CIRCUIT CELLAR INK®

# THE COMPUTER APPLICATIONS JOURNAL

## EMBEDDED CONTROL

The Covert Chordic Keyboard

RF Transponder using the DS1209

Embedded Borland and Microsoft C Code

Design Contest Winners

December 1993 — Issue #41



12
0 74470 75349 0

$3.95 U.S.
$4.95 Canada

# EDITOR'S INK

## Our First Month of Email



It has certainly been an interesting month. We made our first successful connection between the Circuit Cellar BBS and our Internet host two days before I wrote my editorial for the last issue. It worked so well that I took a gamble telling everybody about the connection so early in the testing. It turns out my gamble paid off. We've had no major problems with the connection, and usage has blown well beyond my expectations.

The kind of traffic we've been seeing has also been very different from what I expected. Those on the outside coming in have been requesting the information file, the list of available files, and an occasional article-related file. These are all handled automatically by the system. I've also received many very nice notes from readers welcoming us to the outside world.

Users of the BBS looking out have been sending a lot of test messages to try out the new gateway (as I didexpect). Many are sending notes to colleagues with whom they haven't made contact in a long time. A handful have subscribed to mailing lists (which alone can increase the message traffic dramatically and is why I try to keep such subscriptions to a minimum). We even have some parents keeping in touch with kids away at college.

What I haven't seen as much of as I'd hoped for is reader feedback about the magazine and its content. Mail from our readers has always been on the slow side. I expected that opening up a new, more convenient avenue might bring in more letters to the editor, but so far it's still been slow. If you haven't taken the time to request our information file (send Email to info@circellarcom) or send your comments, drop me a line (ken.davidson@circellarcom). I'd love to hear from you.

One area where we did get excellent turnout is our Fifth Annual Circuit Cellar Design Contest. We had more entrants this year than last, making the judging even harder. There was a definite increase in PIC-based projects, which shows a tendency toward compact, low-cost solutions to problems that could only be solved with much larger processors not too long ago. I want to congratulate all our entrants and winners on a job well done and encourage you to write full-blown articles about your projects so all of our readers can see just what you've accomplished.

In our features this month, we have two articles that deal with Dallas Semiconductor parts (which always seem to show up in embedded designs). The DS1209 can be used in an RF transponder, but there are some tricks the data sheet won't tell you; and adding assembly language routines to your C code to access the clock/calendar in the DS5000T is easy if you know the right steps. Next, take notes in a crowded lecture hall or on a bumpy bus without anyone knowing it using the Covert Chordic Keyboard. Finally, we conclude our two-part introduction to field-programmable gate arrays.

Ken


CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Michael Swartzendruber

ASSOCIATE EDITOR
Rob Rojas

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Susan McGill

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street. Suite 20, Vernon, CT 06066 (203)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 0. Box 7694, Riverton, NJ 06077 or call (609)786-0409.
POSTMASTER: Please send address changes to The Computer Applications Journal, Circulation Dept P 0 Box 7694, Riverton, NJ 08077



**HAJAR** ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

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

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

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

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

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



Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/144k bps, 8 bits, no parity, 1 stop bit, (2033871-t 986.24001 9600 bps Courier HST, (203) 871-0549



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

*Circuit Cellar INK* makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects. Circuit Cellar INK disclaims any responsiblity 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 © 1993 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.

14 Secrets of Using the DS1209 in an RF Transponder
*by Maurizio Ferrari*

22 The Covert Chordic Keyboard
*by Scot Colburn*

30 Accessing the **DS5000T** Timekeeper from C
*by J. Conrad Hubert*

40 Designing with **FPGAs**/Part 2: An Example
*by Del Hatch*

48 **Results of**
The 5th Annual Circuit Cellar Design Contest

52 ❑ Firmware Furnace
Beyond Small: Mainline C for the '386SX Project
*Ed Nisley*

60 ▣ From the Bench
Measuring Up an Electronic Caliper Interface
*Jeff Bachiochi*

64 ❑ Silicon Update
It's Showtime
*Tom Cantrell*

72 ▣ Embedded Techniques
Conversing with Batteries
*John Dybowski*

# INSIDE ISSUE 41

2 Editor's INK
Ken Davidson
Our First Month of **Email**

6 Reader's INK
Letters to the Editor

8 New Product News
edited by Harv Weiner

77 Patent Talk
Russ Reiss

**ConnecTime** 84
Excerpts from Circuit Cellar BBS conducted by Ken Davidson

Steve's Own INK 96
Steve Ciarcia
Embedded in Everything

Advertiser's Index 81

# READER'S INK

## Fluke Pickup

I thoroughly enjoyed Derek Matsunaga's article on the data acquisition system for the Fluke 80-series multimeters (October '93). My favorite type of design work is finding unusual ways to use undocumented "features" of hardware, whether at the component or at the system level, and the article struck close to home.

I would, however, like to offer some suggestions concerning the front end of his system. Derek mentions his use of a telephone "microphone" to pick up the meter's acoustic output, and its attendant pickup of EMI/RFI. In my experience, these transducers are not microphones (in that they do not respond to acoustic energy) at all, but are induction coils wound around a permeable core. They respond instead to the varying magnetic field present in the dynamic (moving coil or moving iron) transducer in the earpiece of the telephone receiver. As such, they make very good antennas when properly oriented!

I believe a far better solution would be to use one of the miniature electret capacitor omnidirectional microphone elements, such as the Panasonic WM-60AY sold by Digi-Key, in close proximity to the multimeter's loudspeaker. The necessary supply voltage for the internal FET follower is already available in his system, as is the required amplification, and the microphone's wide frequency response and reasonable sensitivity should yield more than satisfactory results. Stray EMI/RFI pickup shouldn't be any problem, while stray acoustic pickup will be minimized by the small working distance. Retuning the already-present single-pole high-pass filter to just below **16** kHz will provide an extra measure of protection from low-frequency acoustic interference.

Steve Hebrock
Akron, Ohio

## Furnace Safety

I enjoyed your article on furnace control in the October issue. This is the sort of article I like seeing (i.e., a comprehensive treatment of the overall subject written so as to be generally understandable). I am going to get me one or more of the R2868 tubes and play!

Without wishing to be critical of the article per se, I would like to say that I do not think such an article should be published without some reference to the danger of explosion and safety measures to prevent same. If at any time, the flame has been lost and a combustible fuel/air mixture has continued to flow into the combustion chamber [CC), even for a short time, there is a strong probability that the CC will remain filled with an explosive mixture for quite some time and any attempt to restart the furnace before the CC has been purged will very likely result in a serious explosion. I have seen a 300-megawatt water tube boiler with all of the furnace wall tubes bent outward by such an explosion,

I appreciate that the design is a simple one, not for industrial use, but I do think any application of the system should include at least a simple protective device which can be easily applied using some fairly easily written software. I believe that a simple pressure switch, capable of sensing the burner channel back pressure just downstream of the fuel feed tube orifices to guard against any possible blockage which restricts combustion air flow but not fuel flow. With such a switch, we could furnish software which would, on sensing loss of combustion air pressure, shut down the furnace and could also be used to validate a preignition CC purge cycle which should be implemented at any starting or restarting of the furnace. I noted in the article a mention of pilot lights, but cannot remember any reference to these being included in the control description. Quite obviously if the furnace control is off/on as described, then the pilots have to be visible to the R2868 tubes and remain lit while the main fuel is off.

