte	2	2.4
Average across all opcodes			255	2.5

Table 1—On average, the DS80C320 will give a 2.5-times speed improvement over a standard 8051 using identical instructions and clock speeds.

4000 dhrystone loops. The DS80C320 ran this procedure 2.22 times faster than the 80C32. (Due to space limitations, I haven't listed the source code for the dhrystone code here. It is available to download from the Circuit Cellar BBS, however.)

INTERRUPT HANDLING PERFORMANCE GAINS

The speed improvements are achieved by simply replacing an 80C32 with the DS80C320 without regard to clock frequency. These simple one-task programs illustrate the speed

improvement that develops from an uncontrolled mix of instructions. Note that while the basic machine cycle of the DS80C320 is 3x faster, not all instructions are performed in a single machine cycle. Thus not all instructions are improved by 3x. However, the DS80C320 improves the execution speed of every instruction. The relative improvements run between 3.0 and 1.5. Some improve by a factor of 2.0 or 2.4. Thus the actual improvement to a single program is usually between 2.0 and 3.0, but varies depending on the instruction mix.

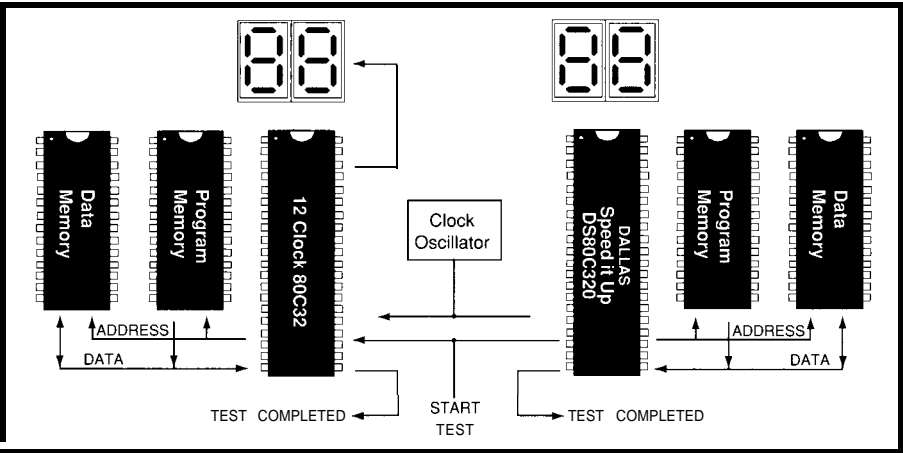
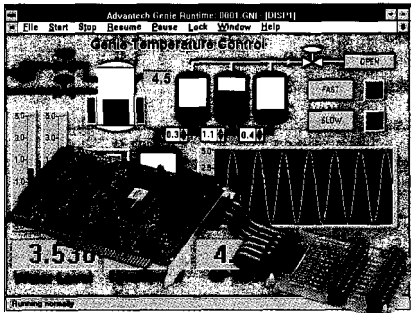


Figure 2—By putting a DS80C320 side-by-side with a standard 80C32 running identical code, the overall speed improvement realized by the new part becomes obvious.

Data Acquisition & Control Solutions

Temperature Measurement
High Speed Data Streaming
Remote Data Acquisition
Strip Chart Recording



Temperature Measurement & Control
Solution Package ~\$895

Key Features:

- ❖ For J, K, S, T, B, R, E thermocouples
- ❖ 4 8 thermocouple channels, expandable to 32 channels
- ❖ 0.1°C resolution with 12-bit A/D
- ❖ Up to 250 samples per second
- ❖ 16 D/I and 16 D/O channels
- ❖ Powerful Windows SCADA software
- ❖ Icon based, no programming required

7 General Purpose DAS Card 8 A/D, 1 D/A, 16/16 DIO, wiring kit	\$295
1 Multifunction DAS Card 16 A/D, 2 D/A, 32 DIO, counter	\$395
7 High Performance DAS Card 100KHz, programmable gain	\$595
7 24 Ch Digital I/O Card	\$100
1 24 Ch Digital Input w/Interrupt	\$180
7 32 Ch Digital I/O Card	\$170
1 32 Ch Digital Input w/Interrupt	\$235
7 144 Ch Digital I/O Card	\$295
1 8 Relay & 8 Digital Input Card	\$210
7 High Speed Data Streaming Pkg. 100KHz data streaming to disk	\$595
7 PC Strip Chart Recorder Pkg. 16 channel recording at 15KHz	\$695
7 Remote DA&C Starter Kit RS-485 based, up to 4000 feet	\$495

Free
120-page Solution Guide
for quality minded,
budget conscious
Engineers



1-800-800-6889

Industrial & Lab Automation with PCs
ADVANTECH

750 East Arques Ave. Sunnyvale, CA 94086
Tel: (408) 245-6678 FAX: (408) 245-8268

Listing 1—The benchmark **code to compare the DS80C320 to a standard 80C32** is written in Franklin C and can be compiled using their off-the-shelf compiler.

Dallas Semiconductor DS80C320 Benchmark Code

```
// Slave code:
// On reset clear P3.4 (PRESENT) to let master
// know we are present and running.
// Read a 3-bit number from port 3, bits 0-2;
// execute the corresponding request.
// Clear p3.3 (DONE) to signal master test done.
// Define PRINT to get results from the pi & prime tests.
// #define PRINT

// Define FRANKLIN when compiling for the 8051
// Franklin C compiler, V3.40.
// The following commands are used to compile & link code.
//      c51      slave.c code large
//      151      slave.obj
//      ohs51     slave hex (slave.hex)

#define FRANKLIN
#ifdef FRANKLIN
#include <reg51.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef unsigned char uchar;
typedef unsigned int  uint;
typedef unsigned long ulong;

#ifdef FRANKLIN
sbit LATCH1 = P1^4; // latch for 1st LED on pl
sbit LATCH2 = P1^5; // latch for 2nd LED on pl
sbit DONE   = P3^3; // clear to notify master we are done
sbit PRESENT = P3^4; // clear to notify master
sfr CKCON   = 0x8e; // address of 320 clock control register
#endif

// program globals
uint cnt;
uchar digit1;
uchar digit2;

void inc_and_display(void)
{
    if (++digit1 > 9)
    {
        digit1 = 0;
        if (++digit2 > 9)
            digit2 = 0;
    }
}

#ifdef FRANKLIN
P1 = digit1 | 0xe0;
LATCH1 = 1;
P1 = digit2 | 0xd0;
LATCH2 = 1;
#endif

void clr_cnt(void)
{
    cnt = 0;
    digit1 = 9;
    digit2 = 9;
    inc_and_display();
}
```

(continued)

Listing 1-continued

```
//*****
// prime begin 6495 primes are available in the first
// 65025 integers

#define MAX      65025      // look for primes up to this number
#define SQRTMAX   255      // square root of max
#define MAX8      8129      // max divided by 8

uchar d[MAX8];
const uchar b[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
uchar bit_set(uint i)

    return d[i/8]&b[i&7];

void clear_bit(uint i)

    if (++cnt == 1290)

        cnt = 0;
        inc_and_display();

    d[i/8] &= ~b[i&7];

void prime(void)

    uint i, j, n;
    clrnt();
    for (i = 0; i < MAX8; ++i)
        d[i] = 0xff;
    for (i = 2; i < SQRTMAX; ++i)
        if (bit_set(i))
            for (j = i + i; j < MAX; j += i)
                clear_bit(j);

#ifdef PRINT
    printf("\nPrimes:\n");
    n = 0;
    for (i = 2; i < MAX; ++i)
        if (bit_set(i))
        {
            printf("%7u", i);
            if (++n == 11)

                printf("\n");
                n = 0;

        }
    printf("\n");
#endif

// prime end
// *****
// pi begin

long pm[300];
long tm[300];
long mm, mn, nn;

int checkxb(long *cm, int cb)    // Reduce current block number
(speed up)
{
    long *cmm;
    cmm = cm;
    cm += cb;
    if (*cm == mm && *--cm == mn && *--cm == nn)
```

(continued)

\$149 ICE?!!!

Yes, that's right! HTE has dropped the price of its popular 8031132 Enhanced **DryICE** from \$269 to \$149. Now you can't afford to be without one! This ICE performs single step, real-time execute to breakpoint, disassembly and more. 8K user code space, expandable to 32K. Order yours today!

More ICE

Our **DryICE Plus** has so many features, the price is unbelievable. 48K of user code space, real-time execution, and expandability to nearly all of the 8051 family processors. We are now supporting pods for the 8031/2, 8751, 80CL51/32, 8051 Fx, 80C154, 80C410/1, 80C451, 80C528, 80C550, 80C535, 80C537, 80C552/562, 80C575, 80C652, 80CL781/2, and 80C851. With pods priced at only \$149, going to a different, higher integration processor is easy. Pod and base unit are just \$446 complete!

80C552

At \$149, our 552SBC-10 OEM board has the price and features you need right now! It's an 8051 core processor with an eight channel, 10-bit A/D, two PWM outputs, capture/compare registers, 16 digital I/O, one RS232 serial port, four JEDEC memory sockets. Add options like two more RS232/422/485 ports, 24 more digital I/O, real-time clock, serial EEPROM, and battery-backup for clock and RAM. Start with the Development board; all the peripherals and a debug monitor for only \$349. Download and debug your code, assembly or 'C', right on the SBC, then use the low-cost OEM board of your choice for production. We also do customs - call for a free quotation.

80C188 SBC

NEW! Starting at only \$249. Two serial RS232/422/485, Parallel, expand to 16 ch. 12-bit A/D, 8 ch. 12 bit D/A, Keybd, LCD, Relay I/F, more. Call for details!

**Call for your custom product needs.
Quick Response**



HTE

HiTech Equipment Corp
9400 Activity Road
San Diego, CA 92126
(FAX: (619) 530-1458)

(619) 566- 1892-

70662. 1241 @compuserve.com

When this speed improvement is applied to a real-world problem in a critical area, the speed improvement can be larger. The following situation explains how this results.

An example system implements one main loop and one interrupt service routine. The system requires that an external interrupt be serviced at an 8-kHz rate. In an 80C32 based system, the example ISR occupies about 60% of the time. This means for an interrupt period of 125 ms (8 kHz), the 80C32 ISR needs 75 ms, leaving 50 ms to perform the main loop. When the DS80C320 is dropped in, this situation changes for the better.

Assuming a conservative speed-up factor of 2.25 applied to the ISR, this code now takes only 33.3 ms. This improvement in a bottleneck area leaves an extra **41.7 ms** for the main loop. Thus the main loop now has a total of 91.7 ms for execution. Applying the same speed-up factor of 2.25 to this code, the DS80C320 can perform 4.1 times the work of a standard 80C32 in the main loop. This results from being 2.25 times faster in the main loop and getting more time in the main loop by doing the ISR 2.25 times faster. The more interrupt bound an existing design, the more improvement the DS80C320 can achieve. This can easily occur by introducing more interrupt sources. Depending on the application, the Dual Data Pointer can further improve performance.

Although not all systems are performance limited, most can benefit from extra performance. In the same way that the example ISR was slimmed down to make more time for a main loop, a main program also can be slimmed to make room for new software features. The beauty of this speed improvement is that the user's software is still ordinary 8051 code.

CONCLUSION

The prevalence of 8051 designs and the lack of real evolution has created several design challenges that can be addressed by the DS80C320. First, the obvious need for more performance. This applies to existing 8051-based platforms that need increasing performance. It also applies

Listing 1-continued

```

    cb--;

    cmm += cb;
    mm = *cmm;
    mn = *--cmm;
    nn = *--cmm;
    return(cb);

void multiply(long *cm, long x, int cb) // cm *= x

    long *cmm;
    long c, z; // carry
    for (c = 0, cmm = cm + cb; cm < cmm; )

        z = *++cm*x + c;
        c = z/10000;
        *cm = z % 10000; // *cm = z % 10000

void divide(long *cm, long x, int cb) // cm /= x

    long *cmm;
    long c, z, q; // c=carry, z, q=temp
    for (c = 0, cmm = cm, cm += cb; cm > cmm; )

        z = *cm + c;
        *cm-- = q = z/x;
        c = (z - q*x)*10000; // c = (z%x)*10000

void add(long *cm1, long *cm2, int cb) // cm1 += cm2
{
    long *cmm;
    long c; // carry
    for (c = 0, cmm = cm1 + cb; cm1 < cmm; )

        if ((*++cm1 += (*++cm2 + c)) >= 10000)
        {
            *cm1 -= 10000;
            c = 1;
        }

        else
            c = 0;

void display(long *cm, int cb)

    int i;
#ifdef PRINT
    printf("%ld.", *(cm += cb)); // left of decimal point
    for (i = cb - 1; i >= 3; i--)
        printf("%04ld", *--cm); // right hand side
    printf("\n");
#endif

void initial(long *cm1, long *cm2, int cb)

    long *cmm;
    for (cmm = cm1 + cb; cm1 < cmm; )
        *++cm1 = *++cm2 = 0;
        *--cm1 = *--cm2 = 10000 / 2;

```

(continued)

Listing I-continued

```
void pi(int s)

    int b, xb;
        // number of necessary blocks (elem.) in matrix
    long n, t;          // n=current term, t=max. terms
    clrnt();

    mm = 0;
    mn = 0;
    nn = 0;

    s = ((s + 3)/4)*4; // digits
    b = xb = s/4 + 3; // blocks
    t = s + (2*s)/3;   // terms

    inc_and_display();
    initial(pm, tm, b);
    inc_and_display();
    for(n = 1; n <= t; n++)
    {
        if (n < 232)
            multiply(tm, (4*(n*n n) + 1), xb);
        else
        {
            multiply(tm, (2*n 1), xb);
            multiply(tm, (2*n 1), xb);

            if (n < 115)
                divide(tm, (n*(16*n + 8)), xb);
            else
            {
                divide(tm, (8*n), xb);
                divide(tm, (2*n + 1), xb);
            }
            add(pm, tm, xb);
            xb = checkxb(pm, xb);
            if (++cnt == 2)

                cnt = 0;
                inc_and_display();

        }

        multiply(pm, 6, b);
        display(pm, b);
        inc_and_display();

    }

// pi end
//*****
// shell sort begin

#define SORTMAX 4096

void ss(int n)

    int i, inc, j, x;
    inc = 1;
    while (inc < n)
        inc = 3*inc + 1;
    inc /= 3;
    while (inc)

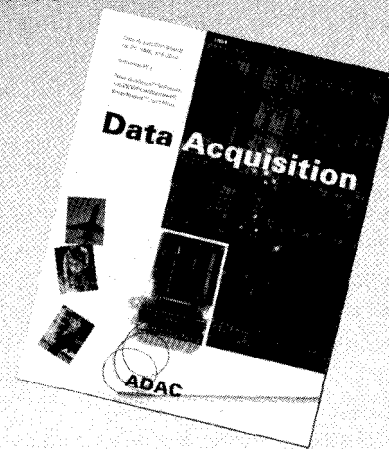
        for (i = inc; i < n; ++i)

            x = d[i];
            j = i - inc;
            if (d[j] > x)
            {
```

/continued)

NEW Data Acquisition Catalog

Covers expanded line.



FREE!

1994 120 page catalog for PC, VME, and Qbus data acquisition. Plus informative application notes regarding anti-alias filtering, signal conditioning, and more.

NEW Software:

LabVIEW[®], LabWindows[®], Snap-Master[™], and more

NEW Low Cost I/O Boards

NEW Industrial PCs

NEW Isolated Analog and Digital Industrial I/O

New from the inventors of plug-in data acquisition.

Call, fax, or mail for your free copy today.

A D A C


American Data Acquisition Corporation
70 Tower Office Park Woburn, MA 01801
Phone: (800) 648-6589 Fax: (617) 938-6553

#1 27

to other platforms that could not have traditionally used an 8051. At a peak of over 6 million instructions per second, with higher speeds coming, the DS80C320 begins to look like a 16-bit microcontroller in terms of speed.

Second is the need to lower power consumption. While the DS80C320 uses a similar power compared to a standard 80C32 running at the same crystal frequency, it can be run slower to do the same job. At less than half of the operating frequency, the DS80C320 does the same work as an 80C32 but uses less power. The actual saving depends on the choice of manufacturer and version of the older part. As an alternative, the DS80C320 can complete its work more quickly, then enter an ultra low power (1 μ A) STOP mode. This can result in a lower RMS power consumption.

A third consideration is the speed and cost of external EPROM. To achieve 3 MIPS, an 80C32 must run at 36 MHz and use 55-ns memories. Finding 55-ns, 32K \times 8 EPROMs that are also guaranteed to run down to 4 V (5 V \pm 20%) can be difficult. The more efficient DS80C320 can achieve the same performance at less than 18 MHz using widely available 120-ns EPROM.

So, no matter which way you look at it. The DS80C320 is a serious contender for any 8031 design that needs a little more horsepower. 

Michael Alwais is the product manager for microcontrollers at Dallas Semiconductor. He is currently involved in feature definition for advanced 8051 products.

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" in this issue for downloading and ordering information.

I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

Listing 1—continued

```

        do
        {
            d[j + inc] = d[j];
            j -= inc;
        }
        while (j >= 0 && d[j] > x);
        d[j + inc] = x;

        if (++cnt > 187)
        {
            cnt = 0;
            inc_and_display();
        }
    }
    inc /= 3;
}

void sort(void)
{
    int i;
    clr_cnt();
    srand(523);
    for (i = 0; i < SORTMAX; ++i)
    {
        d[i] = rand();
        if (++cnt > 200)
        {
            cnt = 0;
            inc_and_display();
        }
    }
    ss(SORTMAX);
}
// shell sort end
//*****

void main(void)
{
    uchar i;
    i = 1;
#ifdef FRANKLIN
#ifdef PRINT
    PCON  |= 0x80;    // set smod bit
    SCON   = 0x52;
    TMOD   = 0x20;
    TCON   = 0x69;
    TH1    = 0xff;    // 57600 at 11.0592 MHz
    TL1    = 0xff;
#endif
#endif
    PRESENT = 0;        // signal master we are here
    CKCON   = 0;        // set memory stretch to 2 cycles
    i = P3 & 3;
#ifdef FRANKLIN
    if (i == 0)
        dhrystone();
    else if (i == 1)
        prime();
    else if (i == 2)
        pi(116);
    else if (i == 3)
        sort();
    DONE = 0;          // signal master we are done
    while (1)
        ;
#endif
}

```

DEPARTMENTS

48

Firmware Furnace

56

From the Bench

60

Silicon Update

66

Embedded Techniques

75

Patent Talk

82

ConnecTime

Testing the '386SX Project's Bitmapped LCD Panel



Now that
the
hardware
for the

large-panel LCD
interface is ready, it's
time to twiddle some
bits. Be sure to take
things one step at a
time, though, because
one misconnection can
put you back at square
one.

FIRMWARE FURNACE

Ed Nisley



Although Steve's favorite programming language remains solder, even he admits most projects are useless without some programming. Last month I covered the molten metal part of the Graphic LCD Interface.. now it's bit-twiddling time.

This month I'll cover the test code I wrote to exercise my panels and explain some tricks and techniques. I used Micro-C, but you'll surely want assembler code once you understand how your panel works. If this isn't a case for hand-tweaked optimization I don't know where you'll find one.

But, first, let's check out your wiring. Because this project has a lot of fairly tricky hardware, I'm going to spend more time than usual on bring-up testing. With any luck, your board will work perfectly the first time, but if it doesn't, these hints help.

TESTING: ONE, TWO...

The GRAPHLCD files appeared on the BBS last month. You can use the BIN file directly or, if you make source code changes, recompile the Micro-C source code into an Intel hex file and convert it into binary. Copy that file onto a diskette with the appropriate boot loader, reset your '386SX system, and select the appropriate option when GRAPH LCD squirts its menu over the serial port.

If you prefer Borland or Microsoft C with Paradigm's Locate, you can adapt the Micro-C code using the suggestions in the December column. There are enough differences that I didn't want to tackle it this month, but I will use Locate for several future

projects. Don't despair... the hardware doesn't care which C you use!

The CPU's view of the Graphic LCD Interface is a write-only output port byte, two 82C54 channels, and a 32K-byte block of RAM. GRAPH LCD D's first four tests wiggle the control port bits, give you manual control of the 82C54 counters, and read and write the RAM in loops.

These tests work fine without an LCD panel, so you can start wiring and testing while waiting for your new panel to arrive. If you do connect a panel, remember that it won't display anything without the correct sync signals and jumper settings. I strongly recommend delayed gratification: keep that panel in the box until your hardware passes the first few tests.

First, verify that all eight bits in U51, the LCD Control Latch, go on and off correctly. As it turns out, I managed to wire the port backwards, so this simple test prevented some serious headscratching.

Next, set the 82C54 row and frame counters. Because there's no LCD panel attached (right!) you can use three clocks per row and nine clocks per frame to simplify your scope display. Each row must have at least two clocks and the frame count must be a multiple of the row count.

Scope the LCD Address Counter, U43 and U44, to verify its inputs and outputs. The address drivers are active (well, they *should* be active) only when Dot Clock is low. The Frame Sync pulse resets the counters with a signal from U56b, so check that the count begins with zero at the right time.

The RAM read and write tests toggle the printer port bits at key points in the loop for scope triggering. The read test uses **REP LODSB** to hit every RAM byte, so you get one sync at the start. The write loop touches each byte with C code that produces a sync pulse for each RAM access. In either case, U56a should stall the Dot Clock in the high state at the end of each -SMemR or -SMemW pulse.