The software associated with the combustion air pressure switch could be configured as follows:

1) At power up, if the flame sensor is true (i.e., it says that a flame is present), then under no circumstances should any attempt be made to start the furnace and an alarm of indication signifying lockout should be effected. Possible causes are R2868 tube failure or afterglow due to visible or infrared radiation from some furnace system component which has been heated by furnace combustion. At this point, if the combustion air fan (CAF) is running, then it could be left running-with the fuel valve closed.

2) Whenever it is desired to start or restart the furnace (other than controlled admission of fuel subsequent to a short fuel-off period during which pilots have been continuously ignited), then the CAF should be run [with continuous validation by the pressure switch) for a timed purge period (say about 30 seconds for such a small furnace) in order to remove combustible mixtures from the CC before an ignition attempt.

3) After satisfaction of condition (2) above, then ignition can be attempted by energizing the spark plug(s) and opening the main fuel valve. If a flame has not been detected after, say, 15 seconds, then the sequence should

# READER'S INK

be aborted with a new prepurge sequence being run before another ignition attempt.

4) After flame has been continuously present for, say, three seconds, then the spark plug(s) can be deenergized and normal operations assumed.

5) After normal operation is commenced/assumed, then shutdown should be effected on any detection of loss of flame for more than, say, one second or on failure of CAF as signified by the combustion air pressure switch.

Although I hesitate to suggest more complex controls, I think it might also be desirable to implement a simple thermocouple upscale burnout features. If the thermocouple shorts somewhere in the circuit or goes open circuit (either of which would be interpreted as a low temperature without this feature), then this would be interpreted as a high temperature which would restrict fuel flow or fuel valve open time according to which control system was adopted.

Bill Brown
Katy, Tex.

## 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) 8752199.

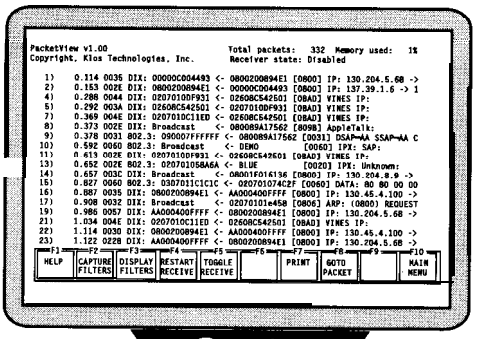
Fax: All faxes may be sent to (203) 872-2204.

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.


Introducing: PacketView™

PacketView is thsfint truly low-cost network analysis and debugging tool available for the PC! Supporting Ethernet, Token-Ring, and ARCNET, PacketView can go places noother software-only analyzer can! PacketView is an essential toolwhen developing network driven, protocol stacks, routers, and many other network based software products! And at only $299 how can you go wrong? Download the demo from our BBS at (603) 429-0032 and see for yourself just




Only $299




Features:

- Protocolsrecognized: TCP/UDP/IP, SNMP, Novell IPX/SPX/ NCP, AppleTalk, XNS and Banyan VINES
- Use your own PC and Ethernet network adapter (in most cases)
- Programmable Filters for Capture and Display
- Custom Protocol Decoderscan be developed with C or Assembler
- 24-Hour BBS and 6 Months Free Software Updates
- SerialView Coming Soon! For SLIP and PPP Protocol Analysis

Klos Technologies, Inc.

604 Daniel Webster Highway, Merrimack, New Hampshire 03054
Voice: (603) 424-8300 FAX: (603) 424-9300 BBS: (603) 429-0032

03



Embedded PC with on-board Ethernet and Super VGA




- 25 MHz 386SL™ CPU; including up to 16 MByte DRAM
- On-board Super VGA LCD/Video controller
- On-board Ethernet, Featuring AUI and 10 BASE•T interfaces
- On-board SCSI, Floppy controllers and 2 MByte Flash Epram Solid State Disk
- 3 Serial Ports, Parallel/Printer port

PC/II+

4"x 4" Small Rugged Format

For more information call:
Megatel Computer Corporation
125 Wendell Ave., Weston, Ont.
M9N3K9 Fax: (416) 245-6505

(416) 245-2953

Megatel is a registered trademark of the Megatel Computer (1986) Corporation 386SL is a trademark of Intel Corporation

megatel

#104

# NEW PRODUCT NEWS

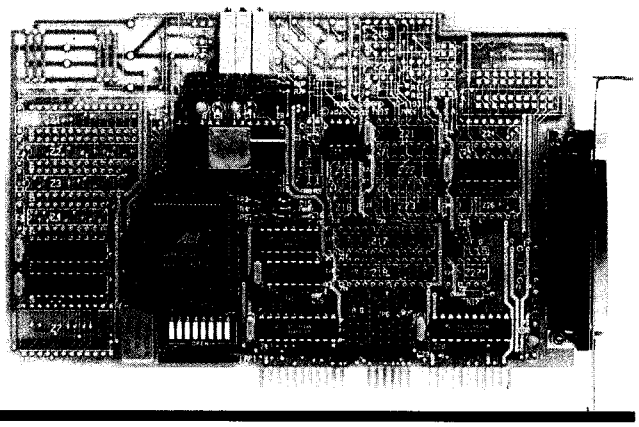
Edited by Harv Weiner

## LOW-COST DATA ACQUISITION BOARD

ADAC has announced a high-speed data acquisition board for the PC/XT/286/386/486 family of computers. The **5508LC** is one of the lowest priced A/D converter boards available and features 100-kHz A/D throughput with 12-bit resolution. The half-sized board contains eight analog inputs and offers automatic channel scanning.

The 5508LC eliminates hardware and software compatibility problems with a unique Universal Software Interface (USI). USI allows true cross compatibility between software and hardware from different vendors. Through a unique use of custom ASIC technology, USI enables ADAC's plug-in boards to change their software interface to the PC bus. This enables the cards to match the Control and Status Registers of other manufacturers' plug-in boards and become instantly compatible with their software.

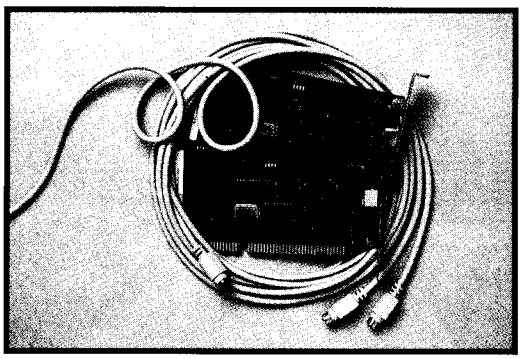
USI also provides compatibility with virtually every DOS- and Windows-based data acquisition and control software package available. Some of the software packages that the 5508LC is compatible with include ADAC's Adlib PC DOS drivers, LabVIEW and LabWindows from National Instruments, Driver LINX Windows driver package, Genesis, ASYST, Control EG, and Keithly/Metrabyte drivers. The 5508LC High-Speed Data Acquisition board sells for **$245**.



**ADAC Corp.**
**70 Tower Office Park**
**Woburn, MA 01801**
**(800) 648-6589**
**Fax: (617) 938-6553** **#500**

## SINGLE-USER PC NETWORK

PC InterConnect has introduced the **One-Man-LAN**, a single-user personal computer network. This product is for the individual who works with more than one PC and would like to have LAN capabilities without the complexity and expense of a conventional LAN. It also can be used to keep an older computer from becoming obsolete.



With One-Man-LAN, a user seated at a main (or primary) PC can access files located on one or more secondary PCs as if they were located within the main PC. Any of the user's PCs can be temporarily converted into a secondary (or server) PC by typing "OML." Typing "Q" turns it back into a primary PC. The user can also redirect output to a printer attached to a secondary machine.

The One-Man-LAN hardware implements a high-speed, bidirectional serial data communication path that can interconnect PCs point-to- point or through one or more hubs. A PC can be interfaced using either an add-in card or a special parallel port interface module. The data transport medium is an industry-standard 8-conductor cable and is usable with lengths in excess of 100 feet. Extremely fast data rates are maintained; files on a second, third, or fourth PC can be accessed as rapidly as if they were in a network file server.

One-Man-LAN is compatible with sector-level disk utility programs, and such programs handle disk volumes installed on the secondary PC as though they were installed on the primary PC. DOS commands such as DISKCOPY are not usable with conventional LANs, but are fully functional with One-Man-LAN.

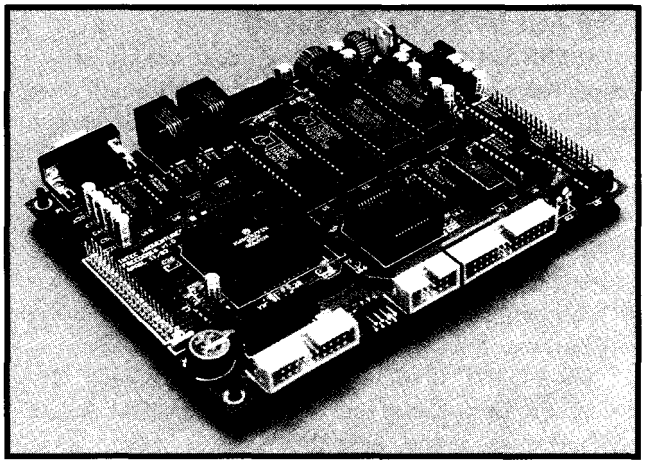
One-Man-LAN consists of two half-size ISA add-in cards, a thin 12-foot interconnect cable, software, and an installation and user guide. Additional cards and cables are available to allow connection of additional PCs. One-Man-LAN sells for **$199**.

**PC InterConnect, Inc.**
**106 Library Plaza • 15 North 100 East • Provo, UT 84606**
**(801) 374-8880 Fax: (801) 374-2306**

**#501**

# NEW PRODUCT NEWS

## FULL-FEATURED MICROCONTROLLER



A powerful, general-purpose, full-function control and data acquisition system has been announced by Intec Inoventures Inc. The Freedom **16** microcontroller offers speed, memory, and I/O capabilities to address virtually any requirement.

The Freedom 16 is built around the 16-bit Motorola M68HC 16, a chip designed specifically for microcontroller applications. Control of specific features such as pulse counting, high-speed inputs and outputs, PWM, and 8 channels of fast 10-bit A/D conversion are included, as well as DSP capabilities. RS-232, RS-485, debug ports, a dedicated LCD and keypad port, 34 programmable digital I/O channels, and an expansion header (containing both Motorola and Intel Read & Write signals) are also included. With its on-board regulation, the entire board can be powered by a battery or common AC adapter.

Programmability in C and up to 256K bytes each of RAM and flash memory (64K of each is standard) along with an extensive, well-documented library of functions makes for fast, reliable development. By using flash memory and a novel debug cable, the Freedom 16 frees the designer from the need for an expensive emulator and the programming of EPROMs.

Development tools include IASM16—an MS-DOS based editor, cross-assembler, and communications package blended into a single environment. This is combined with the Dunfield C compiler to provide in a single package everything needed for software development in C, assembler, or combinations of the two. Preprocessor, optimizer, and many utilities such as MAKE and TOUCH are included with the package.

The Freedom 16 can be used as an in-circuit debugger/emulator by using P&E's ICD16 software and ICD (in circuit debugging) cable. The ICD software talks to the M68HC16 processor's background mode via the ICD cable which connects to a standard PC printer port on one end and a Motorola 10-pin Berg connector on the target system. With this interface, any target system processor can be turned into its own in-circuit emulator/debugger. The ICD16 runs on any MS-DOS-based PC with 512K RAM and a printer port. The Freedom 16 microcontroller sells for $299 (U.S.). The controller with all the necessary C software development tools sells for $499 (U.S.).

Intec Inoventures, Inc. • 2751 Arbutus Rd. • Victoria, B.C. V8N 5X7 • (604) 721-5150 • Fax: (604) 721-4191 **#502**

## MOTORIZED WINDOW CONTROLLER

The Win-Trol M-2100 converts virtually any crank-operated window or skylight into a power window. Casements, awnings, vents, and roof or skylight windows can be easily motorized whether they are made of aluminum, wood, vinyl, or composites. The unit can be installed in new construction or retrofitted to existing windows or skylights in commercial or residential applications. The M-2100 is operated with a simple touch-and-run switch.

The conversion system requires one wall switch-style mounted control unit for each individually operated window/skylight. For groups of windows, one controller will operate as many as three motor-equipped windows sequentially.

The motor is encased in a unit that attaches to the window to be controlled, replacing the crank handle. The unit has proven extremely reliable and the cover is available in several colors. The wall switch controller unit replaces a standard light switch and is connected to each motor to be controlled.

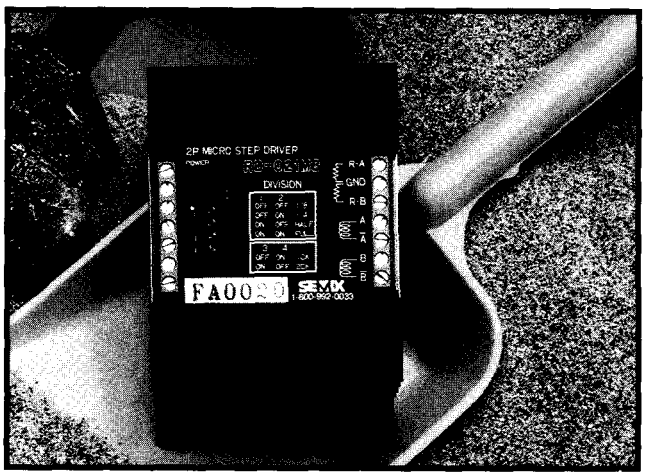
Other options to the M-2100 include electric latches, rain sensors, battery backup, power distribution for large applications, connections for external triggering (X-10, thermostats, humidistats, home automation systems), and remote control.