The fifth test writes a 32K-byte pseudorandom pattern into the LCD Refresh RAM and verifies that the bytes return unchanged. While this

Listing 1—*The Optrex DMF651640x200 LCD panel has a comparatively simple layout: a single frame requires 200 sets of 160 Dot Clocks, each transferring four bits. This code shows how to load the 82C54 timers and set the LCD Control Port. The SetData routine puts four data bits in the low nybble and creates the right bits for the high nybble to produce the desired blinking effect.*

```
outpw(GLCD_CTL5,GLCD_OFF);          /* disable clocking */
LoadTimer(1,0x04,160,I8254_BASE);   /* clocks per row */
LoadTimer(2,0x04,32000,I8254_BASE); /* total clocks/frame */
outpw(GLCD_CTL5,GLCD_ON + GLCD_BLMED); /* start and set blinking */
for (Row = 0; Row < Z00; Row++) {
    RAMAddr = Row * 160;
    for (Col = 0; Col < 160; Col++){
        poke(GLCD_SEGMENT,RAMAddr+Col,0x00); /* clear row */
    }
    SetData(RAMAddr+0,BLINK_OFF,Row >> 4); /* show row number */
    SetData(RAMAddr+1,BLINK_OFF,Row);
    SetData(RAMAddr+2,BLINK_FG,0x09); /* dividing line */
    SetData(RAMAddr+3,BLINK_OFF,RAMAddr >> 12); /* show RAM address */
    SetData(RAMAddr+4,BLINK_OFF,RAMAddr >> 8);
    SetData(RAMAddr+5,BLINK_OFF,RAMAddr >> 4);
    SetData(RAMAddr+6,BLINK_OFF,RAMAddr);
    SetData(RAMAddr+7,BLINK_FG,0x09); /* dividing line */
    SetData(RAMAddr+8+(Row >> 2),BLINK_BG,0x08 >> (Row & 0x03)); /* diagonal */
}
```

isn't an exhaustive RAM test, it should reveal obvious wiring errors, shorts, and dead chips. If you don't have the equipment needed for the other tests, you can run this one and hope it works, but should it fail, you have a problem that needs fixing and a tough job ahead.

The RAM test may reveal intermittent data errors if your RAM or buffers are a little too slow. Remember that the RAM access must occur in slightly less than half of a Dot Clock cycle. My numbers say a 120-ns RAM is OK, but your mileage may differ. In any event, don't fiddle with the LCD panel until the RAM works *perfectly*: it's hard to find bugs when you can't trust your test data.

Once your dots work, recheck the LCD power supply voltages. If you're using the LM337 circuit shown last month, remember that you can destroy a panel with one twist of R65. Set the supply voltage to match your panel's data sheet and adjust the contrast trim pot to midrange.

Next, build a cable to connect the Graphic LCD Interface's 2x13 ribbon cable header to your panel. If you can find a connector for your panel, great! Otherwise just solder the appropriate ribbon cable wires directly to the panel's header and be done with it. I generally tape the cable to the back of the panel for testing, but surely you can come up with a better, more permanent arrangement.

Listing 2—*These definitions combine useful LCD Control Port bit patterns info easily remembered groups. You may want to change the GLCD_OFF and GLCD_ON values to match your panel.*

```
#define GLCD_ENCTR 0x0001 /* enable addr counters */
#define GLCD_DISMUX 0x0002 /* 1 disabled output mux */
#define GLCD_ENDISP 0x0004 /* enable LCD (if used) */
#define GLCD_DISBLINK 0x0080 /* 1 disables blinking */
#define GLCD_BLSLOW 0x0050 /* slowest blinking */
#define GLCD_BLMED 0x0040 /* moderate blinking */
#define GLCD_BLFAST 0x0030 /* fastest blinking */
#define GLCD_BLZERO 0x0000 /* output logic zero */
#define GLCD_BLCLOCK 0x0060 /* output LCD clock */
#define GLCD_BLONE 0x0070 /* output logic one */

#define GLCD_OFF (GLCD_DISMUX | GLCD_DISBLINK)
#define GLCD_ON (GLCD_ENCTR | GLCD_ENDISP)
```

AMXTM

The Real-Time Multitasking Kernel

680x0, 683xx
80x86/88 real mode
80386 protected mode
i960[®] family
R3000, LR330x0
Z80, HD64180

Features

- Full-featured, compact ROMable kernel with fast interrupt response
- Preemptive, priority based task scheduler with optional time slicing
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility eases system construction
- InSight[™] Debug Tool is available to view system internals and gather task execution statistics
- Supports inexpensive PC-hosted development tools
- Comprehensive, crystal clear documentation
- No-hidden-charges site license
- Source code included
- Reliability field-proven since 1980

Count on KADAK.
Setting real-time standards since 1978.

For a **free** Demo Disk
and your copy of our excellent AMX
product description, contact us today.

Phone: (604) 734-2 796
Fax: (604) 734-8114



KADAK Products Ltd.
206 - 1847 West Broadway
Vancouver, BC, Canada V6J1Y5

AMX is a trademark of KADAK Products Ltd.
All trademarked names are the property of their
respective owners.

Photo I--The test pattern on this Toshiba TLY-365-121 640x200 LCD panel is based on Figure 2. The first several dozen dots in each row display the test code's row number (0 through 199) in binary, the row's RAM address, and a diagonal line. The code inserts blinking separator patterns between the data values which appear solid in this photograph.

If you have an LCD backlight inverter, wire it up to the panel using a separate cable. Don't succumb to the temptation of running 100 VAC power through the same cable as your precious sync and data signals..

The final step is a simple ohm meter continuity check: verify the connections from the Graphic LCD Interface circuitry to the LCD panel. This is your last chance to get the power supply voltages on the right pins, so don't blow it now.

The Optrex DMF65 1 640x200 panel has a helpful legend silkscreened on the back of the circuit board:

UP↑

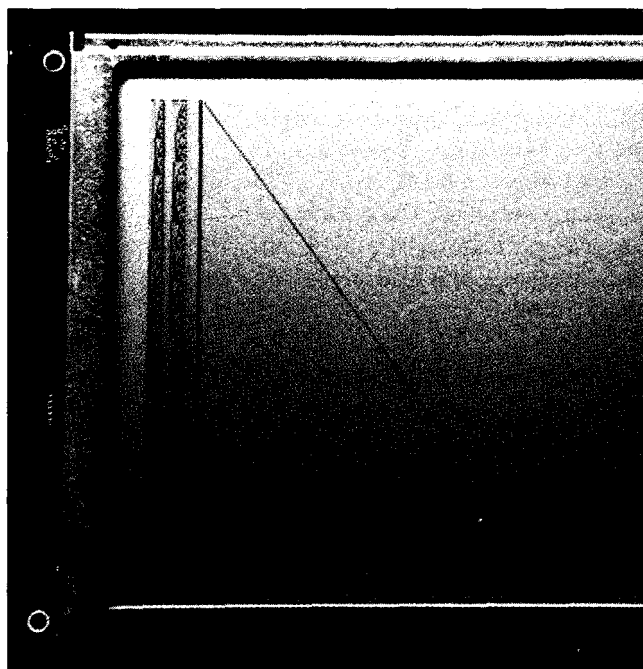
All of the other panels assume you can derive this key datum from the documentation. The remaining tests

help you debug the hardware and screw the panel down correctly: plug it in and let's do some dots..

TESTING: ONE...

Probably the first thing you ask about a graphic LCD panel is, "How big is it?" From the number of rows and columns you can calculate the size of the LCD Refresh RAM in bytes. For example, the DMF65 1 displays 128,000 bits or 16,000 bytes of data.

The Graphic LCD Interface measures LCD panels somewhat differently. Instead of rows and columns, it must know the number of Dot Clocks in each row, the total number of Dot Clocks in a complete



RAM Address 7C60	Test Code Row	Physical LCD Row	
0000	0	1	
00A0	1	2	
		3	
3CA0	97	99	
3D40	98	100	
3DE0	99	101	
3E80	100	102	
7620	197	199	
7BC0	198	200	

Figure I--The Optrex DMF651 has a fairly simple layout. The top row of dots appears to be out of sequence because the test code numbers the rows starting with Row 0 at address 0000. The Graphic LCD Interface hardware produces a Frame Sync pulse when it resets the LCD Address Counters to 0000, so the data immediately before the pulse comes from the highest RAM addresses. The LCD panel displays the data before Frame Sync on the top row, so the test code's Row 199 is on top.

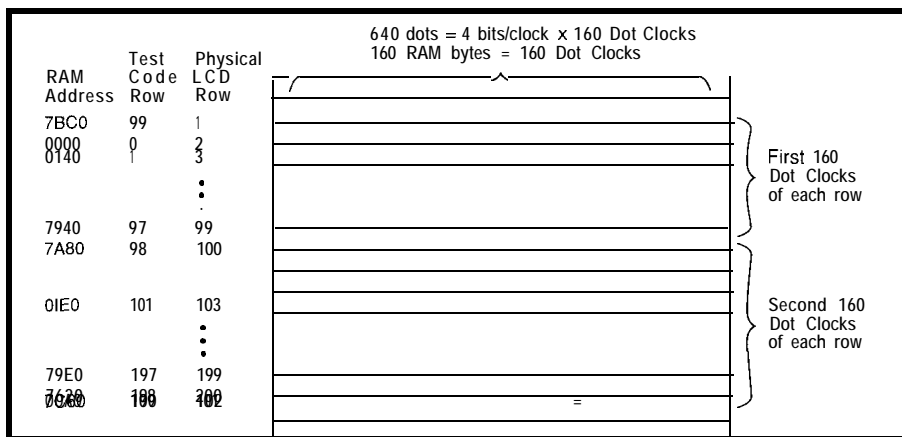


Figure Z—The Toshiba TLY-365-121 splits each logical row of 320 Dot Clocks in two: the first 160 dots appear in the upper half of the panel and the second 160 dots appear in the lower half. The 320 clocks before the Frame Sync pulse appear on physical rows 2 and 102. Photo 1 shows this test pattern in real life.

Listing 3—Panels such as the Opflex DMF651 need only four bits per Dot Clock cycle, so the Graphic LCD Interface can produce blinking by switching between the two nybbles in each byte a few times per second. This routine puts the data bits in the low nybble, then sets the high nybble to produce the desired blink pattern.

```
SetData(Addr,Blink,Data)
WORD Addr:
int Blink:
WORD Data:

    Data &= 0x00FF;          /* force high nybble off */
    switch (Blink) {
    default :
        printf("Invalid blink mode = %u\n",Blink);
    case BLINK_OFF:
        Data |= Data << 4;
        break;
    case BLINK_FG:
        break;
    case BLINK_BG:
        Data |= 0x00FF;
        break;
    case BLINK_INV:
        Data |= (~Data) << 4;
        break;
    }
    poke(GLCD_SEGMENT,Addr,Data);
}
```

Listing 4—The Toshiba TLY-365-121 is a 640×200 panel, but each row has 320 Dot Clocks. The dots in each electrical row appear on two separate physical rows, which complicates the RAMAddr calculation. Once you know where the row begins, however, the bit patterns are familiar.

```
for (Row = 0; Row < 200; Row++) {
    RAMAddr = 320 * (Row % 100) + ((Row > 99) ? 160 : 0);
    for (Col = 0; Col < 160; Col++){
        poke(GLCD_SEGMENT,RAMAddr+Col,0x00); /* clear row */
    }
    SetData RAMAddr+0,BLINK_OFF,Row >> 4); /* show row number */
    SetData RAMAddr+1,BLINK_OFF,Row);
    SetData RAMAddr+2,BLINK_FG,0x09); /* divider */
    SetData RAMAddr+3,BLINK_OFF,RAMAddr >> 12); /* show RAM address */
    SetData RAMAddr+4,BLINK_OFF,RAMAddr >> 8);
    SetData RAMAddr+5,BLINK_OFF,RAMAddr >> 4);
    SetData RAMAddr+6,BLINK_OFF,RAMAddr);
    SetData RAMAddr+7,BLINK_INV,0x09); /* divider */
    SetData RAMAddr+8+(Row >> 2),BLINK_BG,0x08 >> (Row & 0x03));
}
```

frame, and the number of data bits transferred on each Dot Clock. As you saw last month, however, there is only the slightest relation between a panel's physical size and the signals that control it.

The DMF651 accepts four bits on each Dot Clock, so each 640-dot row requires 160 clocks. The panel has 200 row drivers, one for each physical row, so a frame has 32,000 Dot Clock cycles. The dots occupy one nybble in each of the first 32,000 LCD Refresh RAM bytes.

The test code shown in Listing 1 initializes the Graphic LCD Interface hardware for a DMF651, then writes a test pattern into the LCD Refresh RAM. Listing 2 defines the LCD Control Port bit patterns used in the code. The code on the BBS produces several trace outputs that I've removed from these listings to save space.

Recall that both 82C54 channels must begin counting on the same Dot Clock cycle. The first line disables the counters by lowering their Gate inputs. After loading the row and frame lengths, the code enables the 82C54 and sets the blink rate. The Graphic LCD Interface immediately begins sending data, Dot Clocks, and sync pulses to the panel even though the Refresh RAM isn't loaded with data yet.

The test loop iterates 200 times, once for each row. The rows are contiguous in RAM, so the starting addresses are just 160 (decimal) times the row number: 0000, 00A0, 0140, and so on. I set RAMAddr to the start of the current row at the start of each iteration to simplify the rest of the code.

All of the test routines use the LCD's high dot density to display the row number and RAM address in binary in the first few dozen dots of each row. You may need a magnifying glass to read the dot patterns, but each line has an unambiguous identifier! A diagonal line provides an easy visual check that all the rows appear in the right order.

Remember the sound of one hand clapping forehead last month?

Figure 1 sketches the row numbers, RAM addresses, and the diagonal

line for a DMF651. The test code numbers the top row 199 because it's the last one in the buffer, but the panel displays those dots on the first row because they precede the Frame Sync pulse.

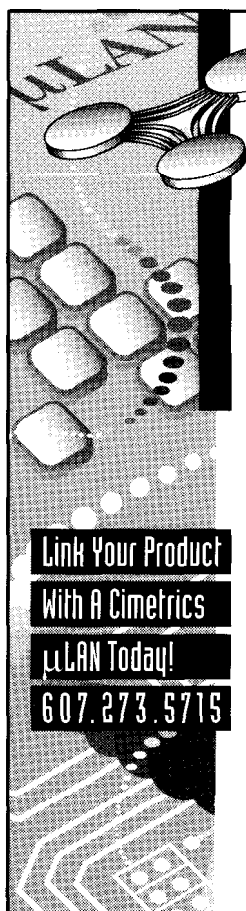
Listing3 is the SetData routine that writes the bit patterns into RAM. I defined only four blinking effects, but you can come up with some true oddities without much effort. The blinking rate is controlled by the initial LCD Control Port setup shown in Listing 1.

The DMF651 is about as simple a panel as you're likely to encounter. The physical dot layout matches the electrical sync signal pattern. Sad to say, that is not generally the case. The other panels can give you a bad case of brain burn..

The Toshiba TLY-365-121 is also a 640x200 panel, but it has 320 Dot Clocks on each row, so the firmware sees it as a 1280x100 array. Listing 4 shows the test pattern code. Notice the difference in 82C54 setup and RAMAddr calculation.

Listing 5-The Matsushita EDM-LG64AA4D panel accepts eight bits on each dot clock, so the firmware sees it as two 640x200 panels. The test code loops through all 400 rows and sets RAMAddr to the start of each one. Align selects the appropriate nybble and the StoreNybble function inserts the data bits. Because the row number now exceeds 255, the firmware uses 16 bits to display it.

```
outpw(GLCD_CTL5, GLCD_OFF); /* disable clocking */
LoadTimer(1, 0x04, 160, I8254_BASE); /* clocks per row */
LoadTimer(2, 0x04, 32000, I8254_BASE); /* total clocks/frame */
outpw(GLCD_CTL5, GLCD_ON | GLCD_BLZERO); /* get bits 0-3 from mux */
*/
for (Row = 0; Row < 400; Row++) {
    RAMAddr = 160 * (Row % 200); /* buffer address */
    Align = (Row >= 200) ? 4 : 0; /* high/low bit alignment */
    for (Col = 0; Col < 160; Col++) {
        StoreNybble(RAMAddr+Col, Align, 0); /* clear our nybble */
    }
}
for (Row = 0; Row < 400; Row++) {
    RAMAddr = 160 * (Row % 200); /* buffer address */
    Align = (Row >= 200) ? 4 : 0; /* high/low bit alignment */
    StoreNybble(RAMAddr+0, Align, Row >> 12);
    StoreNybble(RAMAddr+1, Align, Row >> 8);
    StoreNybble(RAMAddr+2, Align, Row >> 4);
    StoreNybble(RAMAddr+3, Align, Row);
    StoreNybble(RAMAddr+4, Align, 0x08); /* divider lines */
    StoreNybble(RAMAddr+5, Align, RAMAddr >> 12);
    StoreNybble(RAMAddr+6, Align, RAMAddr >> 8);
    StoreNybble(RAMAddr+7, Align, RAMAddr >> 4);
    StoreNybble(RAMAddr+8, Align, RAMAddr);
    StoreNybble(RAMAddr+9, Align, 0x06); /* divider lines */
    StoreNybble(RAMAddr+10+(Row >> 2), Align, 0x08 >> (Row & 0x03));
}
```



Cimetrics

TECHNOLOGY

*Linking Microcontrollers.
Breaking Boundaries.*

The 9-Bit Solution

The Cimetrics Technology 9-Bit Solution is a complete microcontroller network (μLAN) that supports the 8051, 68HC11, 80C186EB/EC, and many other popular processors. The 9-Bit Solution takes full advantage of microprocessor modes built in to microcontrollers. The 9-Bit Solution allows simple and inexpensive development of master/slave multidrop embedded controller networks.

- 8051, 68HC11, 80C186EB/EC compatible
- A full range of other processors supported
- Up to 250 nodes
- 16 Bit CRC error checking with sequence numbers
- Complete source code included

**Link Your Product
With A Cimetrics
μLAN Today!**

607.273.5715

120 West State Street • Ithaca, New York 14850
Ph 607.273.5715 • Fx 607.273.5712

#130

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" 27641128 EPROM programmer, three 8-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.

PROCESSOR

- 80C528-bit CMOS processor w/BASIC-52
- Three 16-bit counter/timers
- Six interrupts
- Much more!

MEMORY

- 48K RAM/ROM, expandable
- Five on-board memory sockets
- Either 8K or 16K EPROM

Input/Output

- Console RS232 - autobaud detect
- Line printer R-232
- Three 8-bit parallel ports
- EXPANDABLE!
- Compatible with 12 BCC expansion boards

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

Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (02) 888-6401

Distributor Inquiries Welcome!

#119

Listing 6—Because 640x400 panels display twice as much data as 200-line panels, both nybbles hold dot data. This routine places four bits into either the upper or lower nybble as controlled by the Align variable. It fetches the byte, clears the selected nybble, inserts the new bits, and then stores the result. Because ISA memory is quite slow, it might be a good idea to use a "shadow" RAM buffer to eliminate the read access.

```
StoreNybble(Addr,Align,Data)
WORD Addr;
int Align;
WORD Data;

WORD OldData;

OldData = peek(GLCD_SEGMENT,Addr) & (0xf0 >> Align);
Data = ((Data & 0x0f) << Align) | OldData;
poke(GLCD_SEGMENT,Addr,Data);
```

The loop still iterates 200 times, once for each visible dot row, but finding the row starting address is more complex. The first 160 bytes appear on one row with the remaining 160 bytes 100 rows further down the panel. Row 99 is on top, with Row 199 in the middle of the panel.

To prove I'm not making this up, Photo 1 shows the actual test pattern, and Figure 2 sketches the layout.

...TWO...

As I mentioned last month, there are two ways to deliver the increased bandwidth needed by 400-line panels: more dots per clock or more clocks per frame. The Graphic LCD Interface can handle either method as long as you set the right bits and jumpers.

The Matsushita EDM-LG64AA-44D accepts eight data bits on each Dot Clock cycle. The LCD Control

port value sends a constant zero to the LCD Data Multiplexer, forcing it to gate only the low nybble to the panel. The high nybble is connected directly to the panel, so it sees the entire LCD Data Latch on each Dot Clock cycle. Listing 5 shows the test code.

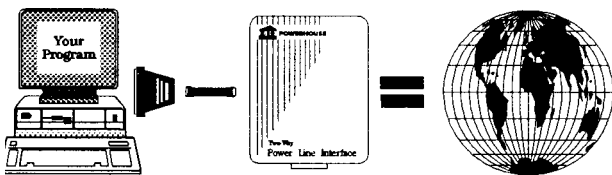
As the loop iterates through 400 visible lines, RAMAddr goes through the same 200 addresses twice. There are no spare bits to provide blinking, so StoreNybble in Listing 6 inserts four data bits in the proper half of the byte as dictated by the Align variable.

The Sharp LM64015T panel uses a double-speed 240-ns Dot Clock, which is selected by jumper JP12. The LCD RAM continues to run at the same 480-ns rate, so the code in Listing 7 sends Dot Clock to the LCD Data Multiplexer, forcing it to switch between the two LCD Data Latch nybbles every 240 ns. The high nybble goes out in the first half cycle, so, surprisingly enough, the dots are laid out left to right.

Because only the LM64015T sees the 240-ns 2x Dot Clock, the rest of

TW523 Power Line Interface Developers Kit

Interface Your Computer To Transmit And
Receive X-10 Codes Over Your AC Power Line.
Two-Way Communication.
Real Time Environment Control.



Kit Includes
TW523, Cable, Interface Connector (S/P)
Documentation. Source Code Supplied in
"C", Pascal, BASIC or Run Time.
Disks 5.25in & 3.5in Format.

\$65.00

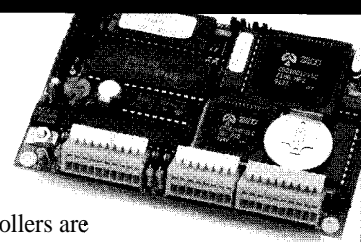


Baran-Harper Group Inc.

Phone (800) 661-6508 (905) 294-6473
Fax (905) 471-3730 BBS (905) 471-6776

LOW COST MINIATURE CONTROLLERS

Save time.
Save money.



Our wide range of C-programmable miniature controllers are ideal for control applications, data acquisition, and test and measurement. Compact and low in price (the Little PLC™ hove is 2"x3" and \$195), these controllers are programmed with our easy-to-use Dynamic C™ development system. Our controllers feature digital I/O, ADCs and DACs, relays and solenoid drivers, RS232/RS485 serial ports, battery-backed memory and time/date clock, LCDs, keypads, enclosures and more!



1724 Picasso Ave.
Davis, CA 95616
916.757.3737
916.753.5141 FAX

24-Hour AutoFAX 916.753.0618. Call from your FAX Request catalog #18.

the Graphic LCD Interface thinks the panel has 160 ordinary 480-ns Dot Clocks on each row and 32,000 clocks per frame. The timing is the same as the DMF651, but blinking must be disabled. JP10 and JP11 should select the double-speed sync signals that are active for only half of the normal Dot Clock cycle.

Unlike the other panels, the LM64015T turns a 1 bit into a transparent dot. The panel starts off opaque and the backlight shines through the on bits. I find this disconcerting, but you can easily display dots either way. The test pattern uses a transparent background with dark data dots.

The LM64148 1 640x480 panel is similar to its 400-line ancestor, although, as I mentioned last month, you'll need more RAM to fill all the dots. GRAPH LCD includes a test pattern if you need it. Running it with only 32K bytes installed will repeat some of the data in the "missing" addresses.

...FOUR...

The Hitachi LM215XB 480x128 panel is a horse of a somewhat different stripe: it needs half the data, runs at half the speed, and sprinkles dots all over RAM. Listing 8 shows the test code you'll need. Remember to install the half-speed Dot Clock/2 jumper in JP12 for this panel.

Each of the 64 rows has 240 half-speed Dot Clocks, so the 82C54 thinks this panel has 480 Dot Clocks per row. The panel uses the data from odd-numbered RAM addresses, so there are $2 \times 240 \times 64 = 30720$ regular 480-ns Dot Clocks in one frame.

The test code iterates through all 128 visible rows, but displaying the row number requires the routines shown in Listing 9 because RAM data bits are not contiguous on the panel. The dot addressing complexity makes this code considerably harder to follow!

The `WriteBit` function computes `RAMAddr` based on the dot's row and column location. The `Align` variable then specifies the bit location within that byte, which obviously depends on how you wire the Graphic LCD Interface's data bits to the LM215 connector. The `Blank` parameter

Listing 7-The Sharp LM64015T runs at twice the speed of the other panels, so it requires a nybble on each half-cycle of the 480-m Dot Clock. The LCD Data Multiplexer switches between the two nybbles and a jumper on the Graphic LCD Interface board routes the double-speed 2x Dot Clock to the panel. Note that the data is inverted: a 1 bit is transparent rather than opaque.

```
outpw(GLCD_CTL5, GLCD_OFF);          /* disable clocking */
LoadTimer(1, 0x04, 160, I8254_BASE); /* Dot Clocks/row */
LoadTimer(2, 0x04, 32000, I8254_BASE) /* total clocks/frame */
outpw(GLCD_CTL5, GLCD_ON | GLCD_BLCLOCK); /* alternate at clock rate */
for (Row = 0; Row < 400; Row++) {
    RAMAddr = 160 * (Row % 200) + ((Row > 199) ? 80 : 0);
    for (Col = 0; Col < 80; Col++) {
        poke(GLCD_SEGMENT, RAMAddr + Col, 0xff) /* 1 = clear */
    }
}
for (Row = 0; Row < 400; Row++) {
    RAMAddr = 160 * (Row % 200) + ((Row > 199) ? 80 : 0);
    poke(GLCD_SEGMENT, RAMAddr + 0, ~(Row >> 8));
    poke(GLCD_SEGMENT, RAMAddr + 1, ~(Row & 0x00FF));
    poke(GLCD_SEGMENT, RAMAddr + 2, ~0x99); /* divider lines */
    poke(GLCD_SEGMENT, RAMAddr + 3, ~(RAMAddr >> 8));
    poke(GLCD_SEGMENT, RAMAddr + 4, ~(RAMAddr & 0x00FF));
    poke(GLCD_SEGMENT, RAMAddr + 5, ~0x99); /* divider lines */
    poke(GLCD_SEGMENT, RAMAddr + 6 + (Row >> 3), ~(0x80 >> (Row & 0x0007)));
}
```

controls the corresponding bit in the high nybble.

`WriteByte` simply calls `WriteBit` for each bit in a byte-sized value. There is considerable overhead involved in this process, but it's still reasonably perky...and will get better with assembler!

The LM215 test pattern displays the row and `RAMAddr` values in all four panel quadrants to verify all the data bits. If you have one of these panels, examine the dots (with a magnifying glass!) to see that the same RAM addresses appear four times.

Listing 8-The Hitachi LM215XB uses a half-speed 960-m Dot Clock and directs the four data bits to separate quadrants. The panel uses only the data in the odd-numbered RAM addresses, but this code duplicates the dots in the even addresses to simplify debugging.

```
outpw(GLCD_CTL5, GLCD_OFF);          /* disable clocking */
LoadTimer(1, 0x04, 480, I8254_BASE); /* Dot Clocks/row */
LoadTimer(2, 0x04, 30720, I8254_BASE); /* total clocks/frame */
outpw(GLCD_CTL5, GLCD_ON | GLCD_BLFAST); /* enable & set blinking */
for (Row = 0; Row < 64; Row++) {
    for (Col = 0; Col < 240; Col++) {
        RAMAddr = 2 * (240 * Row) + Col;
        poke(GLCD_SEGMENT, RAMAddr + Col, 0x00); /* unused */
        poke(GLCD_SEGMENT, RAMAddr + Col + 1, 0x00);
    }
}
for (Row = 0; Row < 128; Row++) {
    RAMAddr = 2 * (240 * Row) + 1;
    StoreByte(Row, 0, BLINK_OFF, Row);
    StoreByte(Row, 8, BLINK_FG, 0x81);
    StoreByte(Row, 16, BLINK_OFF, RAMAddr >> 8);
    StoreByte(Row, 24, BLINK_OFF, RAMAddr);
    StoreBit(Row, 32 + Row, BLINK_FG, 1);

    StoreByte(Row, 240 + 0, BLINK_OFF, Row);
    StoreByte(Row, 240 + 8, BLINK_BG, 0x7E);
    StoreByte(Row, 240 + 16, BLINK_OFF, RAMAddr >> 8);
    StoreByte(Row, 240 + 24, BLINK_OFF, RAMAddr);
    StoreBit(Row, 240 + 32 + Row, BLINK_FG, 1);
}
```

..AND MORE!

That covers most of the interesting LCD panels in my stash, but doesn't come close to exhausting the market. You can probably adapt the Graphic LCD Interface to drive whatever panel you've got, although

I'm sure there are some that just won't work.

Assuming you can get the power supply and signal leads connected correctly, GRAPH LCD D will help you figure out the panel's sync requirements. The panels are tolerant of

incorrect signals as long as you don't exceed their voltage specs, so try a few experiments. In fact, once you get a panel working, lie to GRAPH LCD just to see what the results look like.

CAUTION: even though the panels won't blow up if you apply the wrong sync signals, a DC bias will degrade the liquid crystal material and the panel's transparent electrodes. Turn the power off while you dope out the jumpers and clocking. Don't use incorrect signals longer than it takes you to realize that things just aren't working right.

So.. if that doesn't get your juices flowing, you really are reading the wrong magazine. Heat up those soldering irons, get those compilers whirring, and let's see some dots!

Next month I'll explore dot plotting using Conway's Game of Life to generate the patterns. If everything goes according to plan, you can turn your PC and LCD into a piece of decorative art..

RELEASE NOTES

GRAPH LCD D appeared on the BBS last month, but I won't post another copy so Ken doesn't have to keep two files up-to-date. If I come across any other panels, I'll expand GRAPH LCD D to check them out.. and you can always upload your efforts, too. 📁

Ed Nisley, as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of the Computer Applications Journal's engineering staff. You may reach him at ed.nisley@circellar.com or 74065.1363@compuserve.com.

SOURCE

Pure Unobtainium has the complete Firmware Development Board schematic, as well as selected parts. Write for a catalog: 13109 Old Creedmoor Rd., Raleigh, NC 27613. Phone or fax (919) 676-4525.

413 Very Useful
414 Moderately Useful
415 Not Useful

Listing 9—The Hitachi LM215XB panel has a peculiar bit arrangement, so these two routines are more complex than you'd expect. The low nybble of the odd numbered RAM addresses holds the normal dots, the high nybble holds the blinking dots, and each bit drives a separate quadrant. The StoreByte function writes all eight data bits into the display buffer so they appear as consecutive dots.

```
StoreBit(Row,Col,Blink,Data)
WORD Row;
WORD Col;
WORD Blink;
WORD Data;

WORD Temp;
WORD Mask;
WORD RAMAddr;
WORD Align;

RAMAddr = 2*(240*(Row%64)+(Col%240)); /* even address */

Align = ((Col > 239) ? 2 : 0) + ((Row > 63) ? 1 : 0);
Mask = 0x0001 << Align;
Data &= 0x0001;

Temp = peek(GLCD_SEGMENT,RAMAddr + 1); /* fetch the old bits */
Temp &= ~(Mask | (Mask << 4)); /* strip target & blink */
Temp |= Data << Align; /* insert new data bit */

switch (Blink) {
default :
    putstr("Invalid blink mode in Storebit: %u\n",Blink);
case BLINK_OFF
    Temp |= Data << (Align + 4); /*no blink duplicate the data bit */
    break;
case BLINK_FG: /* zero the blink bit, already done */
    break;
case BLINK_BG:
    Temp |= Mask << 4; /* set the blink bit */
    break;
case BLINK_INV:
    Temp |= ((~Data) & Mask) << 4; /* inverse of data bit */
    break;

poke(GLCD_SEGMENT,RAMAddr,Temp);
poke(GLCD_SEGMENT,RAMAddr+1,Temp);

StoreByte(Row,Col,Blink,Data)
WORD Row;
WORD Col;
WORD Blink;
WORD Data;

StoreBit(Row,Col,Blink,Data >> 7);
StoreBit(Row,Col+1,Blink,Data >> 6);
StoreBit(Row,Col+2,Blink,Data >> 5);
StoreBit(Row,Col+3,Blink,Data >> 4);
StoreBit(Row,Col+4,Blink,Data >> 3);
StoreBit(Row,Col+5,Blink,Data >> 2);
StoreBit(Row,Col+6,Blink,Data >> 1);
StoreBit(Row,Col+7,Blink,Data >> 0);
```

Designers Spell Relief "IPC"

If you've ever been involved in an electronics project from beginning to end, you know factors such as PCB layout, packaging, and wiring are important obstacles that have to be tackled. Jeff clues us in on an organization specializing in this.

FROM THE BENCH

Jeff Bachiochi

a

round 1957, (can you remember back that far?) a few bright individuals became frustrated with the electronics industry. In an effort to assure quality and equality within their trade association, they formed a cooperative group known as the IPC. Although based in the U.S., the *Institute for Interconnecting and Packaging Electronic Circuits* has an international membership representing manufacturers, assemblers, and users of PCBs, flat and discrete wiring, and hybrid circuits.

The IPC's success in its effort to aid industry has led industry itself to look to the IPC for guidance. To promote touchstone, the IPC provides a full range of standards, guidelines, technical reports, and video presentations to assist the industry, government agencies, and educational institutions.

IPC PUBLICATIONS

The IPC-S-100 is a set of standards and specifications covering electronic interconnection materials, performance, and reliability. Collected within the set are individual documents pertaining to terms and definitions, standards for dimensioning, and project documentation. Other subjects cover computer-aided technology, rigid laminates, flexible dielectric materials, coatings, and joining methods and techniques. These documents also cover related information on connectors, flat cables, and alternate interconnection methods.

IPC Technology Manuals are definitive sources of information

spanning the entire electronic interconnection technology. Each manual describes a particular facet of the industry. Areas as diverse as PCB design, materials inspection, phototool generation and measurement techniques, chemical handling safety, and SMT guidelines can all be found in these publications. To aid in inspection evaluation, slide sets and photo-handbooks clearly reveal what to look for. These guides are available for both bare and assembled PCBs.

Artwork for Standard Test Boards is available on mylar film. These artworks, useful in testing your companies' materials and processes, could be helpful if you are in this phase of the industry. IPC also produces educational and technical video training films. These can be used to help you keep up-to-date with the latest developments in your area of interconnection technology, or to fill in the gaps in understanding other processes. These publications provide complete coverage of many areas from PWB (printed wiring board) materials preparation to hand soldering techniques. A list of these publications is in Table 1.

IPC DESIGNER'S COUNCIL

To remain effective, the IPC has opened its door to you and me through an organized group known as the Designer's Council. Starting in 1992 with 29 local chapters, the Designer's Council is open to designers and suppliers alike, on both an individual and corporate level.

LOCAL LEVEL

I belong to the Southern New England Chapter of the IPC Designer's Council. This chapter meets on a bimonthly basis at various locations throughout southern Connecticut. Corporate sponsors take turns hosting our 23-hour meetings, which often include dinner. A typical agenda consists of a brief business meeting, followed by a guest speaker, and finishes up with an open forum discussion of work-associated problems and their solutions.

The yearly dues of \$25 and an additional charge of \$7 per meeting

IPC-T-50	Terms and Definitions for Interconnecting and Packaging Electronic Circuits
IPC-SC-60	Post Solder Solvent Cleaning Handbook
IPC-AC-62	Post Solder Aqueous Cleaning Handbook
IPC-PC-90	General Requirements for implementation of Statistical Process Control
IPC-QS-95	General Requirements for Implementation of ISO-9000 Quality Systems
IPC-L-108	Specifications for Thin Metal-clad Base Materials for Multilayer Printed Boards
IPC-L-112	Standard for Foil Glad, Composite Laminate
IPC-FC-203	Specifications for Flat Cable, Round Conductor, Ground Plane
IPC-D-249	Design Standard for Flexible Single- and Double-sided Printed Boards
IPC-D-275	Design Standard for Rigid Printed Boards and Rigid Printed Printed Board Assemblies
IPC-D-300	Printed Beard Dimensions and Tolerances
IPC-D-322	Guidelines for Selecting Printed Wiring Board Sizes Using Standard Panel Sizes
IPC-D-325	Documentation Requirements for Printed Boards
IPC-D-326	Information Requirements for Manufacturing Electronic Assemblies
IPC-NC-349	Computer Numerical Control Formatting for Drillers and Routers
IPC-D-356	Bare Board Electrical Test Information in Digital Form
IPC-DG-395	Guide for Digital Descriptions of Printed Board and Phototool Usage per IPC-D-350
IPC-01-645	Standards for Visual Optical Inspection Aids
IPC-QL-653	Qualifications of Facilities that Inspect/Test Printed Boards, Components, and Materials
IPC-ML-960	Qualification and Performance Specification for Mass Lamination Panels for Multilayer Printed Boards
J-STD-001	Requirements for Soldered Electrical and Electronic Assemblies
IPC-D-330	Guidelines for layout, design, and packaging of electronic interconnections
IPC-PD-335	Electronics Packaging Handbook
IPC-MI-660	Test Methods for Inspection and Evaluation of Incoming Raw Materials
IPC-PE-740	Troubleshooting Guide for Printed Board Manufacture and Assembly
IPC-D-390	General Overview of Computer-aided PCB Design Process
IPC-C-406	Design and Application Guidelines Surface-mounted Connectors
IPC-CS-70	Guidelines for Chemical Handling, Safety, in Printed Boards and Manufacturing
IPC-DR-572	Drilling Guidelines for Printed Boards
IPC-A-600	Acceptability of Printed Boards
IPC-ET-652	Guidelines and Requirements for Electrical Testing of Unpopulated Printed Boards
IPC-R-700	Suggested Guidelines for Modification, Rework, and Repair of Printed Boards and Assemblies
IPC-CM-770	Component Mounting Guidelines for Printed Boards
IPC-SM-780	Component Packaging and Interconnecting with Emphasis on Surface Mounting
IPC-SM-782	Surface Mount Land Patterns (Configurations and Design Rules)
IPC-TR-460	Troubleshooting Checklist for Wave Soldering Printed Wirina Boards
IPC-QE-605	Printed Board Quality Evaluation Handbook
IPC-QE-615	Printed Board Assembly Quality Evaluation Handbook
IPC-A-36	Cleaning Alternatives

Table 1—IPC Technical Publications (abbreviated list).

help to offset the costs for the chapter's newsletter, invited speakers, and refreshments. As of this writing, this chapter has a total of 65 paying members. The membership covers a diverse cross-section of the industry from chemical and laminate suppliers to design houses and product manufacturing. About 10% of the members of this chapter are *not* corporate sponsored.

FOR YOU?

From absolute beginner to Senior Designer, everyone learns through interaction and discussions with other chapter members. It seems there is always someone among us who has experienced the same problem you're facing and can suggest corrective

action. The solutions might be as easy as changing the solder mask type or as radical as changing vendors. The Designer's Council promotes and disperses information regarding the current activities and current developments in design technology.

Speakers, seminars, and workshops are presented by request from the membership. Do you want to learn more about flexible printed wiring, layout guidelines for automatic assembly, artwork scanners providing Gerber output, or even visit a "fab" house? These are typical requests for presentations at future meetings. If you become a chapter member, your input formulates these programs.

As a growing force in the industry, the Designer's Council is preparing a

Certification and Accreditation Program. Now employers and designers alike will be able to establish standardized qualifications for Design Level Expertise. This program will cover more than 20 different areas of design. Each area, or *Focus Module*, will have multilevel experience ratings. The designer can attain these various levels either through education or through work experience. See Table 2 for more details.

Being a member of the Designer's Council allows you access to the IPC Library and Standards. Copies of all of the standards, specifications, and literature are available to you at a discounted rate (document costs range from \$15 to several hundred). If you have a specific area of expertise you

Core Trainina Module

Type I

Associate Designer

Type II

Designer

Senior Designer

Principal Designer

Focus Modules


- M1— Single and double-sided rigid boards
- M2— Single and double-sided flexible boards
- M3— Basic CAD development including libraries
- M4— Power supply printed boards
- M5— Multilayer printed boards without blind and buried vias
- M6— Multilayer printed boards with blind and buried vias
- M7— Impedance control/high-speed requirements
- M8— Experience computer-aided design techniques
- M9— Multilayer constraining core boards without blind and buried vias
- M10— Multilayer constraining core boards with blind and buried vias
- M11— MCM-L
- M12— MCM-C
- M13— MCM-D
- M14— Ceramic printed boards
- M15— Hybrids
- M16— Molded printed boards
- M17— Three-dimensional circuits
- M18— Discrete wiring
- M19— Polymer circuits
- M20— Radio frequency printed boards
- M21— Product Safety

Table 2—The Core Training Module establishes designer competency and level of proficiency, while Focus Modules specify specific areas of expertise.

may wish to participate on any of the committees which review and/or update the IPC standards. As you can well imagine, keeping pace with this changing industry is a must. These tools will support your efforts to remain an effective designer by allowing your products to approach zero defects through quality control.

NOW IT'S YOUR TURN

Can you afford to slip behind today's technologies? Will accreditation

improve your position in the marketplace? Are you willing to share your knowledge with others? Do you wish to impact the proposed design-related standards? Get connected. Do yourself a favor and join a local chapter of the Designer's Council. 

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

CONTACT

Institute for Interconnecting and Packaging Electronic Circuits (IPC)
(708) 677-2850

I R S

416 Very Useful
417 Moderately Useful
418 Not Useful

CONVERSE WITH YOUR PC and receive an intelligent reply!



has Artificial intelligence specialist Joseph Weintraub won the Loebner Prize for Artificial Intelligence three years in a row. Weintraub is president of Thinking Software, Inc. In 1991, at the Boston Computer Museum, his PC Therapist became the first program in history to pass a limited Turing Test and win the Loebner Prize for Artificial Intelligence. PC Therapist convinced five out of ten judges conversing with it that it was a human-not a computer program! Then in 1992, PC Professor, a program that talked about Women's

Lib, won the Loebner Prize again. Finally, in of December last year, Joseph Weintraub's PC POLITICIAN told Clinton a thing or two and won the prize for the third year in a row. PC POLITICIAN asks the provocative question, "Are you a good solid conservative or a liberal piece of fruit?"

Now all this award-winning technology is available to run on your own PC. PC Therapist III is text only, and is \$59.95. In the animated PC Therapist IV, a clever bearded Therapist moves his eyes and mouth as he talks through your PC Speaker... he will have you howling with laughter in no time for \$69. For the smooth-talking silicon buddy you've always wanted add \$18.95 for the SoundBlaster or Covox version!

PC Politician is great for Presidents, Lawyers and Speech Writers! Yes, Bill, we have a free copy just for you...please drop us a note for your FREE PRESIDENTIAL COPY. All you other clods & freeloaders send \$119.95.

THINKING SOFTWARE, INC. THE AI SOURCE

46-16 65TH PLACE, DEPT. SXS2 1. WOODSIDE, NY 11377

AMEX
VISA, MC (718) 803-3638 • Fax (718) 898-3126

LOW COST DEVELOPMENT TOOLS

We have a complete line of 'C' compilers (MICRO-C) for: 68HC08, 6809, 68HC11, 68HC16, 8051/52, 8080/85/Z80, 8086 and 8096 processors. Cross assemblers for these plus others. Source code and porting packages are available!

Development Kits: \$99.95 + s&h

Includes C compiler, Cross assembler, ROM Debug monitor, Library source code, Editor, Telecomm program and everything else you need to do 'C' and Assembly language software development for your choice of processor.

Emily52, a high speed 8052 simulator: \$49.95 + s&h

High speed (500,000 instructions/sec on 486/33). Hardware emulation mode accesses real ports/timers on your target system. Includes PC hosted 'in circuit' debugger.

BD52, a complete 8052 Development System: \$249.95+ s&h

Everything you need for 8052 development, including:

Hardware: 8032 based single board computer with 32K ROM, 32K RAM, RS-232 port, Hardware debug support.

Software: DDS MICRO-C Developers Kit, EMILY52 Simulator, PC Hosted "in circuit" debuaaer and kernal in ROM.

Loaded BD52: \$299.95 + s&h --

As above, with extra 32K RAM, 4 A/D + 1 D/A, 2K EEPROM, 7 line relay driver, watchdog and power monitor.,

Call or write for our free catalog.

DUNFIELD DEVELOPMENT SYSTEMS



P. O. Box 31044
Nepean, Ontario Canada
K2B 8S8

Tel: 613-256-5820 (1-5pm EST, Mon - Thu)
(BBS & Catalog Requests 24 Hour with Touch Tone)
Fax: 613-256-582 1 (24 Hour)

Update Your Design On the Fly

SILICON UPDATE

Tom Cantrell

Here's a new trend sweeping through the cubicles of Silicon Valley.

However "In-System Programmability" is a mouthful that doesn't make for zippy ad copy. What's needed is a catchy phrase that grabs the attention of designers and conveys the benefits of the new approach. OK, what would you call something both harder than software yet softer than firmware?

Hmm, how about "Limpware!"

That giant sucking sound you hear is marketing types nationwide losing lunch-martinis and all. OK, maybe someone can come up with a more potent name like "Flexware" or "Versaware."

Whatever you call it, it's worth trying to understand, and exploit, the concept before your competitors do.

BLESS THE EPROM

The saga starts nearly twenty years ago with the invention by Intel of the EPROM. I can still remember the good old 1702 (Figure 1) whose specs-256x8, 1- μ s access time, +5 and -9-V supplies, 48-V programming, and so forth-seem laughable by today's standards.