The Model 2100 Wall Switch Controller Unit retails for $170 and the Model 2100 Motor with Cover and Installation Hardware retails for $180. Quantity discounts are available.

Win-Trol, Inc.
P.O. Box 4425 • Helena, MT 59604
(406) 449-6616 • Fax: (406) 449-3666 **#503**

# NEW PRODUCT NEWS

## COMPACT MICROSTEP DRIVER




SEMIX has announced a low-current and voltage stepper motor driver that features extremely small size and high performance. The **RD-021M8** is an inexpensive driver that is capable of selectable microstepping and provides a selectable clock.

The low current (0.1 to 1.5 amp/phase) makes the RD-021M8 ideal for 11 to **23** frame size stepper motors. The driver accepts voltages as low as 10 V and as high as 40 V, allowing it to be used in applications powered by a **12-V** battery.

The RD-021M8 measures 1.25" x 1.97" x 3.15", allowing it to be designed into tight places. It can execute full, half, quarter, and one-eighth steps to achieve as many as 3200 steps/revolution with a 0.9" motor.

Additional features include photocoupler-isolated inputs, selectable current settings with external resistance, low-voltage protection circuitry, and automatic current limiting (one-half of full current when motor is stationary). The RD-021M8 sells for $175 and is available at quantity prices.

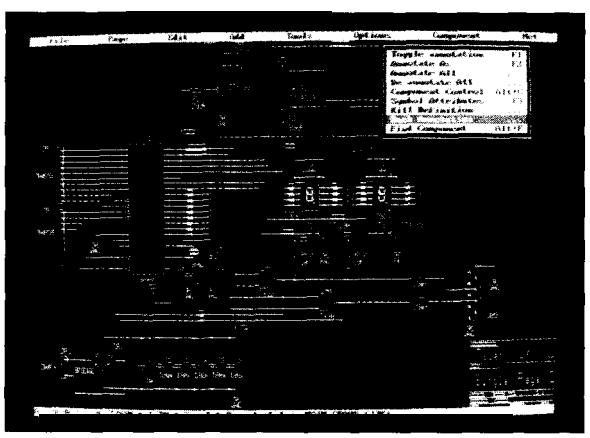
SEMIX, Inc. • 4160 Technology Dr. • Fremont, CA 94538 • (510) **659-8800** • Fax: (510) 659-8444 **#504**

## SCHEMATIC CAPTURE PACKAGE

Tsien has released an intelligent schematic capture package, **BoardCapture**, for electronics engineers. This powerful new PC-based software tool completes the company's track layout and autorouter packages. The complete bundled BoardCapture/BoardMaker/BoardRouter system provides an end-to-end design-to-manufacture environment for both prototype and production requirements at a surprisingly low cost.

BoardCapture automates the generation of schematic circuit diagrams, with direct output of netlist data for BoardMaker. Completely hierarchical, the package will accommodate virtually any size design, from simple modules with a few components to the most complex designs requiring A0 or larger schematics.

One of BoardCapture's most powerful features is speed of operation. Everything the designer needs, from device libraries to the BoardMaker PCB CAD package, resides in memory simultaneously, allowing schematic creation in one fast, fluid operation. This includes smooth scrolling around drawings and multipage diagrams.



Tsien has taken particular care with the look and feel of fundamental schematic creation mechanisms. Connectivity information is available instantly on every element of the schematic by simply pointing and clicking, even if connections are on other drawing pages. Bus and port naming is a further basic procedure that has been automated, freeing the designer to concentrate on the core task.

Software will automatically flag a host of potential errors from overshoots to bad design practices. Also offered are nested undo/redo facilities as well as context-sensitive operation.

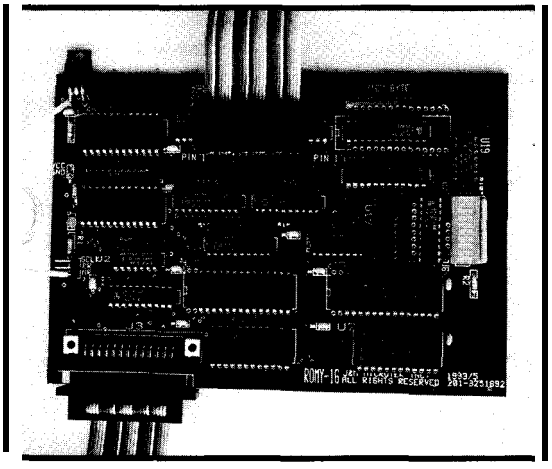
BoardCapture comes complete with a comprehensive device library and powerful editing tools, including a graphical library editor for creating free-format symbol outlines, and a table-based editor to simplify the entry of pin details. BoardCapture sells for **£395** (approximately $612 U.S.) and the complete bundle of BoardCapture, BoardMaker, and BoardRouter sells for **£795** (approximately $1230 U.S.).

Tsien (UK) Limited
Aylesby House • Wenny Rd.
Chatteris • Cambridge PE16 6UT • United Kingdom
Tel: **+44 (0)354** 695959 • Fax: **+44 (0)354** 695957 **#505**

# NEW PRODUCT NEWS

## EPROM EMULATOR

J&M Microtek has released the **ROMY-16 EPROM Emulator,** which is an emulator for EPROMs or static RAMs for 8- or 16-bit microcontroller systems. With the ROMY-16, program code can be downloaded, tested, and results seen immediately without the need to remove the EPROM, reprogram it, and reinstall it in the target system.



The ROMY-16 emulates EPROMs from 2716 to 27010 and static RAMs from 6116 to 628 128. For EPROM emulation, a PC can be used to read data from and write data to the ROMY-16 memory while the target CPU reads from the memory. For RAM emulation, both the PC and target CPU can access the emulator's memory. The ROMY-16 allows the size and type of memory to be varied without hardwire changes.

For 16-bit microcontrollers, the ROMY-16 is set up to emulate two EPROMs which share the same address lines but have separate data lines. With S-bit microcontrollers, the lower byte memory is mainly used to emulate the target ROM or RAM. Eight pins (D8–D15) on the higher byte memory can be used as hardware breakpoints to provide triggers to a timer, two interrupts (IRQ and NMI), or to provide a halt signal to the CPU.

The ROMY- 16 EPROM Emulator sells for $195 (for EPROM types 2716–27256) and $245 (for EPROM types 2716-27010). The optional UMPS V2.1 program sells for $100 each CPU.

J&M Microtek, Inc.
83 Seaman Rd. • West Orange, NJ 07052
(201) 325-1892. Fax: (201) 736-4567 **#506**


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

- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 805 1/803 1 or 8052/8032 CPUs*
- *Compatible with any RAM or ROM memory mapping*
- *Runs up to 50 times faster than the MCS BASIC-52 interpreter.*
- *Includes Binary Technology's SXA51 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51 $ 295.***

508-369-9556
FAX 508-369-9549

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

#105

Build your own Neural Net!
- and Expert System!

**EASY NEURAL NETWORKS**

TSI **Easiest way to quickly learn about this fascinating new technology - includes a working Neural Network you can train! $59 ****

**EASY EXPERT SYSTEMS**

**Build your own Expert System in about one day! Includes 40-Page Knowledge Engineering Manual, Quick Ref Card for quick start, and powerful Stock Market Expert System! $69 ****

***BOTH $99 + Includes 3 BIG Catalogs and FREE Talking Expert System Demo + Order now and receive a free copy of talking PC Therapist - usually $64.95***

**3 Catalogs & Demo Disk - Send your check for $5**
**We always pay postage**

**Please specify disk size or we ship 3.5"**
**- Check, American Express, or P.O. to:**
***THINKING SOFTWARE, INC.***
***46-16* 65TH PLACE - Dept cc201**
**WOODSIDE, N.Y. 11377** PHONE **(718) 803-3638**

#1(

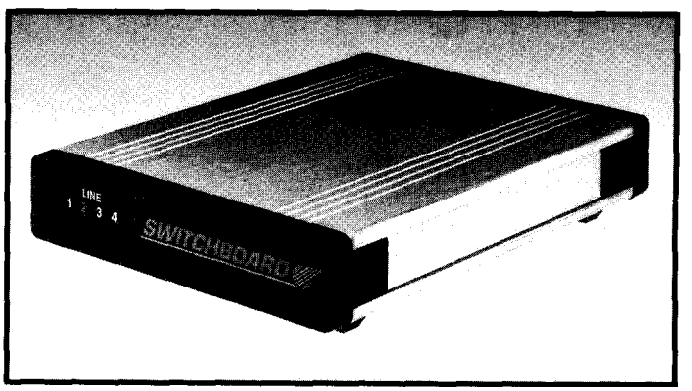
# NEW PRODUCT NEWS

## PHONE LINE SHARING DEVICE

Telephone Products has introduced Switchboard, a telephone-line sharing device that allows customers to use one phone line for phone, fax, and modem communications, each with an individual number. Switchboard offers an inexpensive, easy way to give a home office the professional image of a larger office-separate numbers for fax and phone, with no clicks or tones.

Using the local phone company's "distinctive ringing" service, Switchboard automatically sends an incoming call to the appropriate device. Distinctive ringing services are known in different areas by names such as Smart Ring, Custom Ringing, Personalized Ring, Ident-a-ring, Intel-a-ring, Ring Mate, Multi-Ring, and Ringmaster.

Switchboard is packaged in a 6.5" x 4.5" x 1.25" case and weighs only 12 ounces. It needs only to be plugged into a phone jack; it requires no additional power and is not affected by power surges or outages. Switchboard works with any combination of two, three, or four devices, including phones, answering machines, faxes, or modems. Using only one phone line, instead of as many as four, can save users hundreds of dollars per year. Switchboard (Model T-4) sells for $129.00 and is backed by a five-year warranty.




Telephone Products
P.O. Box 31203
Seattle, WA 98103
(800) **829-5960**
Fax: (800) **829-3940**

**#507**


SU-MAR ENTERPRISES

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

| | | |
|---|---|---|
| x-10 | HOME AUTOMATION MODULES & CONTROLLERS | |
| LEVITON | HOME AUTOMATION MODULES & CONTROLLERS | |
| STANLEY | X-10 COMPATIBLE GARAGE DOOR OPENER | |
| RCS | X-10 COMPATIBLE 4 CHANNEL DPTP RELAY $160 | |
| REDAC | ADD DIGITAL INPUTS. ANALOG INPUTS & RELAY OUTPUTS TO X-10 | |
| ECS SOFTWARE | DO IT ALL IN HOME AUTOMATION $250 | |
| SKYLINE CONTROL | SOFTWARE UPGRADE FOR X-10 CP290P | |
| | EASIER TO EDIT USE DUSK AND DAWN CONTROL $49 95 OR $20 W/CP290P | |
| APPROACHING HOME AUTOMATION | BOOK ON X-10 $18.99 | |
| UNIVERSAL ELECTRONICS | ONE-FOR-ALL REMOTES FROM $11.24 | |
| SSI | DOLBY PRO LOGIC SURROUND SOUND DECODERS, AMPS & SPEAKERS | |
| AR POWERED PARTNERS | COMPUTER SELF POWERED SPEAKERS | |
| FOSGATE | HTS & IN-WALL SPEAKERS | |
| PANASONIC | HYBRID PHONE SYSTEM | |
| PANASONIC | PHONES, FAX & ANSWERING MACHINES | |
| MAGNAVOX | CLOSED CIRCUIT TV SYSTEM | $399 00 |
| RAINDRIP | LAWN GENIE | |
| PC TELE-VISION | WATCHTVON COMPUTER WITH WINDOWS | $449 00 |
| | CAPTURE VIDEO FROM TV, CAMERA OR VCR COMES WITH STEREO SPEAKERS | |
| | CHRISTMAS SPECIALS!! | |
| ES1400e | X-10 TWO-WAY COMPUTER INTERFACE | $319.00 |
| | WITH LATEST SOFTWARE ON SALE THIS MONTH | |
| CD1200 | INTERACTIVE CD PLAYER WITH COMPTON'S ENCYCLOPEDIA INCLUDED $299 VALUE | $499.00 |

CALL OR WRITE FOR CATALOG

DEALERS PLEASE WRITE ORFAX ON COMPANY LETTERHEAD

#107



DO YOU NEED CONTROL ?

**If you're looking for a temperature sensor that allows your computer to not only monitor the temperature but respond to it,**

***look no further.***

**Temp-A-Chip,, is a solid state temperature sensor - providing truly linear measurement of temperature. The Temp-A-Chip™ is an intelligent, user configurable sensor which interfaces with your computer. No batteries are needed to operate the Temp-A-Chip™, it plugs into any standard RS232 interface.**

Temp-A-Chip™

- LCD Display
- Solid State Design
- No Batteries Req'd
- RS-232 Interface
- Easy To Install
- Easy To Use

*Can You Afford Not to Call Today?*
$149.00 $5.00 P&H (800) 274-8699
*Volume Discounts Available* (606) 278-4701

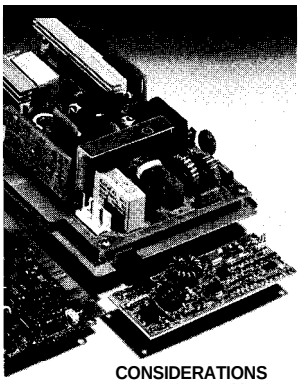
#108

# NEW PRODUCT NEWS

## POWER SUPPLY REFERENCE BOOK

A new 32-page book "Considerations When Specifying Switchmode Power Supplies" is available free of charge from MicroENERGY Corporation's Orlando Engineering Development Center.