Nevertheless, as the first erasable, nonvolatile semiconductor memory, we owe the 1702 and Intel quite a debt of gratitude. The EPROM enabled the widespread adoption of the then nascent microprocessor by small companies that, though now mostly long gone, pioneered the "PC."

Even the flakiest garage shop could afford a stable of EPROMs and a dryer bulb (the "germicidal" bulbs happened to put out the right wave-

length UV). The EPROM, along with the microprocessor and DRAM, were the first shots fired in the "IC revolution" that now puts the equivalent of yesterday's mainframe on our desks (and soon in our pockets).

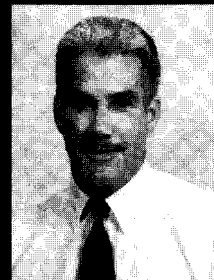
BEYOND FIELD PROGRAMMABILITY

Despite its benefits, the virtues of field programmability are mainly economic, not functional. That is, you can't do anything with an EPROM or PAL that can't be done with a ROM or a gate array, it's just a heck of a lot less nerve wracking.

In-system programmability is another story, referring as much to a philosophy of design as to any particular technology (of which there are many). The idea is to make "softer" systems whose functionality isn't fully determined until long after they've left the factory.

Furthermore, once in the hands of the end user, these systems may continue to evolve, and they will be able to dynamically update their functionality. "Evolve" means both adding new features and/or fixing ones that never worked right in the first place.

A well-known example is the cellular phone in which the programming of basic characteristics, such as the type of service (vendor, roaming, etc.) not to mention the phone number itself, is done by the retailer instead of the factory.



We've all come to appreciate the

convenience of updating firmware or logic arrays by erasing and reprogramming the EPROM or PAL. A new batch of chips is now showing up that can be updated while still in the circuit.

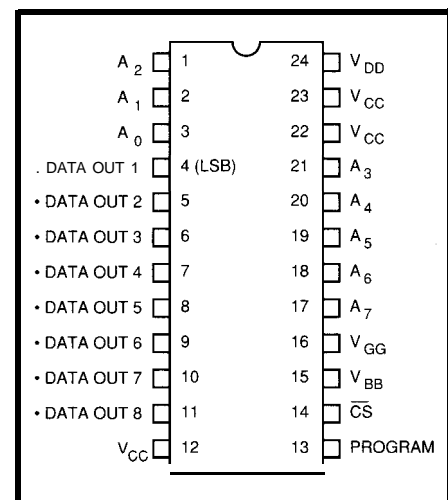


Figure 1--Released in the '70s, the 1702 EPROM pioneered the byte-wide package. Note the non-JEDEC type pinout, which didn't appear until the 2708.

More illustrative of the concept are the recent offerings by some modem suppliers of V.32bis units that can "upgrade themselves" over the phone line to V.32terbo (19.2k) specs. It turns out that the two standards can be implemented by common hardware, with only a DSP software change needed to switch from one to the other.

Type in your credit card

information and voilà, you've got a "new" modem without even yanking a single cable.

In-system programmability also has a number of practical benefits that, though seemingly mundane, save time or money—and there's nothing mundane about that. Put it this way: the job you save may be your own!

Obviously, you don't need a "programmer" (those high-volume gang jobs aren't cheap) since the system itself usurps that role. However, the savings go deeper than just the cost of a programmer itself.

First, consider the handling associated with a mere "field programmable" device. In a typical company, the flow goes something like...

- *Receive the blank parts into inventory
 - Pull the parts from inventory and detube the blank parts
- *Hand load the programmer and burn/label the parts
- *Unload the programmer
 - Chuck any parts that got mangled along the way
 - Retube and restock the programmed parts
- *Pull the programmed parts from inventory
 - PCB assembly

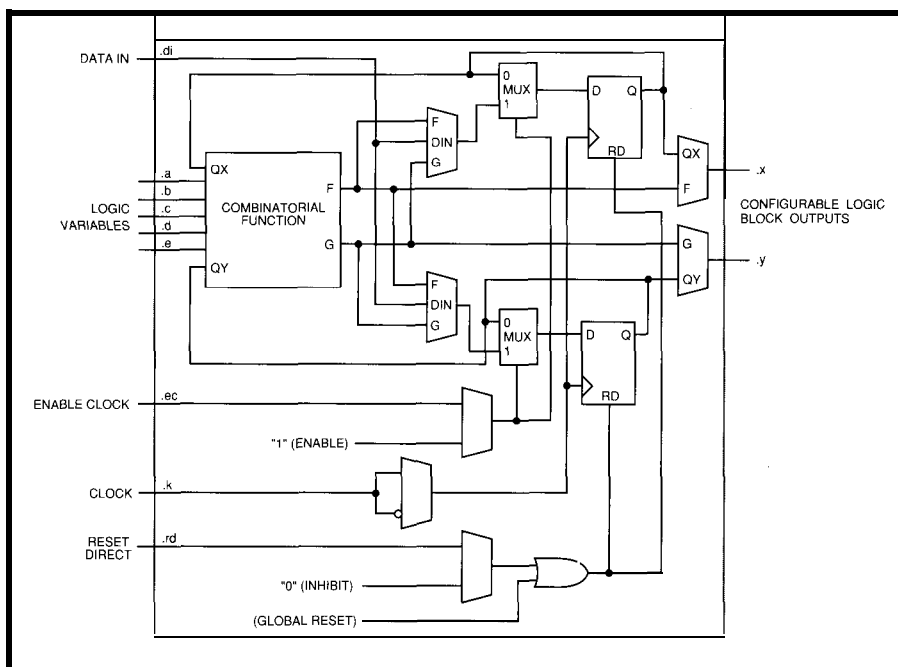


Figure 2—At their core, Xilinx LCAs (Logic Cell Arrays) rely on a table look-up concept to implement combinatorial logic.

As you can imagine, this starts to get tiring, not to mention expensive, especially when the number of different boards and chips is large. The problem is further exacerbated by the trend towards ever more tiny, fragile, high-pin-count, surface-mount chips which are particularly difficult to handle and easy to damage.

By contrast, an in-system programmable part eliminates almost all of the handling. The parts can go directly from the loading dock onto the PCB, with programming deferred to the final phases of system test.

soft design. The best example of this is the "hardware emulators" (pioneered by companies like Quickturn) which are increasingly playing a pivotal role in speeding megatransistor IC designs.

Now that you've got the idea, let's take a look at some of the chips and design issues surrounding the concept.

MARKETING MYOPIA

To my mind, the modern era of in-system programmability was kicked off by Xilinx over five years ago with the introduction of their SRAM-based logic-cell arrays. Indeed, these devices—with variants from 1k to 10k+ gates at about a penny per gate—serve as the guts of the aforementioned Quickturn systems (Photo 1).

It's ironic that the parts are usually saddled with the FPGA or FPLD (Field Programmable Gate Array/Logic Device) moniker that might more appropriately be applied to "harder" EPROM-based devices like those from Altera and AMD.

Maybe they should have called it "GASP" (which would stand for Gate Array System Programmable) which, besides more accurately describing the chip, gives the

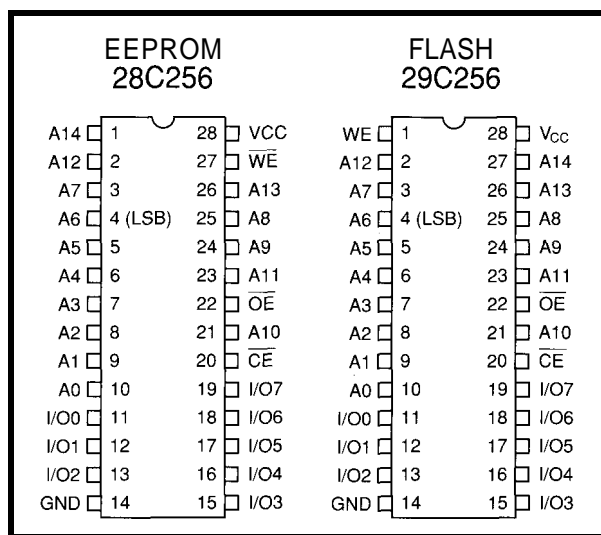


Figure 3—The Atmel 28/29C256 EEPROM/Flash parts have almost identical pinouts and their functionality is pretty much the same, leaving one to ponder if the differentiation is mostly marketing.

long suffering marketing guys something to work with. How about, "System designers, now is the time to make your choice: our last GASP, or yours!" Pretty spiffy, eh?

Actually, the Xilinx parts are coarse-grained LCAs (Logic Cell Arrays) rather than a fine-grained gate array, but SPLCA makes even the folks that came up with PCMCIA seem smart, and the acronym would surely do a number on a long-suffering marketer's digestion.

In fact, it's possible the Xilinx choice of SRAM—a fundamentally in-system programmable technology—had more to do with speed, testability, and package (no window) cost than any great marketing concept. At their core (Figure 2), Xilinx chips rely on a look-up table concept to implement combinatorial logic (for example, a five-input logic block with two outputs can be implemented as a 32x2-bit memory). As you're probably well aware, high-speed SRAMs usually offer much faster access times than EPROMs (i.e., 10 ns vs. 100 ns).

I remember attending a very early Xilinx seminar in which the presenter defensively pleaded that "having to" initialize the parts wasn't "that bad." Not that he didn't have any reason to be on the defensive since I observed widespread skepticism bordering on hostility in the audience of mostly hardware types.

Being perhaps more software savvy, I had less fear and loathing of the idea. Especially if a microprocessor is included in the design, why is initializing a Xilinx chip any different or harder than initializing the typical I/O chip? Perhaps fearing the skeptics, Xilinx took great pains to make the chips easy to set up, offering both serial access (via a microprocessor or serial EPROM) and parallel access (via a microprocessor or parallel EPROM) while allowing daisy-chaining of multiple devices to boot.

They also went to a lot of effort, both in circuit design and testing, to reassure the paranoid that alpha particles or other spurious events wouldn't zap the SRAM bits, rendering the IC witless. To me, the bits in one chip are pretty much as important as

those in any other. Is losing a bit in a Xilinx chip any more likely than losing one in, for example, the CPU register file—and which is worse?

Anyway, having seen the tragedy of ROM and ASIC deals gone sour, I always felt that "having to" initialize really should be viewed as "getting to" initialize. Today, what was once thought a "problem" is now being rightfully and successfully retromarketed as a "feature."

So, look for all the programmable logic suppliers to jump on the in-system bandwagon (even those who originally poooh-pooohed the idea) and be sure to give Xilinx credit for sticking with the concept through thick and thin.

FLASH OF INSPIRATION

Unless you've been on another planet for the last few years, by now you've heard the various pitches for Flash memory and how it will change your life.

Long-time readers know I'm a little skeptical about claims that Flash will replace disk drives. Its density is still too low and its price too high to seriously consider it as an option. Few

people will, for example, run Windows off a Flash disk.

On the other hand, the overrating of the disk replacement concept seems offset by a lack of awareness of other potential uses.

In fact, replacing an EPROM with a nearly drop-in Flash or EEPROM replacement can easily take any system 90% of the way to in-system programmability.

By the way, you may be wondering just what the difference is between Flash and EEPROM! In fact, there isn't much difference—other than that created in your mind by those clever marketing folks. Why, it's enough to make me want to buy them all (a new lunch!

Consider the Atmel AT28C256 (EEPROM) and AT29C256 (Flash) chips. As you can see in Figure 3, the pinouts are distinguished only by the swapping of A14 and WE* (go figure). Otherwise, the main differentiating factor is that an EEPROM is byte- or page-writable, while a Flash is only page-writable (a page is 64 bytes for these chips).

For the Atmel chips, the process and basic cell design are the same with

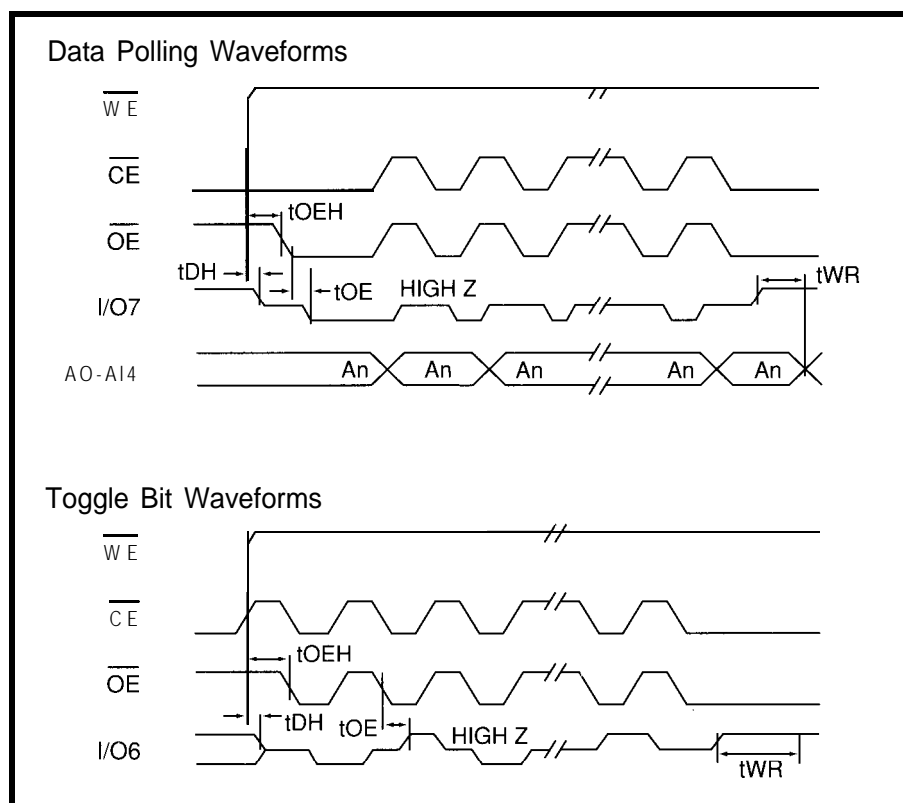


Figure 4—The Atmel chips offer two ways to determine when the page-write is complete: data polling and toggle bit.

the cost difference largely related to the extra circuitry an EEPROM needs for individual byte-write. Furthermore, this particular EEPROM includes ECC circuitry to satisfy Pentagon demands. The additional circuitry explains the price difference, which is \$13.25 versus \$9.25 (100-piece price, 150 ns, plastic, commercial temperature).

For purposes of in-system programmability, page write is fine, so the Flash is probably the way to go. By the way, the Atmel chips only need 5 V to run, and recently released components will run as low as 3 V. Watch out for others that call for a second supply (typically 12 V) during writes (ugh)

During a read operation, these chips look just like an SRAM, relying on the usual mix of signals consisting of CE* (Chip Enable), WE* (Write Enable), and OE* (Output Enable). You can rely on these devices to return the data in 100–150 ns or so depending on the particular chip.

Since the EEPROM also includes page-write, write operations are also similar with both devices relying on a

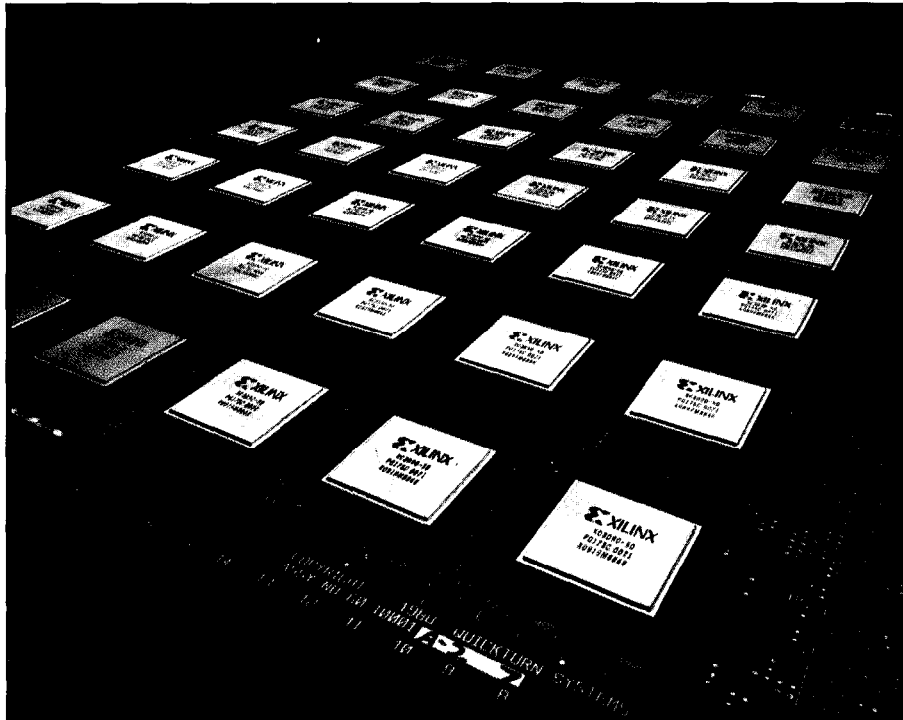
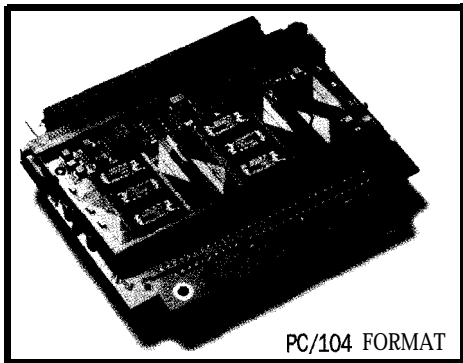


Photo I-In-system prototyping, such as allowed by the Quickturn Hardware Emulator board, is increasingly playing a pivotal role in speeding megatransistor IC designs.

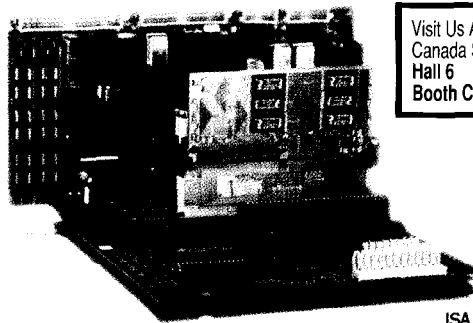
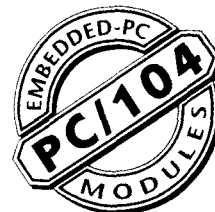
page buffer (64 bytes). The way it works is that individual bytes are written to the buffer (they don't have

to be written in any particular order) in an SRAM-like manner. An on-chip timer monitors the time between byte-

25 MByte **FlasHD** Memory Module



"The FlasHD card allows a system to boot without mechanical drives, execute applications and write up to 25MBytes of non-volatile storage."



Visit Us At:
Canada Stand
Hall 6
Booth C.06

CeBIT '94
HANNOVER

- ▶ Hard Drive replacement; supports M-Systems
- ▶ Non-volatile Flash™ memory board
- ▶ Self Boot mechanism and PC auto detection
- ▶ Available with up to 25MBytes of storage
- ▶ Read/write memory system
- ▶ "Boot Block" of 256KBytes allows DOS in ROM
- ▶ Byte writable, 64KByte sector erasable
- ▶ On-board DC to DC converter for single +5 volt operation

"For the OEM with Embedded PC Applications"

megatel"

(416) 245-2953

For a list of our international distributors, please contact our head office at:

megatel computer corp.
125 Wendell Ave.
Weston, Ont. M9N 3K9 Canada
Fax: (416) 245-6505

Flash is a trademark of Intel Corp. and Megatel is a registered trademark of Megatel Computer Corp.

writes to the buffer and if it exceeds 150 μ s, a page-write is started. The kicker is that buffer bytes not written will be cleared (to FFh) on a Flash, but unchanged on an EEPROM.

The Atmel chips offer two ways to determine when the page-write is complete (typically 3-10 ms depending on the chip). These two methods are data polling and toggle bit (Figure 4). In the former, a read of the last byte written will return the complement of D7 until the page-write completes. The latter also relies on reading, but from any address, with D6 toggling on each read until the page-write completes.

The chips feature some hardware protection such as a low V_{cc} write inhibit, V_{cc} power-on delay (5 ms), and glitch (15 ns) filtering on WE* and CE*.

In my experience, a software faux pas is actually a more likely source of trouble (consider the last time you lost data on a disk, was it "because the bits wore out" or was it because "Oops, I meant FORMAT B:, not C:"). Thus, the chips also include a software data protection feature (Figure 5) that, when enabled, requires a specific three-byte code to authorize each write. Note that the data protection enable/disable state is preserved across power transitions.

MICRO, HEAL THYSELF

Have you ever noticed that software is never "done," instead it's born, then upgraded, and then it finally dies.

Thus, the concept of committing a "final version" to a zillion ROMs makes me want to join the marketers in the sick bay. While the previously mentioned Flash/EEPROM settles the issue (and my stomach) for multichip designs, what about single-chip micros?

Wouldn't it be nice to have a single-chip micro that could easily be upgraded in circuit, perhaps via PC connection, modem, or IR port?

Sure, you may have noticed a variety of "Flash Micros" announced by many of the leading suppliers. Be wary of these because the presence of Flash, EEPROM, or SRAM doesn't

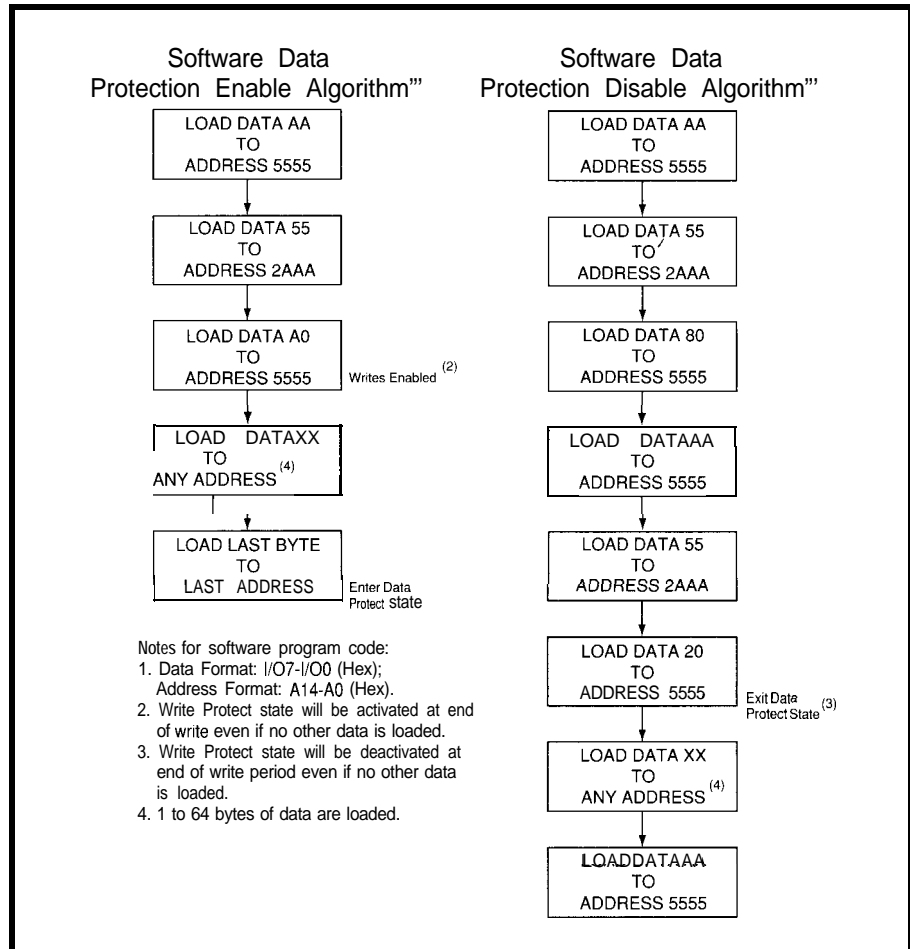


Figure 5—In an effort to prevent accidental (or malicious) data corruption, the Atmel chips provide a software data protection scheme.

necessarily mean a micro is easily in-system programmable.