This book is packed with useful information on topics such as reliability, agency approvals, EMC, power factor correction, strife testing, power density, and valuable appendices including major safety and quality agencies.



CONSIDERATIONS WHEN SPECIFYING A SWITCHMODE POWER SUPPLY

**MicroENERGY,** Inc.
745 W. State Rd. 434
Longwood, FL 32750-4909
(407) 831-2000
Fax: (407) 831-I 100

**#508**

## APPLICATIONS REFERENCE MANUAL

Analog Devices' 1993 Applications Reference Manual is a collection of technical articles, application notes, tutorial material, and design ideas reprinted from a wide variety of sources including trade press articles. This compendium greatly helps any engineer working on analog, mixed-signal, or DSP designs where real-world signals must interface to electronic systems. Topics range from fundamentals (e.g., noise and grounding) through advanced topics (e.g., multichannel systems).

The book is divided into 24 sections such as audio, communications, multiplexers, and switches. The 1993 Applications Reference Manual is available for $9.95.



Analog Devices, Inc.
181 Ballardvale St.
Wilmington, MA 01887
(617) 937-I 428
Fax: (617) 821-4273

**#509**


**µLAN**

**The 9-Bit Solution is the Answer to your Embedded Network Needs**

PC/XT/AT
8051
8096
80C186EB/EC
68HC11
68HC16
Z180

**The Cimetrics Technology 9-Bit Solution is a complete microcontroller network (µLAN) that supports the 8051, 68HC11, 80186, and many other popular processors. The 9-Bit Solution takes full advantage of multiprocessor modes built into microcontroller serial ports. Our flexible software and hardware allow developers to create powerful, yet inexpensive master/slave multidrop embedded controller networks.**

- Up to 250 nodes
- 16-bit CRC error checking with sequence numbers
- Low network overhead and low resource requirements
- RS-485 interface card for the PC
- Complete source code included
- Comprehensive documentation

CIMETRICS TECHNOLOGY
120 West State Street, Ithaca, NY 14850
TEL: (607) 273-5715 FAX: (607) 273-5712

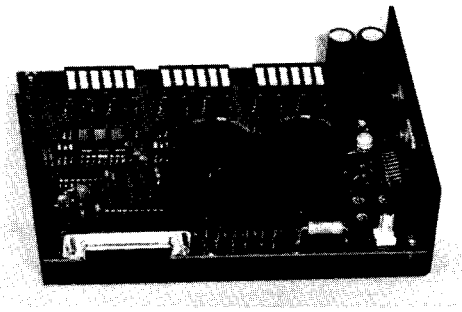
#109

**3-AXIS STEP MOTOR DRIVER**

**Buy our 6006-DB 3-Motor Drive for $165 & we'll throw in up to 3 motors* for $25ea**

*Drives 3 axes independently.
• 1.25 Amps per phase.
*Half-step, full-step and wave drive modes. Remote disable.
• 4 Ampere switching power supply. Adjustable from 5 - 28VDC.
*Clock and Direction inputs. Led's indicate motion and direction.
*Complete. Ready to plug-in and run. Includes all cables.
• *Includes software* for running from an *IBM* PC printer port.

• **60** oz-in. 1500 steps/sec, size 23, 1.8°/step.
*Offer ends December 31st,* 1993



**Ask for our FREE Catalog**

AMSI CORP.
American Scientific Instrument Corp
PO Box 651
Smithtown, NV 11787
(5 16) 36 I-9499 Tel
(516) 265-624 1 Fax

AMERICAN EXPRESS MasterCard VISA

#110

# FEATURES


14 Secrets of Using the DS1209 in an RF Transponder

22 The Covert Chordic Keyboard

30 Accessing The DS5000T Timekeeper from C

40 Designing With FPGAs


FEATURE ARTICLE


**Maurizio Ferrari**


# Secrets of Using the DS1209 in an RF Transponder

The DS1209 is a general-purpose wireless–to–3-wire interface that can be used in many applications. Explore its use in an RF transponder, and find out some tricks the data sheets don't tell you about the part.

Contactless identification has very large commercial appeal due to its enormous range of applications. Automated factory environments, livestock identification, security, and personnel management are just some fields where this idea has already been applied with success. And there are many more, huge, untapped markets. For example, think about replacing the paper tags on your flight luggage with an electronic transponder so conveyor belts would know where to route the baggage at the beginning and at the end of a trip.

With this kind of market potential, it's no wonder chip makers have developed a host of custom, and very specialized, devices. Some, like TI, have come up with extremely small, powerless tags for animal ID (TIRIS), but the resources required to design a fully customized IC are still far beyond the reach of most individuals.

Leave it to Dallas Semiconductor to come to the rescue. In this article, I will explore their DS1209, a low-power device ideal for use as the core of a transponder design. This transponder runs with a small battery, provides

read / write capabilities, and has an optional password for secured access.

Dallas Semiconductor has established itself as a major supplier of RAMified real-time clocks for PCs. Their catalog shows an impressive range of other niche devices. One point that makes Dallas very attractive to the small developer is their direct sale policy.

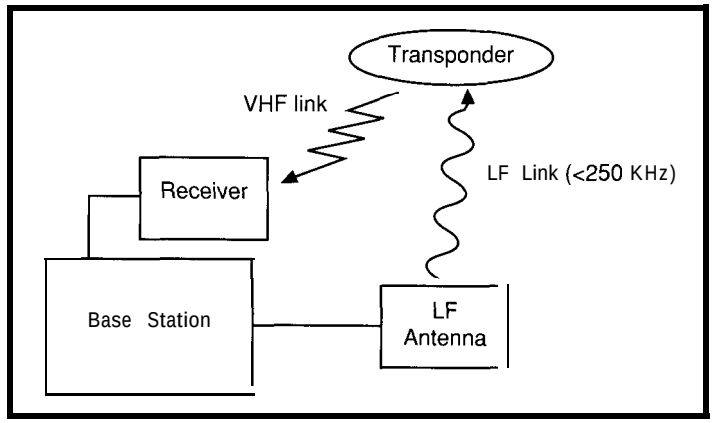



Figure 1—*A* basic *transponder system consists of a base station and receiver, a transponder, and an LF antenna.*

## THE SYSTEM

Why would we want a transponder? First of all, humans tend to be lazy. We generally forget things like arming alarms and sliding badges through readers. In fact, we forget to do things whose desired effect is to signal that we exist at a certain place at a certain time. Is there any other possible method we could use to signify our presence or our absence? A transponder system is one approach to solving this problem. A basic transponder system is shown in Figure 1.

The VHF link and receiver provide the means by which the transponder communicates its data to the central or reader station. What I'm describing here is a way to trigger such a transmission and not how to transport data: in fact, the DS1209 will toggle its TX pin according to the level read on its data line. The device is general purpose enough that it will operate regardless of what it is attached to or the transmission medium being used. You could use an infrared delivery system, but you are free to use any method as long as it can be driven by a single pin. For my purposes, imagine that a wire is pulled between the transponder and the base station. It could be a physical wire, but you would lose the advantage of a transponder! I will refer to it as a VHF link because it is the most likely solution.

The transponder consists of a loop antenna tied to the 1209, which is linked to a 3-wire device whose output is read back by the 1209 in order to key a VHF transmitter back to the base station. Dallas also manufactures special-purpose devices such as the 1205 and 1204. These only need to be wired straight to the CLK, DQ, RST\, and BATref pins to operate. Intended as secure chips, they feature password-protected access and are very expensive, so you might not want to use these unless your security requirements are strict. Also, if you really do require secured access, any 93C06 EEPROM can be used with a bit of effort instead, leading to substantial cost savings.

| Number of Pulses | | | Command word meaning |
|---|---|---|---|
| Min. | Typ. | Max. | |
| 5 | 20 | 29 | Write *0* or READ |
| 30 | 40 | 49 | Write 1 |
| 50 | 60 | 69 | Take RST\High |
| 70 | 80 | 89 | Return to Inactive State |
| 90 | 100 | 109 | Initialize DS1209 |

Figure 2—*DS1209 command words are coded as* varying numbers of pulses. Note the *"center"* value in the *allowable range.*

## THE DS1209S-B1

Unfortunately, the data sheet will not tell you some facts that turn out to be vital in coaxing the best performance out of the part. I have uncovered some of them in my experiments and will pass them on to you here.

The device converts a wireless protocol into a 3-wire serial link. The circuits in this interface are clock, data, and select. It is also Microwire (or, if you like Motorola jargon better, SPI) compatible. Stated another way, this interface gives the user the ability to drive a chip select pin and write to, or read from, a data line with a synchronous clock. It listens for incoming data continuously by monitoring for signal level changes at its A- and A+ inputs, and can receive frequencies up to 250 kHz. Typically, it receives a pulse train from a small loop antenna tuned to a designated frequency. The antenna outputs are tied to the A+ and A- pins of the device. The 1209 counts the number of pulses it receives via zero-crossing detection, interprets the message, and acts on the 3-wire bus. The LF inputs will wake up with a signal as small as 25 mV peak-to-peak, and the device can be addressed with any unique code out of 65,536. This "radio address" can be stored in internal RAM and can be changed at will or be permanently written.

The device recognizes pulse packets at a frequency up to 250 kHz with the protocol described in Figure 2. You can see that every command word has a "center" number and an allowable range. In fact, the pulses are transmitted by a transmit base coil and driven by a square wave that may be generated by a microcontroller. Obviously, the magnetic field will exhibit both a

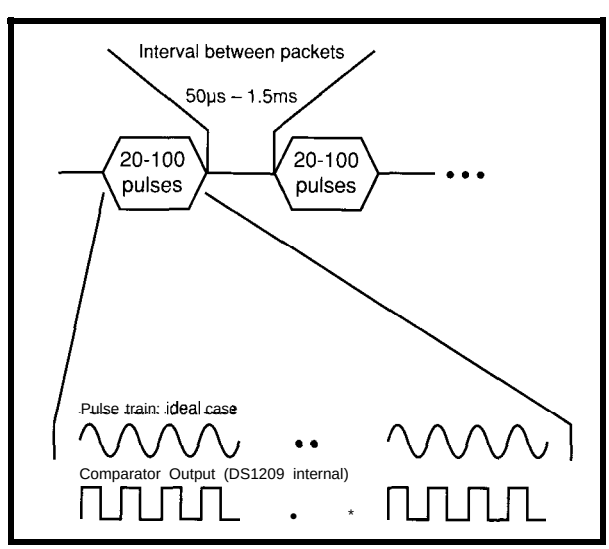



Figure *3-Each 20–100-pulse packet represents a command word. The interval allows for discriminating between packets.*

AMX™
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 V6J 1Y5

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

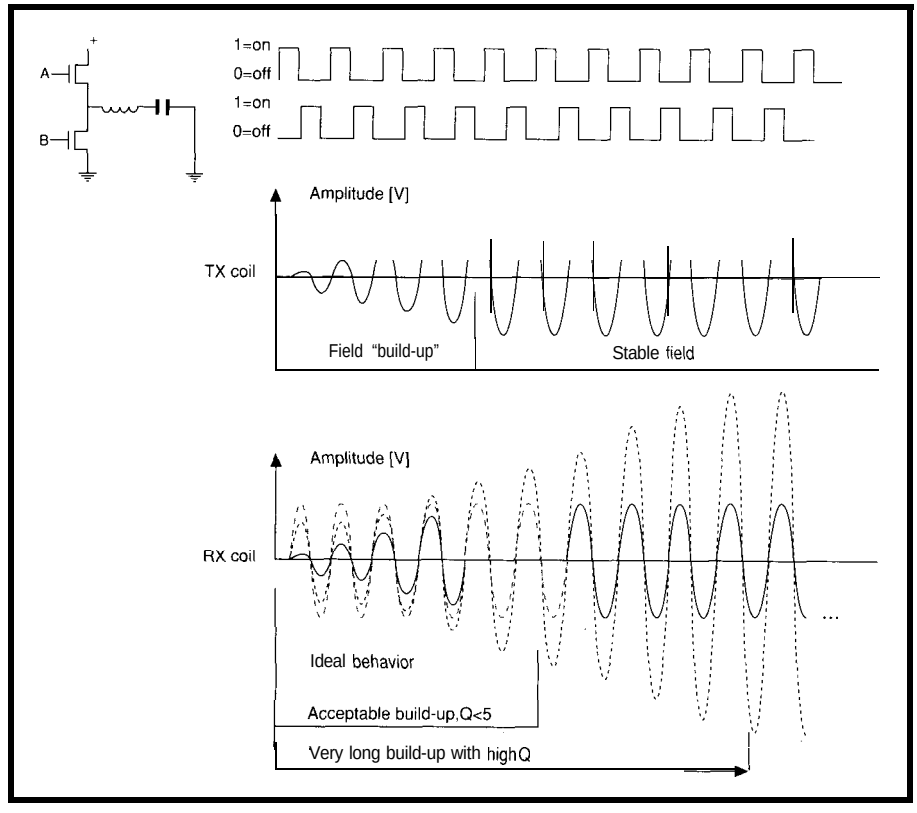



**Figure 4—** *The effects of the coil's Q on the field amplitude growth: ideally, it should be in a stable state from the first pulse (i.e., no build-up time).*

build-up time before reaching stable state and a decay at the end of transmission. The behavior of the field's expansion or decay is influenced by various factors, the most important being the way the wave is stopped on the transmitting antenna and the Q of both the receive and transmit coil. The device's effective range is also deeply related to these parameters.