In some cases a small amount of SRAM or EEPROM is really intended just for data storage. For instance, the Philips 80C851 (\$10.43 in 100s, plastic, commercial temperature) is a familiar CMOS 8051 derivative (4K ROM, 128 bytes RAM) with the addition of 256 bytes of EEPROM. Worse than the small size, the EEPROM is only accessible as data, not as program. Both restrictions mean changing (or fixing) program functionality isn't at all straightforward.

Even if copious program storage is provided, all is not necessarily rosy. For instance, Atmel offers the AT89C51, which replaces the 80C51 4K ROM with 4K of Flash (\$17.00 in 100s, plastic, commercial temperature).

Sounds nice, but closer examination reveals that the Flash programming mechanism is exactly the same as that of an EPROM-based 8751. The

good news is out-of-system programming is easy, because you can just use your existing 8751 programmer. The bad news is that said programming mechanism uses most of the I/O pins, many bidirectionally. Thus, an in-system programmable AT89C51 design would need lots of external control and multiplexing logic to switch pins between programming mode and their application I/O function.

PROGRAMMABLE WIRES

With in-system programmable gates and software in hand, the only "hard" thing about a design is the good old PCB.

Aptix comes to the rescue with the FPCB (Field Programmable Circuit Board) which is based on their FPIC (Field Programmable Interconnect) components.

The FPIC AX1024 is simple to explain, once you get past the shock of dealing with a 1024-pin chip (and

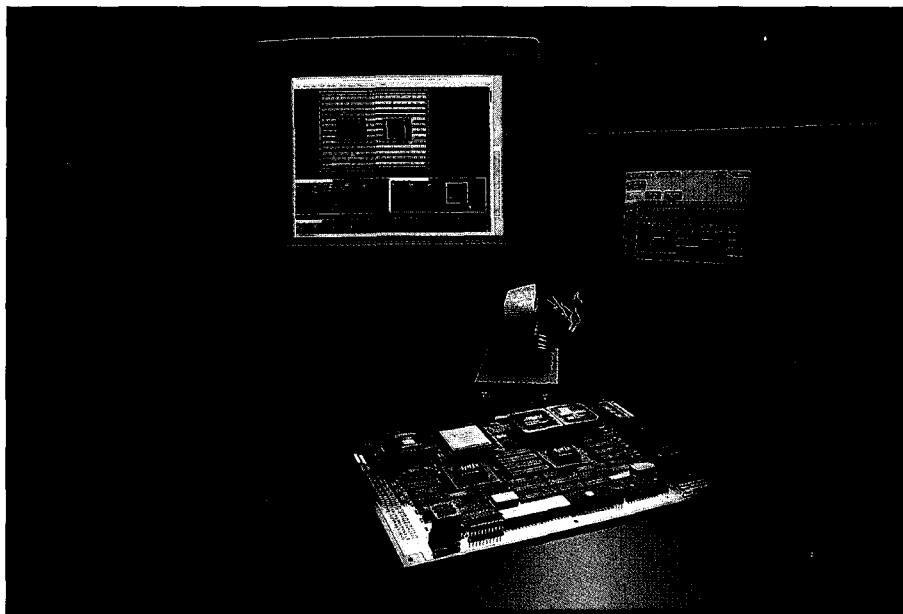


Photo 2—The Aptix AXB-GP2 general-purpose FPCB (Field Programmable Circuit Board) contains over 1700 through-holes laid out in an area of 100×100×300 mils and allows programmable control of the actual PC board layout.

\$1152.00). Indeed, the whole point is that any pin can be routed to any other by setting the SRAM bits at the appropriate intersection.

Two AX1024s are put to work on the AXB-GP2 General Purpose FPCB (\$1450.00 without FPIC). As shown in Photo 2, over 1700 through-holes are laid out in an area of 100×100×300 mils. The holes are split into two regions and each hole is connected to a central FPIC.

Aptix also offers AXBs targeted for PC/AT (ISA) bus, VME bus, and others. Besides allowing quick schematic tweaks, the programmable routing makes debugging with a logic analyzer easy since wires of interest can be soft routed to a hook-it-up-once probe connector.

Most recently, they've introduced the AXB-AP4 (\$6400.00 without any chips) which targets ASIC prototyping with the combination of four FPICs and up to 21 Xilinx FPGAs—kind of a Quickturn-like hardware emulator for the masses.

Gee, in-system programmable gates, wires, and software could make a hardware designer's job easy. Just ship a board full of uncommitted chips now and let the marketing folks figure out what they want it to do later. Better yet, every bug or complaint down the road is automatically a "software problem." The hardware

department has little to do but sit back and figure out who to point their fingers at.

STAY TUNED

When they finally get back from lunch, the marketing folks are going to jump on in-system programmability with a vengeance. To make sure you aren't bamboozled by hype, keep the following tips and traps in mind:

- Look for dedicated programming pins (typically serial), nonintrusive access, and single power supply operation to support easy updating and debugging in place. You'll have to look past the "field programmable" slogan to see if they really mean just "field" (i.e., external programmer) or "in-system" programmability.

- Make sure programming time isn't a problem (typically for serial access and/or Flash and EEPROM). A few seconds programming a high-density chip could prove problematic in certain "nonstop" applications. A related question—just what are the pins doing during programming?

- *Besides basic bit-cell reliability, check for other hardware protections such as power supply monitoring and control signal glitch filtering.

- *Similarly, look for password-like software protection. Remember, in-system programmable is also in-system corruptible.

- For microprocessors, make sure the amount, speed, and accessibility (i.e., code, data, or both) of the Flash/EEPROM/SRAM memory is sufficient. Watch out for chicken-and-egg problems associated with "self-programming."

- Don't forget EEPROM and Flash write endurance limits, which might be as few as 1000 cycles on micros and logic chips. This is fine for time-to-time bug fixes, but wouldn't be suitable for a PC application that programs at every boot. ☺

Tom Cantrell has been an engineer in Silicon Valley for more than ten years working on chip, board, and systems design and marketing. He can be reached at (510) 657-0264 or by fax at (510) 657-5441.

CONTACT

Aptix Corp.
2890 N. First St.
San Jose, CA 94134
(408) 4286200
Fax: (408) 944-0646

Atmel Corp.
2125 O'Nel Dr.
San Jose, CA 95 13 1
(408) 441-0311
Fax: (408) 436-4200

Philips Semiconductors
8 11 E. Arques Ave.
Sunnyvale, CA 94088-3409
(800) 234-7381
Fax: (708) 296-8556

Quickturn Designs
440 Clyde Ave.
Mountain View, CA 94043-2232
(415) 967-3300

Xilinx
2100 Logic Dr.
San Jose, CA 95124
(408) 559-7778
Fax: (408) 559-7114

I R S

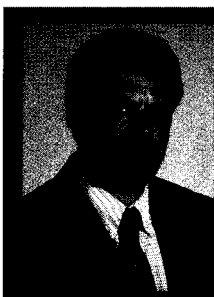
419 Very Useful
420 Moderately Useful
421 Not Useful

Getting to the Embedded Core

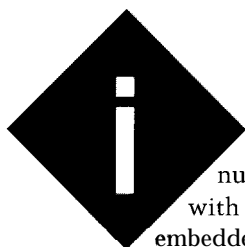
EMBEDDED TECHNIQUES

John Dybowski

The DS2250, Backplane, Power Control, and LCD



Now that the introductory pleasantries are out of the way, it's time to get down to some hardware. John starts the presentation of his embedded control system with the processor module and other core elements.



In spite of all the nuances associated with defining what an embedded computer is really all about, it's probably fair to say that what distinguishes it from its desktop relatives often boils down to the issue of unattended operation. You might find that what screams along on the desktop screeches to a halt once you take it from its intended sphere of application and place it where it must fend for itself. If you've been following Ed Nisley's wonderful '386SX transmogrification, then I'm sure you realize there's more to embedded computing than just putting some computational performance in a box.

The system I am about to describe is designed from the ground up to stand on its own. It brings its own power source, can turn itself on and off, and is capable of accepting various forms of data from a multitude of input devices. And many of the more subtle features needed for reliable unattended operation come built right into the DS2250 processor module that is the brains of the outfit.

The DS2250 provides a good foundation on which to build such an instrument since it was designed with embedded control in mind. Still, you will see that there's a lot more to it than just picking a good processor. I'll be covering these important issues in

some depth at a later time. In the meantime, let me start things off with a detailed explanation of some of the more critical constituents of this embedded computer, what they are, and why they are there. To ease my discussion I'll refer to this system simply as the ec.25.

CARRYING THE LOAD

First of all, as can be seen in Figure 1, the system carrier is the platform on which the ec.25 is constructed. It contains eight identically configured 38-position I/O sites. It is through these sites that all of the standard peripheral functions are interfaced to the DS2250, the power bus, and to one another as appropriate. Common connections are carried from the DS2250's P1, P2, and P3 ports as well as to the RST and \PSEN lines. In addition to these, a special site is provided that connects to PO along with P3 and the common control lines.

The intention of this auxiliary location is to allow access to the system interrupt structure and other special control signals along with the inherently fast digital I/O capability of the directly driven bidirectional PO port bits. Since these pins are fully described in the DS2250 data sheet, it could be said that this presents a truly open architecture. This is partially true and I'll fill in the missing pieces as I provide you with the firmware drivers necessary to fully utilize all of the system peripherals.

Incidentally, when you refer to the peripheral card schematics note that the convention I use to describe the bus pins includes the signal name, the bus connection, the connector designation, and connector pin. The nomenclature GND/38J3-8 tells you that the ground connection is assigned to bus position 38 and this particular card uses pin 8 of J3 for this function.

POWER I/O

The power I/O card contains several different function blocks. These include a line-powered RS-232 port, a CMOS RS-485 port, an I²C port, and a 10-V preregulator and external power jack. Let me describe these individually.

The ec.25 is capable of running off of battery power or an unregulated 12-VDC power supply. The external DC supply provides operational power to the system and the power for the battery management subsection. When using any "standard" wall-mounted 12-volt power pack, you can usually expect the output voltage to range anywhere from 12 V all the way up to 16 V (and beyond). This level is not only dependent on the supply loading but also on the line voltage. The micropower switching regulators used in the ec.25 don't do well with such voltage extremes. In fact, if you hit the main system regulator, a MAX639, with anything over 12 V, it will quietly and irreversibly self-destruct. To prevent this unfortunate occurrence, the low-voltage DC is preregulated to a level of about 10 V using VR1 (an LM317) as shown in Figure 2. The regulator's output is diode isolated since the main power feed also takes juice from the system battery.

Looking further at this schematic you will see that I constructed a line-powered RS-232 interface entirely from discrete components. With all of the single-supply RS-232 transceivers on the market, you may have come to

the conclusion that I'm just plain cheap. Lest you go away with this opinion, let me explain my motives for what I have done.

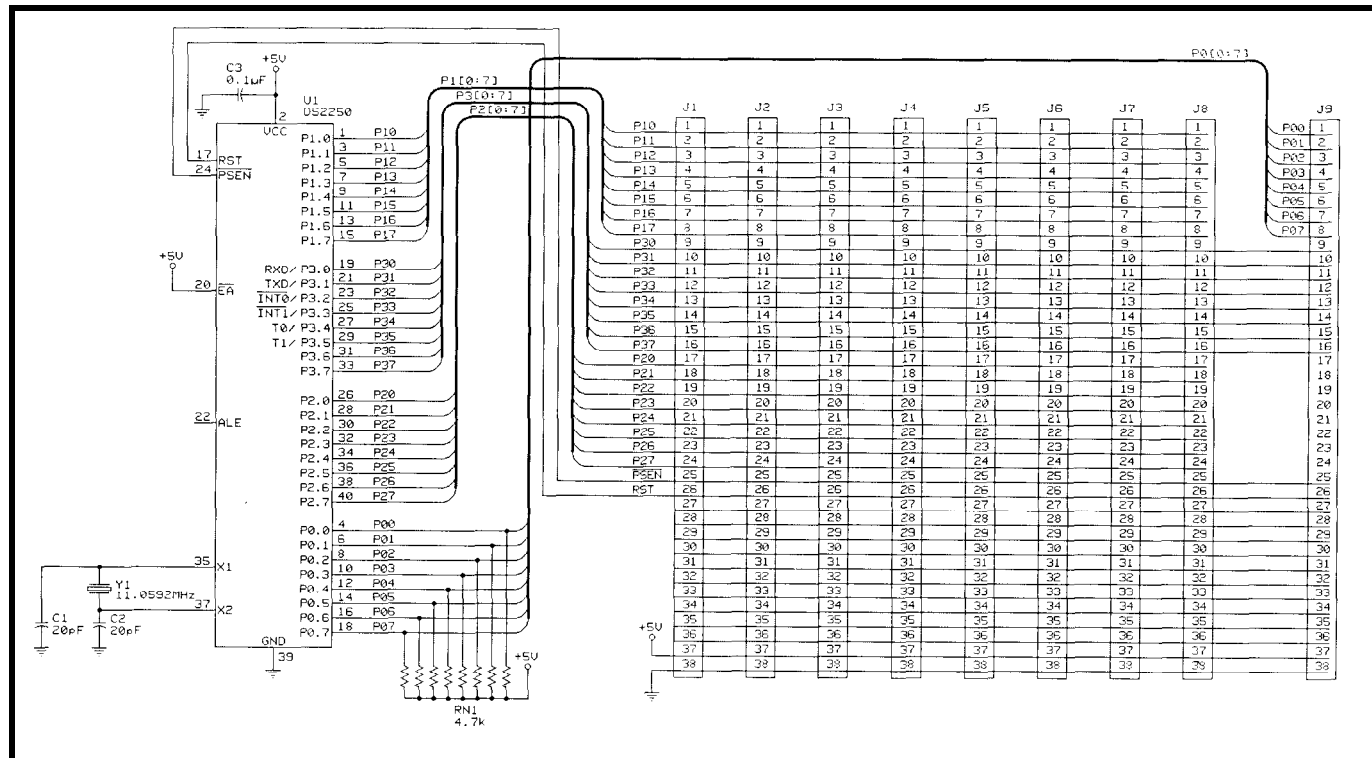
Although many of the integrated RS-232 transceivers are quite miserly as far as power consumption is concerned, they do draw current, of course. This may or may not be a concern, but in any case it's highly desirable to keep the power consumption as low as possible in any battery-operated equipment. I am familiar with Dallas's DS1275 line-powered transceiver chip, but this part is intended only for half-duplex operation. For full duplex, you need two. I can't help but get the feeling that maybe they slipped on this one when they went to silicon, especially since the part has a pin that's designated as a no-connect. Seems to me that full-duplex operation would certainly have been a requisite function.

Anyway, I am always concerned that a powered-down system can become upset by unintentionally introducing a voltage onto the V_{cc} bus. This can happen when you have a live RS-232 connection to an unpowered RS-232 transceiver chip. This backfeeding might cause problems in

any system that contains battery-backed memory and could be especially troublesome in a system based on the DS2250 since the program itself is contained in RAM. Because of this, I wanted to be sure to avoid any such entanglements. Looking again to the circuit diagram, you can see that the RS-232 receiver consists of nothing more than Q1 and a couple of passive components. D1 on the base limits the negative voltage excursions to a level that the transistor can safely tolerate.

The transmitter section steals negative voltage from its receive line by rectifying and filtering this signal via D2 and C 1. This stored energy is presented to the transmit pin via R5 while the line is in its mark state. When a space condition is asserted, Q2 switches this line to V_{cc} through R4.

The line designated DTR performs a special function when using the PC-resident IPL utility. By turning on DTR, the host PC forces the DS2250 into bootstrap mode which allows, among other things, the downloading of executable programs. This mode is entered by simultaneously driving the DS2250's RST high and \overline{PSEN} low. This is achieved through the use of Q3 and Q4 and their associated passive



support components. D6 is the bootstrap mode indicator LED and is illuminated when this mode is in effect. Jumper S1 is provided to manually force the system into bootstrap mode without the need for asserting DTR at the PC. For those cases where inadvertently bootstrapping the system is undesirable, jumper S2 can be removed to inhibit this function.

The RS-485 port is implemented in the conventional manner using U1 the CMOS LTC485 RS-485 transceiver chip. This part conforms to the industry standard established by parts such as the 75176 but draws only a fraction of the current consumed by its bipolar counterpart. Note that the RS-485 transmit enable line is inverted by Q5. Since the RS-485 enable line is driven via a DS2250 port pin, this ensures that the default power-up state leaves the transmitter off. Not only does this prevent the communication line from being glitched as the DS2250 is held in reset, but more importantly

safeguards the network in the unlikely event that the DS2250 does not come up executing its program properly. This also leaves the RS-485 line clear if the DS2250 is being downloaded via the RS-232 port, because during the download the DS2250 is effectively held in reset.

There's one other thing I should make you aware of as far as the LTC485 is concerned, and this is true of competing CMOS transceivers such as the 75LBC176. Even though these parts are billed as fully compliant with the RS-485 specification, I view them as merely RS-485 compatible. The problem is that they are, contrary to what the manufacturers say, very susceptible to electrostatic discharge. My experience has been that discharges in the 3–5-kV range anywhere near the cabling will throw these parts into latchup. Generally the discharge event is not itself destructive, but if the latchup is not released by momentarily interrupting V_{cc} , the part will cook itself to death.

If all you plan to do is connect to an RS-485 network occasionally or if you've got relatively short and well-routed network cabling, then you'll probably be fine with a permanent hookup. In any case, having RS-485 capability on such a computer is quite handy. What clinches the deal in favor of the more fragile CMOS is that a bipolar transceiver would draw more current than the entire system!

Finally, this card also contains an I²C port which consists of a four-position telephone-style connector and some pull-up resistors.

POWER MANAGER

This card, depicted in Figure 3, is responsible for delivering power to the system logic circuitry. It also provides a backup power source for nonvolatile peripherals such as RTCs and RAMs. The main regulator (VR2) is a MAX-639. This device is a high-efficiency step-down switching regulator capable of supplying 5 V at 150 mA. Switch mode regulators which operate at

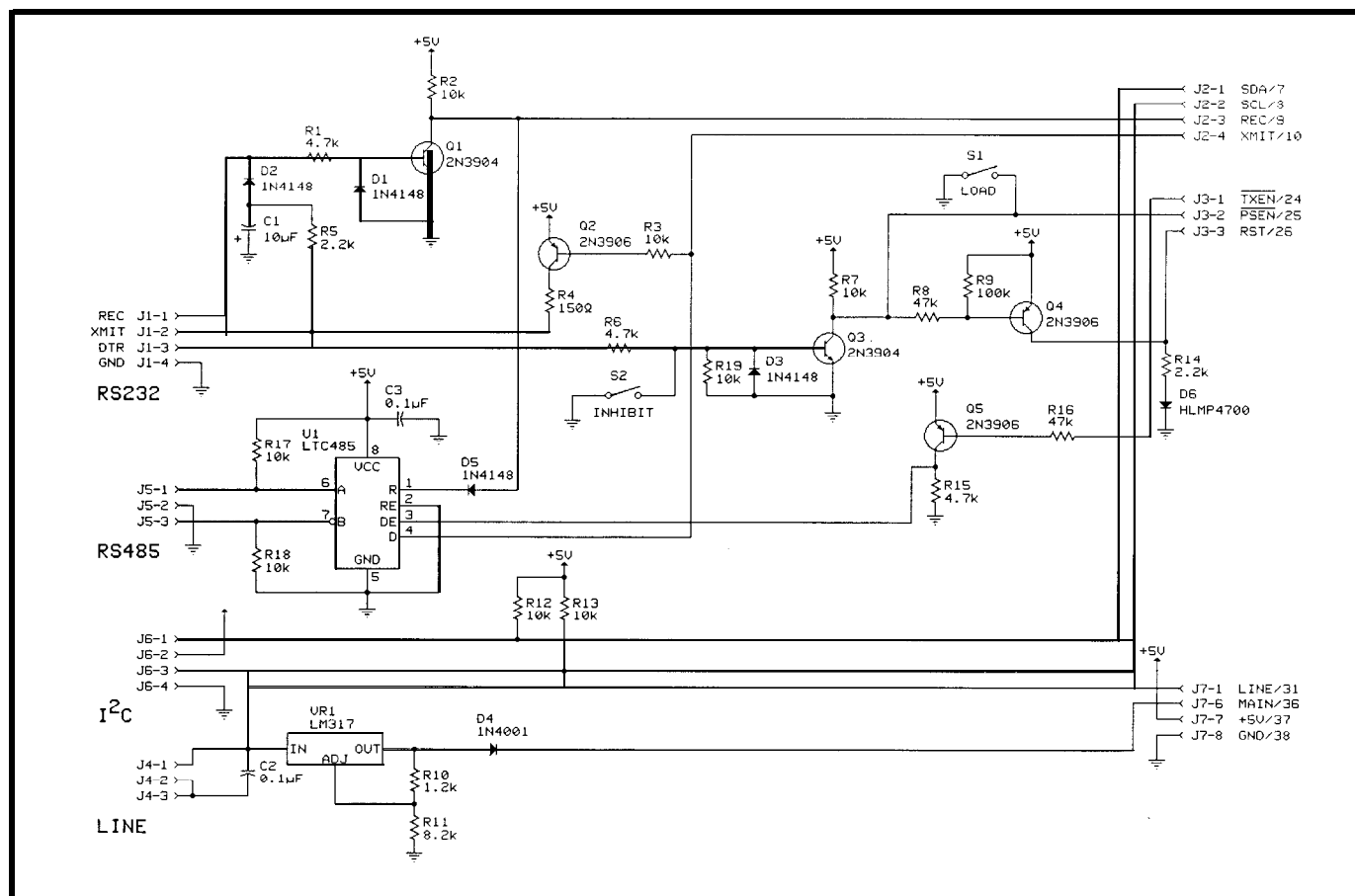


Figure 2—The RS-232 interface is done completely with discrete components and steals power from the device it's connected to. The RS-485 interface uses a very low-power CMOS transceiver chip (LTC485). Finally, to avoid destroying the MAX639 with too much voltage (over 12 V), supply voltage is preregulated to 10 V using an LM317.

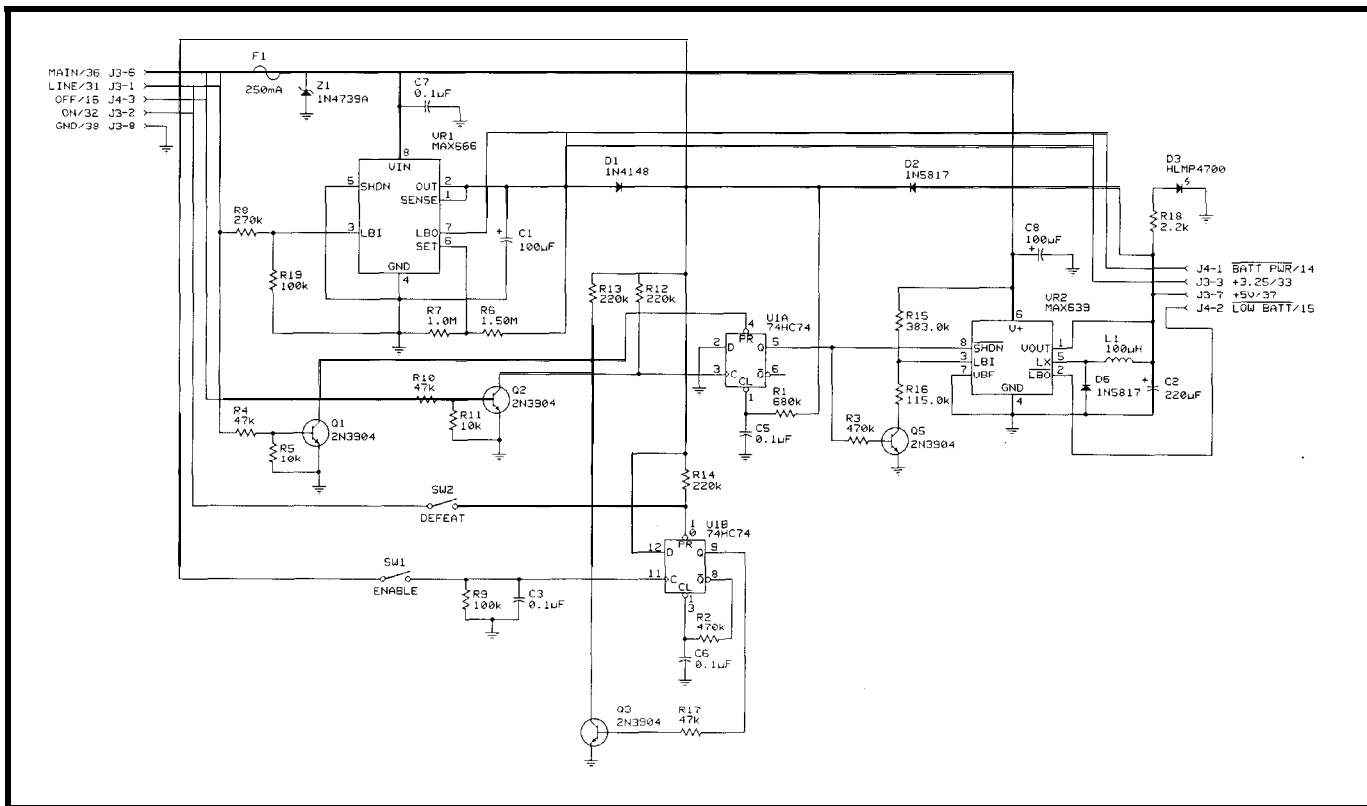


Figure 3—The power manager card delivers power for system logic circuitry and provides backup power for nonvolatile peripherals.

constant power are much more efficient than their constant-current pass-mode counterparts.

Furthermore, the MAX639 uses a gated oscillator, or pulse skipping, technique that offers extremely low quiescent current and makes it especially useful in battery-powered circuits. In addition, the MAX639 varies its on time as a function of input voltage thereby keeping the peak inductor current constant over the entire input voltage range. This minimizes the inductor and filter capacitor requirements and prevents loss of output power over the device's operating range.

Of course there's no such thing as a free lunch, and these advantages must be balanced against increased levels of low-frequency subharmonic noise (not to mention a somewhat restrictive upper input voltage limit) that comes with using such power-saving techniques. In any event, by using good layout practices, proper decoupling, and lots of copper, reasonably good results can be attained.

The MAX639 also features an active-low shutdown pin. When this pin is pulled below 2 V, the oscillator

The Real Logic Analyzer



test your Logic circuits with the printer port of your IBM or compatible computer!

- El 5 Input capture channels via printer port
- ☒ High Speed 64K input capture buffer
- ☒ Glitch capture and display
- ☐ Full triggering on any input pattern
- ☐ Automatic time base calibration
- ☒ 4 cursors measure time and frequency
- ☒ Save, print or export waveforms (PCX)

The Real Logic Analyzer is a software package that converts an IBM or compatible computer into a fully functional logic analyzer. Up to 5 waveforms can be monitored through the standard PC parallel printer port. The user connects a circuit to the port by making a simple cable or by using our optional cable with universal test clips. The software can capture 64K samples of data at speeds of up to 1.2uS (Depending on computer). The waveforms are displayed graphically and can be viewed at several zoom levels. The triggering may be set to any combination of high, low or Don't Care values and allows for adjustable pre and post trigger viewing. An automatic calibration routine assures accurate time and frequency measurements using 4 independent cursors. A continuous display mode along with our high speed graphics drivers, provide for an "Oscilloscope-type" of real time display. An optional Buffer which plugs directly to the printer port is available for monitoring high voltage signals.

LOGIXELL
ELECTRONICS

61 Piper Cr.
Kanata, Ontario
Canada **K2K 2S9**

Requires 286, or higher with EGA or VGA display.

LA20 Software Only \$79.95us
Software With Test Cable \$99.95us
BUFF05 Buffer \$39.95us

Tel: (613)599-7088

Fax: (613)599-7089



and LX power switch are shut off, bringing the power consumption down to about 10 μ A. As shown in the schematic, this pin is driven by a low-level signal that is sourced by the power control flip-flop (U1a). The low-battery input (LBI) monitors the main power source using the divider made up of R15 and R16. The \LBO output is used as a low-voltage warning indicator. Transistor Q5 completes the LBI divider's path to ground only when the MAX639 and the system is operational. There are a couple of implications associated with this arrangement that are worth noting.

First, realize that the \LBO signal is intended to indicate an impending regulator failure condition. Depending on the type of battery the system is equipped with, this level, which is set at 5.5 V, may be too low to safeguard the battery. In other words the battery could possibly have already dipped into the danger zone of deep discharge when this indicator trips. The purpose of this circuit is not to indicate a battery problem but rather a regulator problem and as such fulfills its objective. The idea is to maintain flexibility and not to tie this card to any particular type of battery. I'll show in a subsequent column how one of the ec.25's ADC channels is configured to monitor the battery voltage specifically for purposes of battery discharge monitoring. Secondly, the switching transistor (Q5) which completes the LBI divider circuit may seem superfluous. This may be true if the system is equipped with a hefty battery, but this may not necessarily be the situation.

Consider a specially configured system that is intended to operate for an extremely long time

(one that remains in standby mode for extended periods). In this scenario, the battery of choice would likely not be a rechargeable type at all; an alkaline or lithium chemistry might be a better fit. Here the power savings resulting from disconnecting the battery monitoring divider could be significant. Again, general-purpose flexibility prevails in the design choices.

The backup power regulator is a MAX666 (VR1). This low-power pass stage is set up to deliver an output of 3.5 V. This voltage powers the system's nonvolatile circuits such as the RTC, external RAMs, and the power-control circuitry itself that is contained on this card. Although I could have gotten by using a lower voltage for this purpose, some system peripherals (such as the serial 512K SRAM card that is under development) will have an integrated backup power

source for maintaining critical data in the event of a total loss of main battery power. The failsafe backup source in this case will naturally be a small lithium cell. Using the relatively high primary backup voltage of 3.5 V ensures that no current will be drawn from the failsafe backup cell during periods of normal standby operation. R8 and R19 form a divider that feeds VR1's LBI pin. This noncritical divider that is fed from the line input causes \LBO to pull low in the absence of line power indicating battery operation.

The ability of the ec.25 to control its power is an important element. This function is centered around the power-control flip-flop U1a. This chip gets its power from the 3.5-volt backup regulator during idle periods and from the main regulator when the system is active. This mixing is accomplished by diodes D1 and D2.

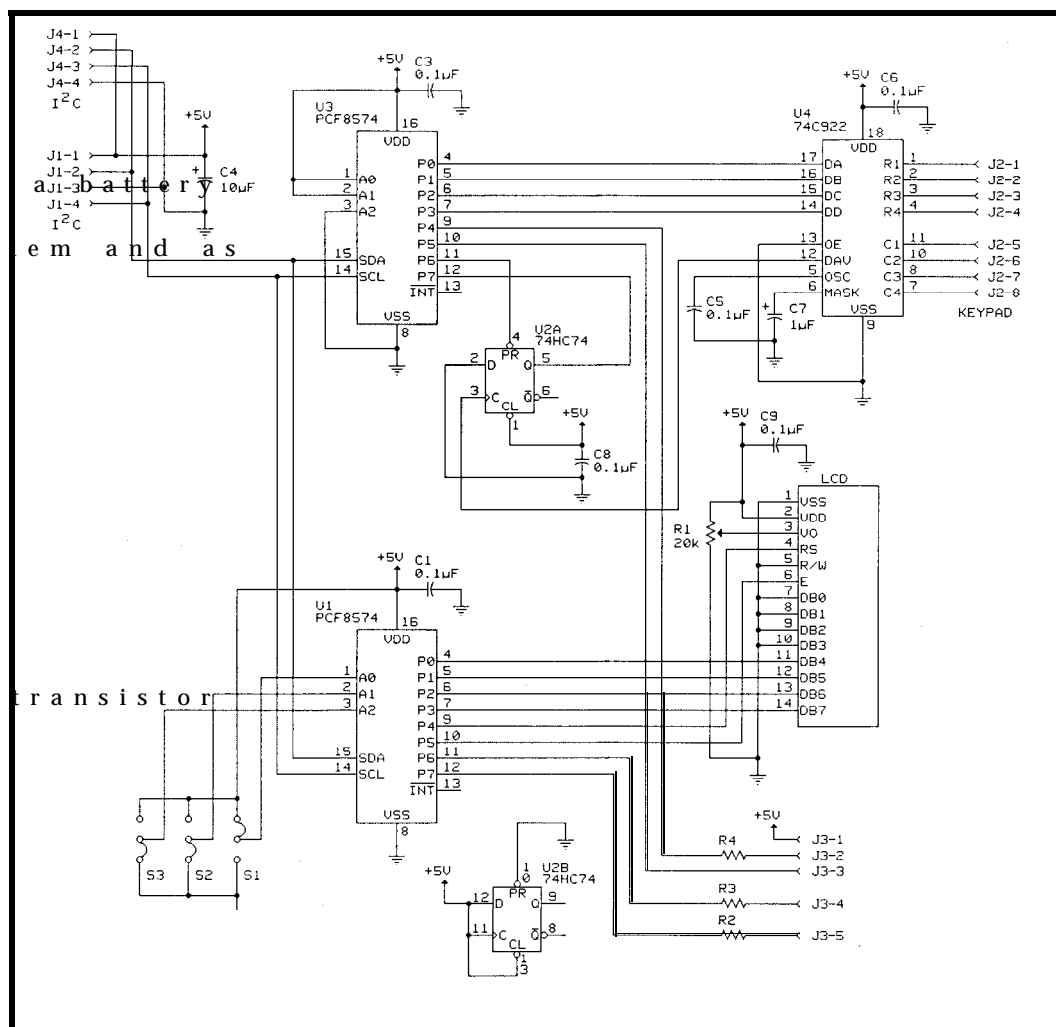


Figure 4—The user interface consists of a keypad and an LCD panel. By using a serial PC bus interface, the board can be connected to the system using just two signal wires, V_{cc} , and ground.

On initial application of battery power, the RC master reset circuit composed of R1 and C5 fires ensuring that the main control flip-flop (U1a) is in the off state. Following this, the system can be manually energized by pressing the push-button switch (SW1) which causes the one-shot built around U1b to emit a pulse that is inverted by Q3 and is presented to the \PR input of U1a. This causes U1a's Q output to drive the \SHDN input of VR2 high, which effectively supplies power to the system. This powered state is shown by D3, which indicates that power is being supplied to the system. Note the time constant of the one-shot RC network made up of R2 and C6 is shorter than that of the master reset circuit. Because of this, if a false enable signal is generated on initial application of battery power, the master reset circuit will mask the event and guarantee that the system doesn't inadvertently power on.

Line power is buffered and inverted by Q1 and drives the \PR input of U1a low, resulting in the Q

output going high and enabling VR2. Application of line power unconditionally enables full power to the system. The \ON signal is supplied by external circuitry (typically an RTC/interval timer contained on another card) and presents a low level to the \PR input of U1b. Since driving \PR low causes \Q and \CL to go low, U1b effectively operates in a pass-through mode. If you check the truth table for the 74HC74, you will see that when both \PR and \CL are asserted simultaneously, the Q and \Q outputs both drive high. In other words, the system master enable remains in effect until the \ON signal is released at its source. Normally \ON is released by resetting the peripheral that is generating the signal. The recognition of the \ON signal can be defeated by removing jumper SW2.

Powering down the system is accomplished via the \OFF line that is inverted by Q2 and is routed to U1a's CLK input. This causes the transfer of the state of the D input (which is grounded) to the Q output and asserts VR2's \SHDN pin. The source of the

\OFF signal is a port pin of the DS2250. Since the enable one-shot (U1b) drives the power control flip-flop's (U1a) synchronous \PR input (which takes precedence over the asynchronous CLK input), the \OFF signal is effectively masked as the system powers up and the DS2250 emerges from reset. The duration of the enable one-shot is set up to make sure any spurious transitions on the \OFF line are ignored during this period.

USER I/O

The ec.25's user interface is a familiar configuration consisting of a 20x4 character-based LCD panel and a 4x4 matrix membrane keypad. Although this is fairly standard stuff, you may find the implementation details interesting. Access to this interface module is via the serial I²C bus and as such is carried out using just two signal wires plus V_{cc} and ground. This module can be mounted remotely to the ec.25 using the modular jack on the power-I/O card or can be contained locally and connected directly to the

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, 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

WORKS

The
only
8051/52
BASIC
compiler
that is
100 %
BASIC 52
Compatible

and
has full
floating
point,
integer,
byte & bit
variables.

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

508-369-9556
FAX 508-369-9549



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



I²C pins of one of the unused peripheral sites.

Figure 4 shows the circuitry contained on this card. The I²C interface is accomplished using two PCF8574 I²C to byte-wide converters providing digital I/O over eight bidirectional I/O pins. As you can see from the schematic, one of these chips is used for the LCD interface and the other handles the keypad I/O. The LCD is operated in nybble mode, where the four lower-order bits of U1 are tied to the LCD's data bus. P4 and P5 are assigned as control pins and drive the register select (RS) line and the enable (E) line. The LCD operates in write-only mode and therefore the read/write line (R/W) is grounded. P6 and P7 are brought out to a header and can be used for driving LEDs or for other general I/O purposes. The LCD's VO pin connects to the wiper of R1 that changes the panel's viewing angle to suit a variety of lighting and viewing conditions.

The keypad interface is almost as simple. Using U3, the second PCF8574, as a bidirectional I/O port, PO through P3 accept data from the 74C922, U4, which is interface chip for the 4x4 keypad interface. The 74C922 independently performs the matrix scanning using an internal oscillator whose scan frequency is set by C5. The debounce interval is selected by C7. When a valid key closure is detected, following the debounce interval, U4 asserts data available (DAV) to indicate that a valid key code is available. This signal is latched and inverted by U2a and its output is made available to P7 of U3. This latch is necessary since if the key is released, DAV will return low. Although DAV is lost in this situation, the key code will continue to be emitted on U4's data bus. The host processor continually polls the status of U3 and recognizes that a key is available if P7 is low. If this is the case, the processor must clear DAV by setting P6 of U3 low. Finally, the sequence is completed by writing an FFh back to U3 to return it to its idle state in preparation for the next key hit. P4 is a general-purpose I/O bit and P5 drives a self-oscillating beeper.

Listing 1--Most of the code needed for driving the I²C-based remote keyboard and other I²C devices consists of parameter mapping between C and the low-level routines written in assembler.

```
#pragma OBJECTTEXTEND LARGE SYMBOLS CODE
#include <reg5000.h>

extern void KEY-OUT (char c);
extern char KEY-IN (void);

/* Wait for a key to be struck */
char _getkey (void)

extern char CheckKey (void);
unsigned char c;

    while (! (c = CheckKey()));
    return (c);

/* Check if a keystroke is available */
char CheckKey (void)
{
    unsigned char c;
    static code char KeyTable [16]={"\001\002\003\004369\n2580147\377"};

    c = KEY-IN();
    if (c & 0x80)
        return (0);
    KEY-OUT (0x9f);
    KEY-OUT (0xff);
    return (c=KeyTable[c & 0xf]);
}

/* Initialize (or flush) the keypad */
void InitKey (void)

    KEY-OUT (0xbf);
    KEY-OUT (0xff);
    return:
}

/* Assembler linkage: Output a byte over I2C bus */
static void KEY-OUT (char c)

extern void Xmit_I2C_Byte (void);

    ACC = 0x46;
    B = c;
    Xmit_I2C_Byte ();
    return:
}

/* Assembler linkage: Input a byte over I2C bus */
static char KEY-IN (void)

unsigned char c;
extern void Rec_I2C_Byte (void);

    ACC = 0x46;
    Rec_I2C_Byte ();
    c = B;
    return (c);
}
```

Since the PCF8574 transceivers are addressable I²C peripherals, you can easily connect multiple display

panels to the second I²C connector by just changing the chip's base address via jumpers S1, S2, and S3. This is not

only handy for those special applications that require multiple display devices, but having extra output capability is also useful for debugging.

METTLE AND METAL

programming embedded computers has come a long way. It used to be that the true mettle of an embedded programmer was synonymous with "bare metal" machine-level programming. Things change, partly due to changing social and economic conditions and, more importantly, due to changes in compiler technology. Having examined the state of the art in some detail, I've concluded, like many of those that come before me, that programming microcontrollers in C is not such a bad idea after all. A number of vendors are now producing truly fine code generators for some furiously complicated controllers.

The turning point in microcontroller compiler design came when some bright individuals got the message that certain controllers didn't possess the architecture necessary for

the canonical and "standard" stack-based C implementation. What resulted was compilers that generated code in the controller's native tongue which finally delivered on the promise of adequate performance and manageable code size. Although programming in a high-level language such as C provides a multitude of useful and necessary support functions, as well as relieving the programmer of the tedium of keeping track of microscopic details, all indications point to a thorough understanding of your controller's architecture. In addition, the code your compiler emits is still very important in crafting programs that yield the required levels of efficiency and performance. I've seen some excellent C programs written by excellent C programmers that thrashed about once they were imparted to their ultimate embedded target.

HOW LOW CAN YOU GO?

Since I'm almost out of space, let me leave you with the littlest piece of code I can find to illustrate the type of

programming I will be covering in subsequent columns. The question that often comes up when programming in high-level languages is: how low can you go? Theoretically, I suppose you can go as low as you want. Practically speaking, though, there comes a point of diminishing returns. And if you already have some low-level drivers coded up in assembler, the question may become moot.

Listing 1 provides all the support services for the I²C keypad that I described. Additionally, this program contains the get key function that allows all of the standard C console functions to run using this particular keypad. The get key function waits indefinitely until a key is pressed before returning. Although this implementation is necessary for a number of standard C functions, most embedded programs will not tolerate suspending all operations when the keypad needs to be sampled. To circumvent this problem, and to support get key, C h e c k Key operates at a slightly lower level and simply

!NEW! FROM CIRCUIT CELLAR PROJECT FILE, VOL. II

\$17.95

SAVE 28% OFF

COVER PRICE OF \$24.95

All NEW Projects to SPARK Your Imagination!

GPZS Audio Sampling System
Wiring Your House for the 21st Century
Multiprocessor Architecture using DSP
ANDMUCHMORE!!!

VISA, MasterCard, or International Postal Money
Order (U.S. funds drawn on U.S. bank only)

Circuit Cellar Project File

4 Park Street
Vernon, CT 06066

Tel: (203) 875-2199
Fax: (203) 872-2204

*includes domestic delivery. Please add \$6 per copy for delivery to Canada & Mexico, ... add \$8 per copy for delivery to other non-U.S. addresses.

► Project Parts ◀

New Stuff

PCF8574 I²C Remote 8-bit I/O port 9.00
DS1620 Remote digital thermometer/thermostat 10.20
SAA1024 I²C 4-digit LED driver 12.30
BQ2010 NiCd/NiMH battery gas gauge 19.25
AD75019 Analog 16x16 MUX w/PLCO socket 41.25
MAX4564/57 Analog 8x8 MUX and video drivers 75.80

Firmware Flyers (postpaid in US / with order)

8051 Firmware Debugging Techniques (65 pg w/3.5" disk) \$19 / 15.00
Introduction to the 8051 Instruction Set (46 pages) \$10 / 8.00
386SX Project: FDB schematics (17 pages or so) \$9 / 5.00
8051 Instruction Reference Card \$3 / 2.00
8255 Cheat Sheet Reference Card \$3 / 2.00

Stepper Motor & Power Drivers

ULN3751Z Power op amp (±3V to ±5V supply, 3.5 A output) 4.60
UCN2939B Dual full H-bridge bipolar stepper motor driver 5.45
UCN5941A Serial-input, 8 latched 500 mA sink drivers 6.90
UCN5956A Serial-input, 8 latched 250 mA source drivers 7.50
UCN5949B Unipolar stepper motor translator/driver 8.00
UDN2956W Dual full-bridge stepper driver: 50 V, 2 A 13.80

IR Remote Control

LD273 dual IR LED (bright, wide beam) 2.10
IR3002A laser control (±5V, LT022/MC/LN9705P) 2.30
IR3007 laser control (±5V, LT022PD/LN9705) 2.85
PH302 fast IR photodiode 3.10
GP1US2Y 40 kHz IR receiver (side-looking) 4.00

IS1U60 38 kHz IR receiver 4.80
IRSAMPLE parts (PH302, LM311, etc. MCIR-Link, INK 23) 5.70
Excellent IR filter (opaque to visible light, 35 mm slide mount) 6.75
MC145030 IR encoder/decoder 6.75
IR Link I/O: IS1U60, LD273, CD4047, schematic (IR Link) 10.40

Other Neat Stuff

UGN3503U Ratiometric linear Hall Effect sensor 2.25
(39 page Hall Effect app notes with 3503 order only) 3.90
DS1232 Micromonitor watchdog 6.75
CS212 S-ART I/O network security monitor 8.10
DS2409 Silicon Serial Number (2 chips) 8.40
TDA9444 I²C bus octal 6-bit DAC 8.75
DS1210 NVRAM power controller 9.75
DS1231 Power monitor 9.80
DS1202 Serial clock/calendar w/crystal 10.20
PCF8583 I²C bus clock/calendar w/crystal 10.70
MT8888 DTMF Transceiver (Intel bus interface) 10.50
MAX691 Power supervisor 11.85
75T204 DTMF decoder 12.00
PCF8591 I²C bus quad 8-bit ADC, single 8-bit DAC 14.10
IL300 Linear optoisolator 12.70
MAX233 self-contained ±5V powered RS-232 interface 14.25
BQ2003 Fast-charge NiCd/NiMH battery controller 14.55
MT8808 8x8 bipolar signal analog crosspoint 16.10
HBCS-1100 High-resolution HP bar code sensor 32.80
Firmware Dev Board ICS, schematic thru #38 45.70
MAX252 opto-isolated ±5V RS-232 interface 64.00
CH1840 FCC pre-approved full-function DAA 72.50

...and much more...

JPS Ground/2nd day \$6.5/9 to 48 US states, COD add \$4.50. PO Boxes and Canadian orders \$6 for USPS mail. Check, MO, or COD only, no credit cards. POs add \$50, call first. NC residents add 6% sales tax.

Quantity discounts start at five parts. Data sheets included with all parts.

Call/write/fax for seriously tempting catalog...

Pure Unobtainium

► Your unusual parts source ◀

13109 Old Creedmoor Road • Raleigh, NC 27613
FAX/voice (919) 676-4525

checks if a valid key is available. This

I²C keypad port. If DAV is high, then the function terminates and returns zero to indicate that no keypad data is available. Otherwise, the DAV latch is cleared and the beeper is energized by outputting a 9Fh. Immediately following this, the port is returned to its default state of FFh. This action results in a keyclick sound being emitted and the keypad interface being rearmed for the next keystroke. The function then translates the low nybble just read from the keypad port to an ASCII code and returns this value to the caller.

I n i t Key simply resets the DAV latch and returns the keypad port to its default state. This function is invoked as part of the power-on initialization procedure and can be called anytime to flush any pending keystrokes.

The linkage to the assembler level is accomplished using the functions

KEY-OUT and **KEY_IN**. These functions do nothing more than move arguments between the native formats of the C code and the low-level assembler. This is what is generally referred to as the *stub level*. Often this argument shuffling is, of necessity, performed in assembler, but here the C compiler can work directly with the 8031 registers. Of course, it would have been possible to simply write the assembler routines to conform with the compiler's parameter-passing conventions, but there were a couple of reasons why I elected not to do so. As compilers evolve, these parameter-passing conventions evolve also. Naturally such drastic changes force you to make modifications to your assembler code as well, which maybe is something better avoided. My main reason for using such stub code, however, is founded on more practical reasons.

The I²C drivers I'm using have long since been completed and debugged. As a matter of fact, I presented these very drivers in their entirety in issue #34. There are a

couple of forces at work here: The code works and I can't waste space here presenting the same stuff with some minor tweaks. In any case, since the ec.25 has about half a dozen I²C peripherals, I will be returning to these low-level drivers frequently as I cover other aspects of this project. Talk about hot spots!

Next month I'll knock off a couple more major subsystems and I'll start getting more heavily involved in some programming aspects. I'm with you: I want to get down to some practical applications. □

John Dybowski is an engineer involved in the design and manufacture of hardware and software for industrial data collection and communications equipment. He may be reached at john.dybowski@circellar.com.

I R S

422 Very Useful

423 Moderately Useful

424 Not Useful

PRO = Positively Rampant Optimization

4 x 5 keypad input
4 × 20 or 8 × 40 LCD display output

Hardware real-time clock
24-bit parallel TTL I/O

CMOS PROCESSORS

• 80C31/Processor or
• 80C52 with BASIC52

8/32K RAM

PRO31 prices start as low as \$289 (**\$219/100**), includes **8K** RAM, watchdog, 56 bits I/O, S-bit ADC, and LCD interface. **And it only costs \$10 more for PRO52!**

8/32K ROM

EXPANSION HEADER

• Port 1 processor lines: 8-bit TTL
• Six decoded I/O address strobes
• Eight address lines

Power inputs
– Single supply or multisupply option
Console serial port

Auxiliary serial port
24-bit parallel TTL I/O
Hardware watchdog

8-bit, 4-channel DAC
8- or 10-bit, 0-channel ADC



PRO31/52 MICROMINT, INC.

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

in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (02) 888-6401 • Distributor Inquiries Welcome!

PATENT TALK

by Russ Reiss



any tools and techniques are available these days to ease the microprocessor system implementor's task of getting a new system running. This month's patent review covers both historical and latest developments in this area, with topics that range from ROM Emulators and ICE devices to Tracing Maps and Software Debugging.

The lead patent should be of interest to many as it relates to the popular In Circuit Emulators (ICEs) from Metalink. Their novel techniques handle the task of emulating an 8051 single-chip microprocessor with no

external buses using an 8031. Emulation of the 8051 is complicated by the presence of its internal ROM. Somehow the emulator must use external memory which is not resident in the target system when these memory addresses are referenced. Metalink substitutes an 8031, which has no internal ROM, for the target 8051. By carefully using Port 0's multiplexed data and low-order address and Port 2's high-order address lines, emulator RAM can instead be accessed whenever on-chip ROM would normally have been read. They have found that the external access (EA) lead must also be controlled in order to avoid clobbering Port 0 as it would if left to run as a normal 8031 at all times.

As we've seen so often in the past, each field seems to have its "generic" patents: those patents at which we shake our heads and wonder if everything under the sun is not covered by a patent! The area of microprocessor develop-

Patent Number 4,809,167
Issue Date 1989 02 28

Inventor(s) Pawloski, Martin B.; Borkar, Shekhar Y.
State/Country AZ
Assignee Metalink Corporation

US References 4,441,154 4,527,234 4,677,586

US Class 3641200

Title Circuitry for emulating single-chip microcomputer without access to internal buses

Abstract An emulator circuit utilizes an Intel 8031 microprocessor with external address and data buses to emulate an Intel 8051 single-chip microcomputer with no external buses by providing external registers into which the contents of the internal 8031 "Port 0" and "Port 2" registers are output and functionally "recreated." The external access (EA) lead is toggled to make the 8031 function as an 8051 during the states in which the 8051 samples its logic levels and destroys Port 0 latches if configured as an 8031. Toggling the EA lead to a high level causes outputting the contents of the Port 0 and Port 2 latches to their respective leads. The emulator circuit generates a "Force Ports" pulse that causes the "recreated" port registers or the external circuitry to "force" external logic levels onto the 8031 Port 0 and Port 2 leads. The address latch enable (ALE) signal is delayed until after the Force Ports signal lapses to allow the internal Port 0 logic to generate address outputs on its leads as part of the external address bus.

1

Patent Number 4,691,316
Issue Date 1987 09 01

Inventor(s) Phillips, Charles R.
State/Country OR
Assignee Support Technologies, Inc.

US References 4,315,321 4,317,199 4,433,412 4,450,519 4,455,654 4,484,329 4,489,414

US Class 371120 371116

Title ROM emulator for diagnostic tester

Abstract A diagnostic test system for microprocessor-based electronics systems comprises a computer and a ROM emulator for emulating the read-only memory in the device under test which provides the microprocessor with its start-up operating program. The ROM emulator gains control of the microprocessor in the device under test and causes it to run a series of diagnostic tests which are controlled from the test unit. Unique capture logic circuitry in the ROM emulator permits test data to be read over the system address bus.

2

PATENT TALK

3

Patent Number 4,985,893
Issue Date 1991 01 15

Inventor(s) Gierke, Daniel
State/Country IL

Title Circuit testing apparatus

Abstract Generally there is provided a ROM memory arranged to be inserted in place of a preexisting ROM on the microprocessor-based circuit board under test, wherein the ROM contains specific test sequences programmed therein. The test apparatus further includes a microprocessor control, a keypad, a display, and an interface. On start up, the circuit under test runs programs stored in the inserted ROM and outputs on the data bus information relative to the results. During some tests the circuit is held in a wait state to allow probing of the circuit under test for logic errors. In an operator-programmed test mode, the operator may cause the system to read and write data in response to entries from the keypad.

4

Patent Number 5185882
Issue Date 1993 02 09
Assignee Westinghouse Electric Corp.
Inventor(s) White, Jr., Harvey H.; Boler, Harry
State/Country MD

Title Bit-slice microprocessor test system

Abstract Test data is incorporated within the microcode of a bit-slice microprocessor to be used during development of the program to verify program performance and during operation of the program as a built-in test. Little additional hardware is required and there is minimal impact on the structure of the program. The program is allowed to operate with the same data that it would have when integrated with the system. During development, the embedded data is used as a substitute for the rest of the system, allowing program development to continue until system integration, using only power supplies and some test equipment. When implemented and used with a commercially available microprocessor ROM emulator, the test data may be varied to highlight difficulties in algorithm design and program development. The operating program cannot tell the difference between live system data and embedded test data. Thus, the program will behave identically during development and system operation. This allows complete algorithm debugging during program development, and permits the rest of the system to be developed in parallel. Developmental test data can be used later for operational program/hardware bit confidence testing with minimal changes.

ment tools is no different. Abstracts 2 and 3 describe a ROM Emulator and a "Circuit Testing Apparatus" in the same broad terms that we've come to expect. Abstract 4 is similar, but at least it is restricted to providing test data in microcode for a bit-slice microprocessor. However, it does go on to discuss use of a ROM emulator and its ability (as we are all aware!) to provide various test data to simulate live operational data. As obvious as many parts of this abstract may seem to us, the patent was issued only a year ago!

Two serious problems in microprocessor testing and emulation are "observability" and "isolation." Intel's patent presented in Abstract 5 addresses the observability issue. Their solution to this problem is to incorporate within the microprocessor itself special circuitry which permits many useful and powerful functions. One is a "capture circuit" to grab (and presumably communicate with) internal address, data, and control signals. A second is a FIFO to store trace history. And the third is flexible state-machine logic which can carry out testing from *inside* the chip automatically. I'd imagine that by incorporating such a sophisticated test circuit within the chip itself, their in-

house task of testing product is greatly simplified. However, it can also become a powerful debugging tool for the user.

The next two abstracts address the "isolation" issue, to which there are really two sides. The first situation arises when one wishes to use an In Circuit Emulator (ICE) on a system in which the microprocessor chip is soldered to the board (or otherwise not conveniently removable). It is necessary to somehow float the on-board microprocessor's output lines so the emulator can take over and run the system. This problem is covered by the patent in Abstract 6.

The converse situation is where it is necessary to test the microprocessor itself while it is yet connected to peripheral circuitry on a board. Here one must isolate both microprocessor inputs and outputs from the other circuitry so that external test signals may be applied and observed under controlled conditions. The patent discussed in Abstract 7 provides for such multiplexing in the embedded system design. It also addresses the issue of driving the peripheral circuitry to a known state. A complete testing paradigm would be one which incorporates the features of

PATENT TALK

Patent Number	4,674,089	5
Issue Date	19870616	
Inventor(s)	Poret, Mark; McKinley, Jeanne	
State/Country	AZ	
Assignee	Intel Corporation	
US References	3,937,938 4,192,451 4,403,287 4,433,413 4,455,654 4,571,677 4,583,179 4,590,550 4,622,669	
US Class	371125 371116 3641200 32473R	
Title	In-circuit emulator	
Abstract	A circuit is disclosed which is implemented on the same silicon chip as a microprocessor to be utilized in a microprocessor system such as a microcontroller, which allows a user to perform in-circuit emulation ("ICE") for the purpose of debugging the microprocessor system. The ICE circuitry comprises (i) capture logic which monitors the contents of the program address register and the internal data bus and various control lines of the processor; (ii) trace circuitry comprising a FIFO buffer which puts data from the capture logic to the output pins of the chip; and (iii) a content-addressable memory and a software-programmable logic array with emulation counters which together function as a finite state machine which performs the desired predetermined testing of the system.	
Patent Number	5226047	6
Issue Date	1993 07 06	
Assignee	Chips and Technologies, Inc.	
Inventor(s)	Catlin, Robert W.	
State/Country	CA	
Title	In-circuit emulation of a microprocessor mounted on a circuit board	
Abstract	In-circuit emulation of a processor that is mounted on a circuit board. The processor is provided with isolation circuitry wherein a particular input signal regime causes all processor outputs to be disabled. The emulator cable terminates in a set of contacts configured to engage the processor pins. In the special case of a surface-mount processor, the contacts are mounted to fit over and around the processor and are spring-loaded. Provision is made via at least one of the probe contacts to establish the specific input signal regime at the appropriate processor pin(s) in order to isolate the processor pins from any processor output signals. This prevents any processor output signals from reaching the emulator or the rest of the board logic, thereby allowing the emulator to operate.	
U.S. Refs	4901259 4939637 4964074	
Other Refs	"Emulators for Microprocessor System Development" by Donnelly et al. 1980 Hewlett Packard Journal pp. 13-20 "Microprocessor-Based Design" by M. Slater pp. 52-54, 1987. "ICE-286 In-Circuit Emulator"; Intel; pp. 48-49. Microcosm, Inc.; "Chapter One-Introduction"; ICE-386 User Manual, pp. 51-53. 3M Electronic Specialty Products; "Surface Mount Test Clip"; p. 258.	

both these patents. It would be able to test both the embedded microprocessor as well as the remaining embedded system circuitry.

Tektronix presents in Abstract 8 a novel means of linking a "mainframe" (presumably a PC) to a microprocessor system under test. The embedded microprocessor's ROM is replaced by a special ROM emulator, a portion of which contains monitor code for debugging. When the emulator detects a particular condition of interest (like a breakpoint), it signals the embedded microprocessor via its interrupt line. This causes a jump to the monitor code where a debug routine specified by the mainframe is executed.

Another type of information that is often useful to the microprocessor debugger is a "trace map" of the execution paths of a program. This is useful in detecting erroneous branching which can occur as the result of improper condition testing or intermediate data results. IBM's patent in Abstract 9 presents a novel way of keeping track of the program flow by using only a single bit per trace point. Before program execution, a bit map is initialized where each bit corresponds to a trace point. When the program encounters the trace point, its corresponding bit in the trace bit map is set. After execution, the map may be used to identify the path of the program. The use of a single bit per trace point reduces both the amount of data to be stored as

PATENT TALK

7

Patent Number 5254940
Issue Date 1993 10 19
Assignee LSI Logic Corporation
Inventor(s) Oke, Timothy P.; Cummings, II, Russell E.; Gavrielov, NachumM.
State/Country CA

Title Testable embedded microprocessor and method of testing same

Abstract A technique of gaining direct access to the inputs and outputs of an embedded microprocessor, otherwise buried behind additional logic, is disclosed. **Multiplexers** are provided for at least the embedded microprocessor inputs and outputs. In a test mode, the multiplexers connect device input and output pads directly to the embedded microprocessor inputs and outputs. In a normal operating mode, the multiplexers connect the additional logic to the input and output pads. Preferably, in order to standardize design criteria, multiplexers are provided on all of the inputs and outputs of the microprocessor which may become embedded behind additional logic. Additionally, it is possible in the test mode to control the additional logic to a well-defined state. The invention provides a simple way to isolate the embedded microprocessor from the rest of the logic and test it thoroughly using test vectors that have already been developed for the stand-alone microprocessor.

8

Patent Number 4,796,258
Issue Date 1989 01 03

Inventor(s) Boyce, Douglas G.; Delegates, Sam M.; Nathanson, Robert M.; Bieber, Timothy E.
State/Country OR
Assignee Tektronix, Inc

US References 4,231,087 4,312,066 4,315,321 4,554,630 4,569,048 4,622,647 4,638,423 4,661,921 4,674,089 4,691,316 4,703,482

US Class 371116 3641900

Title Microprocessor system debug tool

Abstract A microprocessor system debug tool has a mainframe which interfaces with a user. A ROM emulator replaces a ROM unit of the microprocessor system to be tested and has a monitor portion which is used to perform debug functions specified by the user. A user-defined control line is connected to the interrupt system of the microprocessor system to cause the target microprocessor to stop execution of the user's program and jump to the monitor portion upon the occurrence of a user-defined event to execute microprocessor-specific debug code generated by the mainframe in response to the user's input. At the conclusion of debug code execution the microprocessor resumes the user's program. A word **recognizer** is connected to the microprocessor bus to detect the results of the debug code execution, the results being forwarded to the mainframe for display to the user.

9

Patent Number 5,121,489
Issue Date 1992 06 09

Inventor(s) Andrews, Paul N.
State/Country AZ
Assignee International Business Machines Corporation

Title Tracing method for identifying program execution paths using a trace points bit map with one-to-one correspondence with embedded trace points

Abstract An improved method of tracing the paths used in execution of a computer program is disclosed. The method includes using the state of a single bit to denote the referencing of a trace point in the program. The trace points are logically located near the program branch points. One or more bit maps are arranged in a known state at the beginning of program execution and the state of a particular bit in one of the bit maps is set when the associated trace point is referenced. Each bit is associated with a particular trace point according to its position in the bit maps. After program execution, the bit maps are compared to the source listing to determine which trace points were referenced. The use of single bits to denote the referencing of trace points minimizes the degradation of performance efficiency of the target program. Because the bit maps are initialized to a known state at the beginning of each program execution and transferred to retentive storage at the end of each program execution, tracing occurs continuously. Tracing means embedded in a target program and operating according to the aforementioned tracing method, and a method of installing such embedded tracing capability in a target program, are also disclosed.

PATENT TALK

well as the complexity required to set the bits.

Finally, in the area of software source-level debugging, Abstract 10 from Applied Microsystems Corp. uses a pair of special hardware devices to accomplish the task. Their "TAP" (target access probe) is essentially the working part of an ICE. It includes an emulator CPU, RAM [which holds the executable program], and control circuitry. The second piece, the "COMDAP" (communications adapter), links the TAP to the host computer. While details of the system operation are sketchy in the abstract, note the "other references" section at the end of the listing. This is some of the new and very useful information which is now available from the upgraded MicroPatent search software (called U.S. Patent Search Claims & Abstracts). Although the patent is specifically stating that it goes beyond what is covered by the references, these references often provide useful and

Russ Reiss holds a Ph.D. in EE/CS and has been active in electronics for over 25 years as industry consultant, designer, college professor, entrepreneur, and company president. Using microprocessors since their inception, he has incorporated them into scores of custom devices and products. He may be reached at russ.reiss@circellar.com or 70054.1663@compuserve.com.

enlightening background information. This is a real aid to the patent searcher as well as the innovator who uses knowledge from existing patents to motivate and shape his new designs. □

SOURCE

Patent abstracts appearing in this column are from the Automated Patent Searching (APS) database from:

MicroPatent

25 Science Park

New Haven, CT 065 11

(203) 786-5500 or (800) 648-6787

MicroPatent databases include the abstract-only APS version; FullText, which contains the entire patent without drawings; PatentImages, for the complete patent listing including drawings; and other specialized databases for just chemical, computer, or European patents.

425 Very Useful
426 Moderately Useful
427 Not Useful

Patent Number	5228039
Issue Date	19930713
Assignee	Applied Microsystems Corporation
Inventor(s)	Knoke, Robin L.; Johnson, Marvin T.
State/Country	WA
Title	Source-level in-circuit software code debugging instrument
Abstract	A source-level run-time software code debugging instrument includes a target access probe ("TAP") and a communications adapter ("COMDAP") that process emulation commands provided by source-level debugging software operating on a host computer. The TAP includes a TAP CPU that receives target CPU input signals and delivers target CPU output signals for controlling the execution of software code by the target circuit in accordance with command signals provided by the host computer. The TAP also includes a programmable logic cell array and a RAM. The TAP logic cell array routes command and data signals to and from the TAP CPU, and the RAM stores an in-circuit emulation ("ICE") program used by the TAP to operate the target circuit. The COMDAP is physically separate from the TAP and provides an interface between the host computer and the TAP. The COMDAP includes a programmable logic cell array and an EPROM. The COMDAP logic cell array routes command and data signals to and from the COMDAP, and the EPROM stores the commands for configuring the signal paths within the TAP and COMDAP logic cell arrays and stores the TAP ICE program. A flat cable assembly provides a high-speed signal communications link between the TAP and the COMDAP. The TAP uses certain microprocessor signal features and source-level debugging software that runs on the host computer to provide a software engineer with a fully transparent window into the internal functioning of the TAP CPU while executing code in the target circuit environment.
U.S. Refs	4,192,451 4,486,827 4,569,048 4,674,089 4,788,683 4,796,258 4,809,167 4,899,306 4,924,382 4,964,074 5,047,926 5,053,949 5,056,013 5,073,968 5,077,657
Other Refs	Majewski, et al., "Emulator kit multiplies microprocessor choices," 30 Electronic Design, 117-122 (Nov. 25, 1982). Falk, "Emulators keep pace with chip speeds and complexity," 26 Computer Design, 31-38 (May 15, 1987). Everett, "In-circuit emulators keep pace with 16- and 32-bit μ Ps," 32 EDN-Electrical Design News, 252-258 (Jul. 23, 1987). Balthasart, "Development of a low-cost emulator for microprocessor Z80," 95 Bulletin Scientifique No. 4, Association des Ingenieurs Electriciens sortis de L'Institut Electrotechnique Montefiore, 131-136 (1982). Santoni, "Instruments," 26 EDN-Electrical Design News, 212-224 (Jul. 22, 1981). Yen, "Fast emulator debugs 8085-based microcomputers in real time," 50 Electronics, 108-112 (Jul. 21, 1977).

10

CONNECTIME

conducted by Ken Davidson

The Circuit Cellar BBS

300/1200/2400/9600/14.4k bps

24 hours/7 days a week

(203) 871-1988—Four incoming lines

Internet Email: @circellar.com

This month, we'll start with a follow-up discussion centered around a n a r t i c l e that appeared in these pages. The idea is to improve upon a design, though the "better" solution may not always be so.

In the only other thread this month, we entertain a question that can even confuse experienced design engineers: what's the deal with all these different logic families? Are they all really that different, and just what are those differences? Check it out.

Light dimmer control

Msg#:24542

From: ALAIN MARTEL To: ALL USERS

I'm writing in relation with William Von Novak's articles on "Computer Controlled Light Dimmers" which appeared in the March and April '93 issues. I found them most interesting, as I'm currently designing a DMX512 dimmer controller. There are a couple of points which were left incomplete, and about which I'm curious:

1) The design uses coils for power line filtering: one type is used in a four-pole input filter (L3-L6, with C6-C7 in Figure 7a, page 38, April '93), and the other one is in an output filter (L1 in same figure). William says he made them from Micrometals cores. I'd rather find ready-made parts, as I'll be getting in (small scale) production and I cannot make all these coils by hand. What would be a good inductance value for those parts and where can you find them? (I've seen some made by Prem Magnetics; for example, they have a 100- μ H/12-A choke for \$6.50, or a 100- μ H/3.8-A one for \$2.50. Is it much cheaper to make these from cores?) By the way, anybody know where to contact Micrometals?

2) The circuit uses a simplified zero-crossing detector (U38 in Figure 7a) made with an optocoupler to synchronize the microcontroller with the 60-Hz phase, and this approach requires custom adjustment of a time offset—again something I want to avoid in production. I'm using a more precise circuit made with an LM311 comparator, but I'm encountering a difficulty: at low intensities (i.e., when the triac firing is close to the end of the 60-Hz cycle), the small spike that appears in the line voltage when the load is turned on crosses the zero-volt threshold and is sufficient to trigger the sensitive comparator before its time at the end of

the cycle! Even heavy filtering does not seem to totally prevent these spikes. There is always some timing irregularities because of it. Any ideas on how to cure this?

Msg#:25748

From: DAVID MEED To: ALAIN MARTEL

Do you have any info on the DMX512 spec? Where do you get it and how much does it cost!

Contact Micrometals at 1190 N. Hawk Circle, Anaheim, CA 92807, (800) 356-5977, (714) 630-7420.

I have been thinking about the timing irregularities and wondered if maybe you could somehow use a timer or window that will only accept the zero crossing during a small window where it should be. Maybe even set up a timer for exactly 1/120 second and use that for your zero crossing. The opto will be used to make small corrections to the time of this timer, not a blanket zero each time it comes it (integrate the zero crossings...).

Msg#:25790

From: DAVE TWEED To: DAVID MEED

What you are describing, of course, is a phase-locked loop (PLL) with a narrow loop bandwidth. Not difficult at all. Try CD4046, NE565, NE567 for single-chip implementations. You can even use the PLL to multiply the 60 Hz by, say, 512 by putting a 9-stage binary counter in its feedback path (giving 30720 Hz), then using an 8-bit comparator (ignore the MSB) to generate the 120-Hz triac gate signal directly.

If the system in question has a microcontroller with an available hardware timer, you can feed the 60-Hz reference into an external interrupt and do the whole PLL/counter/comparator (or multiple comparators) entirely in firmware.

Now that I look back at William Von Novak's article, the schematic seems to imply that this is what he did. I've never looked at his firmware, however.

Msg#:26978

From: DAVID MEED To: DAVE TWEED

Oh, so that's what a PLL is (my lack of background showeth). In this case, we are after the least complexity, so probably it would be better to do in software if possible.

William's firmware was the "basic get it going, you can add the bells and whistles" type. He may have more

CONNECTIME

advanced firmware in his product, but the stuff here on the board just zeroed the timer at each zero cross interrupt.

Msg#:26298

From: ALAIN MARTEL To: DAVID MEED

The DMX5 12 is a standard protocol developed by the USITT Dimmer Standards Committee and is getting more and more widely used in the industry.

It's basically a balanced RS-422 serial line at 250 kbps, with a string of one intensity byte (i.e., up to 256 intensity values) per dimmer for up to 5 12 (hence the name.. .) dimmers, continuously repeated.

To get the spec, contact Steven R. Terry, Chairman, USITT Dimmer Standards Committee, c/o Production Arts Lighting Inc., 636 Eleventh Ave., NY, NY 10036, (212) 489-0312.

The paper I have is a few years old, so I don't know up to date this info is.. .

The 1 6-MHz 8032 works nicely as a DMX5 12 processor since its serial port can be programmed to operate directly at 250 kbps.

About the zero-crossing problem, a PLL or averaging technique would help things, but I'm still worried about the case when you keep a low-intensity value for long durations: the spike would be very close to the normal crossover point and after a while could become a consistent time reference and shift the PLL time base. I guess the best solution is a good line filter that will erase all such spikes, cutting the problem at its root!

Msg#:25897

From: DAVE TWEED To: DAVID MEED

OK, I took a look at the code. It's interesting that he defines enough memory for eight dimmers, but the interrupt service routine (ISR) only does six of them, and the schematic only shows two. The text of the article says that the CPU doesn't have enough bandwidth to support all eight at 12 MHz.

Both the text and the schematic show 12 MHz as the CPU clock frequency, but the constants in the firmware imply that it must really be 13.5 MHz. That's probably how he got all eight channels working in his prototype.

At first, I thought I could improve on the firmware, but upon closer inspection, I realized that it was about as good as it gets on a 1-MIPS processor. (Lately, I've been doing most of my programming on DSP chips, where 8 MHz means 8 MIPS.)

A similar topic came up here on the BBS some time ago, having to do with generating control waveforms for R/C servos. These devices need timing resolution down to a few microseconds. I came up with a scheme for pregenerating the required waveforms in a 256-byte

memory area and using the data pointer to read them out (8 channels in parallel) at high speed.

You could combine this technique with a software-based PLL for smoother timing to potentially save cycles in the interrupt routines and/or get better timing resolution at the outputs.

Msg#:26980

From: DAVID MEED To: DAVE TWEED

I seem to remember this article was written up from a project he had already done. For the article he only defined two channels in the schematic, but it was modular, so you could do as many channels as you wished.

How did you figure out the processor frequency? I never noticed it (and I was running at 11.052 or whatever). There is a timeout on the zero cross, I think, but that is just purposely set too long so it will never timeout before the next zero cross.

I had thought of a similar idea to your table-based scheme—a table of port values and timer values and you just output the port value and put the new timer value until the next interrupt. It made the interrupt routine very short (almost fit in 8 bytes!), but I never finished the idea.

I am planning to use a PIC for the processor (four channels/PIC). It's just that I haven't gotten the time yet.

Msg#:27018

From: DAVE TWEED To: DAVID MEED

Well, I was looking at the value loaded into the timer: DFh, which is 223 decimal. The only way this makes sense is if you have a chip whose timer runs at OSC/2 instead of OSC/12—in which case $120 \text{ Hz} * 256 \text{ levels} * 223 . 2$ is 13.70 MHz. With a slower clock, at low brightness levels, you'd get a bad flicker because the triacs would only be firing at 60 Hz. The channel counters would never timeout *the second time* before the next zero-cross interrupt. Maybe this is the effect you mentioned before. You should have also seen them jump to half-brightness when the *first* firing was late enough to fall in the second half-cycle.

Amazing. I worked out the same table of port values idea myself a few days ago (combining it with the PLL). However, I found that the interrupt routine required a total 26 CPU cycles, which gives *less* timing resolution than William's original code. The problem is that you can't just pop new values into TLO and THO—you need to do what's called a "precision update" of the timer: You stop it and then *add* the new values into the current values (which in turn implies that the ISR needs to save ACC and PSW). This is because the timer has actually run past zero by some variable number of cycles (because of interrupt latency, etc.), and if you don't account for this, the accumu-

CONNECTIME

lated timing jitter will make the dim lamps (late firing) really flicker.

I think the concept would work just fine on a PIC, however-even without hardware interrupts.

Then I went back and worked out the table-driven waveform idea. The timer is allowed to freerun (like William's version), and the interrupt routine needs just 8 cycles. The drawback, of course, is finding 256 bytes of memory to keep the table in.

Msg#:29234

From: DAVID MEED To: DAVE TWEED

Some random thoughts:

I was thinking (in the preliminary idea stages so far) of just sending the PIC a two-digit value of microseconds since the last zero cross. Then you can do all your fancy dimmer brightness curves in the controller (computer or dimmer control board) and the dimmers don't worry about anything but how long from the zero cross to the firing.

I am just wondering how fine a resolution is necessary to give the appearance of infinite levels (i.e., your eye can't see a jump between levels.) Working out William's equation for 1/2% change in brightness shows about 32 us as the finest needed. But can you see a 1/2% increase/decrease in brightness? I don't know. (Is 100 levels enough? 256? 512? 1000?) From DMX512, I presume 256 is enough that the pros use it.

Msg#:29563

From: DAVE TWEED To: DAVID MEED

I'd be inclined to do the brightness curve in the PIC since these curves are not very linear, and to get a uniform increment in brightness, you'd need significantly more steps in the phase angle resolution. 256 steps in brightness is probably plenty, but 256 steps in phase angle is not. William said in his article, "... no intensity correction is done at the dimmer. This becomes clear at low intensities, where a change of one count is easily perceptible."

William's math in Figure 5 wasn't very rigorous; it should read more like this:

$$\begin{aligned} \text{phase-angle} & \text{ varies from } 0 \text{ to } \pi \\ \text{power-ratio} & = (\text{phase-angle} - \sin(2 * \text{phase_angle})/2)/\pi \\ \text{intensity-ratio} & = \text{power_ratio}^2 \\ & \quad 1 - \exp(-2 * \text{intensity_ratio}) \\ \text{perception-ratio} & = \frac{\text{intensity-ratio}}{1 - \exp(-2)} \end{aligned}$$

Power-ratio, intensity-ratio, and perception-ratio all vary between 0 and 1, where 1 represents "full" power/intensity/brightness for the lamp. You need to put these together, and then solve for phase-angle as a function of

brightness (perception-ratio). Then, take the derivative, which will tell you what kind of resolution you need in phase-angle, for a given resolution in brightness. This is left as an exercise for the reader.. :-)

I wouldn't give up altogether on the "reload the timer in the interrupt" approach on the 8051. The big advantage of this approach is that it gives you single-CPU-cycle (1-ys) resolution on phase angle for a single channel. The problem comes when two channels are very close to (but not exactly) the same brightness: the ISR needs to run once for each channel, but its minimum execution time gets in the way.

There are a number of ways to address this problem. The most extreme solution would be to have some RAM in the code space of the processor and dynamically generate the ISR code on the fly-putting strings of NOP instructions in between two closely spaced updates of the outputs.

I fell in love with the concept of the PIC when I read the first article on it in CAJ, but I haven't actually played around with one. I suspect that with the original chips in the family, phase locking to the power line, driving four triac gates with good resolution, and simultaneously trying to listen to a DMX512 interface at 250 kbps might be impossible. I know that newer members of the family have real hardware interrupts-do any have a hardware serial port? Either or both features would probably make it doable.

Msg#:32345

From: DAVID MEED To: DAVE TWEED

I was thinking of trying for 1 -μs resolution (8,333 steps per cycle). That would translate easily from a 16-bit variable for each channel. Playing with William's math on my spreadsheet indicated that I could get 256 changes in brightness with a minimum of 12-μs resolution, so 1 μs should be plenty for linearity.

I haven't really gotten much past the "preliminary bright idea" stage. I'm hoping I can do it with a PIC and save all the extra wiring for an 8051, latch, and EPROM.

250 kbps would surely be nice, but I would be happy for 9600 or even 31,250 bps (MIDI rate). I've got to get at this Real Soon Now.. .

Msg#:32771

From: DAVE TWEED To: DAVID MEED

I was playing around with the math in MathCAD and got it to graph the overall phase-angle versus brightness curve. It's actually fairly linear over most of its range; say, from about 10% to 90% brightness. Some of the nonlinearities cancel each other out. It does get pretty steep at the ends, though, which is where you'll really need that 1-μs resolution.

I actually plotted brightness as a function of phase angle. I tried to solve the equation for phase angle as a

CONNECTIME

function of brightness, but ended up with the equation in the form $Y = X - \sin(X)$, which is next to impossible to solve for X . MathCAD couldn't do it, and a mathematician friend of mind couldn't offer any insight either. He did correctly point out that you can still build the dimmer without knowing the closed-form solution of the inverse function; just pick the numbers you need off of the graph. My next step will be to see if I can get the curve-fitting functions in MathCAD to come up with a "best" piecewise linear approximation to the curve. If I have any success, I'll let you know.

Msg#:35871

From: DAVID MEED To: DAVE TWEED

At either end of the curve the light is nearly fully on or off anyway and I didn't think a microsecond one way or the other is going to make much difference. It's in the middle that you need to be careful.

I played with the equations and came up with something similar. Nice to know I did it right and I haven't forgotten all of my high school algebra.

I've got a spreadsheet where I did the equation in 250 steps (0, 0.4, 0.8, 1.2...) and got the numbers. Figures out to about 12 μ s at the finest resolution.

Logic families

Msg#:36635

From: JOHN DEARDEN To: ALL USERS

On returning to designing hardware having been away about ten years, I notice a whole slew of new logic families available: HC, HCT, ALS, HC, and so forth. Is there any reference, a magazine article perhaps, that divulges the characteristics of each family? Are particular logic families better for glue logic for microprocessor projects than others?

These questions are prompted by some earlier messages I read, where someone changed a glue logic chip from one family to another and ran into unexpected problems.

Msg#:36656

From: GARY CORDELLI To: JOHN DEARDEN

I don't know about a text on the subject, although I'm sure there is one that's already out-of-date being used in most colleges :-)

If I were you, I would just get some databooks for logic ICs from National Semiconductor or TI for each of these families and check out the sections (usually in the front) that describe the characteristics of the family. Then you can look up the same part across the families in each book to get a good comparison of what is different and you will

probably be able to figure out yourself where one family would be better or worse than another or why you would choose one over the other for a specific task. Start with something simple like a 7408 as an example of plain gates and a 74163 as an example of clocked logic. I could give you some cursory differences between the families here, but I think it would do you a disservice-I couldn't possibly cover in a BBS message what you could conclude after checking out the complete story in the databooks.

The short answer is that ALS (advanced low-power Schottky) gives you higher speed at about half the power of plain LS; AS (advanced Schottky) is plain S on steroids; HC (high-speed CMOS) gives you faster performance than plain C for only a *tiny* increase in power; HCT is HC with TTL level translation to allow interfacing to TTL logic families (TTL, LSTTL, ALSTTL, etc.); AC and ACT are members of the FACT family, which is Fairchild's version of HC and HCT; F is the FAST family (originally from Fairchild, too) which generally gives you a pin-compatible plain S part with higher speed yet 3-4 times lower power dissipation.

If you are looking for data books that actually give you some family characteristic background in addition to the individual part data, I'd recommend National Semiconductor's ALS/ASdatabook over TI or anyone else's that comes to mind. Good luck.

Msg#:37756

From: MARTIN VUILLE To: JOHN DEARDEN

A slight correction to Gary's message:

FACT is *not* the same as HC/HCT. The AC/ACT parts are "Advanced CMOS" and "Advanced CMOS TTL-compatible." They are faster than the HC/HCT parts, and their edges are also faster. The available drive current is much greater than HC/HCT (except buffers/drivers).

The significance of faster edges is that the edge transition time (rather than the chip propagation time) is what determines when circuit connections can be looked at as lumped elements or as transmission lines. This can be a cause of grief when "plug-compatible" parts of a faster logic family are used to replace a slower family. That is, if you take a working circuit with LS parts and replace them all with ACT parts, the circuit should still work, on paper, but may not work in real life because of reflections and other transmission line effects.

Msg#:37947

From: GARY CORDELLI To: MARTIN VUILLE

You are, of course, right about the FACT stuff. I was trying to be brief and got mentally lazy. The thought I wanted to convey was that FACT and HC/HCT both took CMOS in the same direction. FACT is more Fairchild's "answer" to HC/HCT than its "version" of it (ignoring

CONNECTIME

chronology). The FACT stuff is definitely "better," although with the caveats you mentioned about the faster edges screwing up a design that worked with slower logic.

I'm an engineering consultant, and I've had to fix a few companies' designs that used FAST parts where they could have used LS, but thought it would be "safer" to use the faster ("better") part. I found it is really hard to convince some digital-heads that digital is more like a "mode" of analog circuit operation than it is a whole different beast. The road from 0 to 1 and back is sometimes a winding one.

Msg#:37035

From: JAMES MEYER To: JOHN DEARDEN

In addition to the technical help that seems to be already coming your way, you need to be aware of the sometimes spotty supply of some types of chips in some of the families. It makes little sense to find and design around the absolute best specs if the chip is only available in huge quantities and the lead time is six months.

Msg#:37264

From: ED NISLEY To: JAMES MEYER

You mean like HCT parts that are all of a sudden unavailable? The story I got is that National is dropping their HCT line, replacing it with something called VHCT, and you can't get the old ones, can't get the new ones, and can't get VHCT databooks either.

I'd been thinking about offering a bunch of logic for the Graphic LCD Interface, but gave up after a few phone calls to my friendly neighborhood parts pushers. You need a full-time purchasing department to keep up with that stuff!

Msg#:37557

From: JAMES MEYER To: ED NISLEY

You got it. The 7400 isn't a very sophisticated part. But a better part that you *can't* get is worse, no matter how good it looks on paper. If I can't find it in the Digi-Key catalog, it *don't* go into none of *my* designs. 8-)

Actually, if I can't find it in my junk box, it usually don't go.

Msg#:37916

From: ED NISLEY To: JAMES MEYER

I get a call every now and again from somebody asking where to get parts. I find I'm aiming 'em at Digi-Key more often; they've pretty much reached critical mass.. if only they had better microcontroller stuff!

Somehow I have trouble thinking of a 74LS245 as Pure Unobtainium material!

Msg#:38417

From: JAMES MEYER To: ED NISLEY

Humidity sensors, \$20 optical rate gyros, CCD camera parts, voice synthesizer chips, TVRO converters, radar waveguides, tunnel (Esaki) diodes, 250-kV Tesla coil caps, GM tubes, and so forth are *real* Unobtainium candidates.

Have you considered handling any Army or Navy surplus stuff? There isn't anything closer than 1,000 miles that I can find.

Msg#:38627

From: ED NISLEY To: JAMES MEYER

.andh.if I ever come across a stash of those old TI sound chips, I can retire the next year!

I've got a rather limited "vision" for Pure Unobtainium. There are lots of surplus dealers out there already, so that niche is taken. There are lots of junk dealers, and I can't afford to stock up on that stuff. There *aren't* a lot of tiny volume sources for the current production neat things that show up in INK.. that's the niche.

The fact of the matter is that I need to sell roughly 100% of the stuff I stock, in fairly short order. If I buy a stash of, say, radar waveguides and sit on them for years while the five guys who need waveguides and who aren't connected to the usual surplus outlets, I may as well keep the money in a CD and be done with it.

But, yeah, if I ever find some good humidity sensors..

We invite you 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 Email to info@circellar.com.

ARTICLE SOFTWARE

Software for the articles in this and past issues of *The Computer Applications Journal* may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only \$12.

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 8752199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

428 Very Useful 429 Moderately Useful 430 Not Useful

STEVE'S OWN INK

Superhighway Limits



Like many people in our industry, I subscribe to a number of trade journals. While I might argue that some of what others publish is vaporware, reading them at least updates me on the directions and developments in the industry. I'll even admit that some of this timely information has been instrumental in winning a contract or designing a new product.

But, unless you are heating your house with rolled magazines, there is a point of diminishing returns. Those of us with a wide range of interests soon find ourselves amid mountains of paper as too many publications start stacking up all over the place. Even browsing through them becomes tedium. I've concluded that I just have to temper my choices from the 96 periodicals that presently show up per month to only the ones that are really of value.

The same holds true for nontraditional sources of information, like the Internet (sometimes inappropriately called the "information superhighway"). Recently the Circuit Cellar BBS gained access to the net and messages have been piling up on our little off ramp at a rate of over 350 per day (up from about 30-50 per day, and equivalent to about 150 typewritten pages a day)! While this little information pile-up may not seem like too much, the problem is that a lot of it is of no specific interest or concern to me. So, like having too many magazines again, I now find that the challenge is in determining which of these threads has any relevance and which messages I read.

I guess what I need is a personality server that can scan all of this incoming traffic for keywords or specific discussion topics that I would find interesting. Given my saturation status on even a minimal introduction to such networks, however, it would seem to follow that before any information superhighway becomes of any tangible advantage to most of us, there must be some suitable filtering mechanism that strips the drivelt out of the traffic. I have a real problem spending time reading gigabytes of junk mail.

Speaking of filtering, are we really sure we want the government stepping up their involvement in the information superhighway proposals? It was a stroke of genius when the DOD started the ARPANET many years ago, but if it weren't for a lot of private individuals and companies getting involved in its advancement, do you think the Internet would be what it is today? I suspect what is really happening here is that Washington senses the emergence of a new kind of democracy, and is promoting its own involvement in order to protect its own interests. I offer the Clipper debacle as evidence. While not declaring that there is an insidious unspoken agenda, I am always suspect when so much effort is expended merely to get one's fingers into the pie. The Internet is doing just fine without government.

I suppose if I weren't a damn Yankee I wouldn't be so concerned about who controlled the information filter. If no one else does, we better get ready to welcome the new Department of Politically Correct Data Transfer when it joins the IRS in making our lives more exciting.