The first notable design issue is that, at a close distance, the receive and transmit coils are strongly coupled. As the transmit field decays, energy is coupled to the receive antenna. This raises the energy level at the A+ and A- pins of the device above the 25-mV threshold. This creates the situation where the decay can be interpreted as valid pulses and leads to two different anomalies:

*Insufficient dead time (less than 50 µs, see Figure 3) between commands. This condition creates pulse trains that "merge" in a continuous stream. Remember that the device's receiver wakes up at 25 mV, and it triggers by recognizing a zero crossing rather than relative amplitude. In contrast with the 1991 data sheet, the 1209 does not have an internal Automatic Gain Control and its input gates are zero-cross triggered. In fact, the latest specs (1992/1993) do not even mention the AGC.

*Misunderstanding of command words. This occurs when the number of received pulses becomes greater than the ones generated by the microcontroller, thus crossing the limits in Figure 2.

These effects can be overcome by three combined actions:

- Keep the Q low to minimize build-up time and decay. This has the side effect of decreasing sensitivity (i.e., range), but will make the system much more reliable. In my experience, the small receive coil should have a Q that is less than five. See Figure 4.

*Design the driving bridge with logic-level MOSFETs and turn them both off when the packet is sent. This dumps the transmit coil in the shortest time, as shown Figure 5. Leaving one of the drivers on will keep the transmitting antenna oscillating for a

longer period, scrambling the protocol, as explained in Figure 6. (Keeping them both on will cause the magic smoke to come out of the circuit. And as every engineer knows, when the blue puff has escaped from its plastic case, somehow the device ceases to work. This is sure evidence that electronic circuits are based on magic smoke. QED)

*Wait long enough between packets to give the receive coil enough time to settle down for 50 µs between packets.

At longer, end-of-scale distances, the build-up time prevents valid pulses from being recognized, which serves to decrease the effective range. Somehow, this is generally a smaller problem than the previous one. Figure 5 shows a decent compromise based on minimizing the impacts of these two competing design problems.

At this point, we have tuned our antennas, timing circuits, and microcontrollers thinking we might be through with the transmission issues.

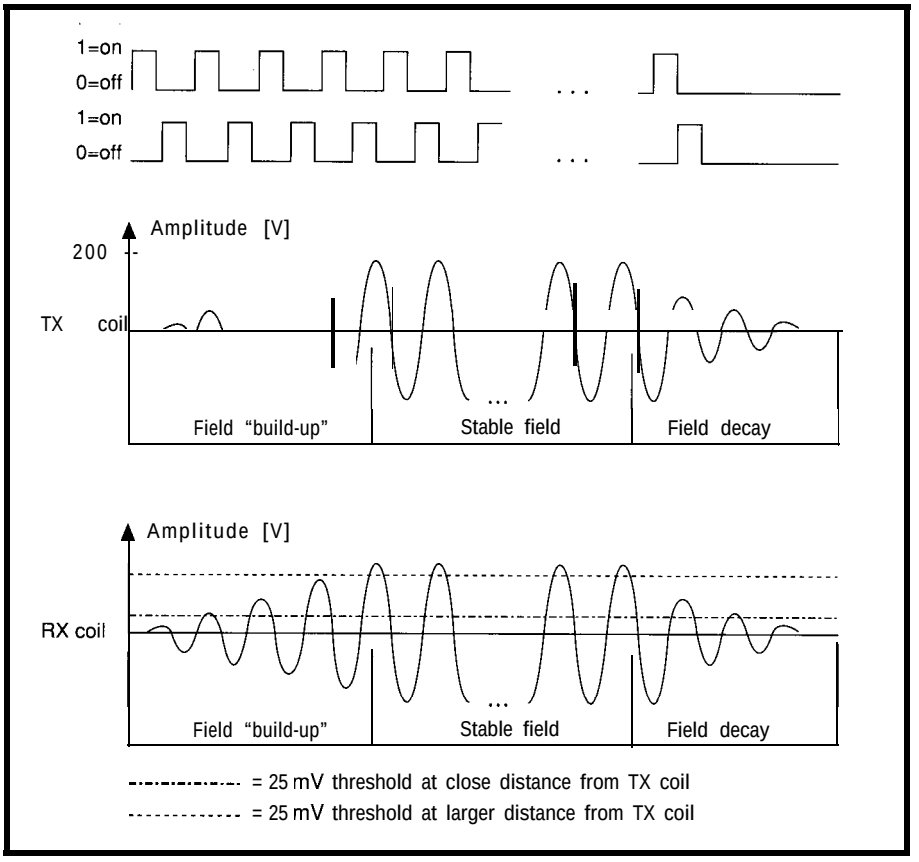
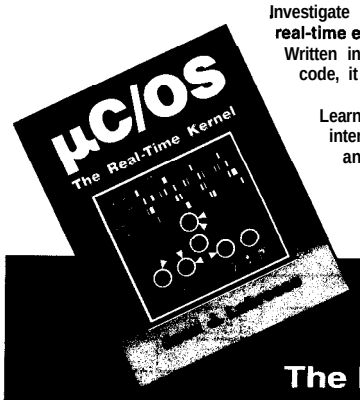



Figure 5—*Turning the drivers off dumps the oscillation in the shortest time.*


Push the Limits of Real-time Design!




Investigate the fundamentals of building real-time embedded kernels with µC/OS. Written in C with minimum assembly code, it is portable and ROM able.

Learn about task priority scheduling, intertask communication, interrupts, and performance benchmarking.

µC/OS is complete preemptive multitasking operating system that handles 63 tasks.

µC/OS
The Real Time Kernel
by Jean J. Labrosse

Secrets of Embedded Systems Revealed!

- Performance compares to commercial kernels
- Written in C with assembly code minimized
- Assembly code minimized for easy portability
- Includes System Code & Users Manual

Companion Disk for $24.95

Only $29.95 plus shipping

Order Today!

(order book W60, book & disk W62)

913-841-I 631 (ext. 62)

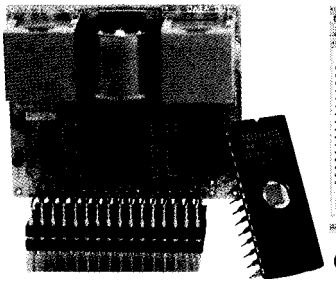
FAX 913-841-2624

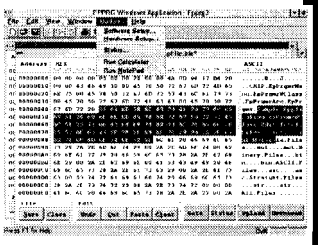
VISA MasterCard

R&D publications, inc.

#112

PROMJet™ Compact EPROM Emulator





Optional windows software with powerful binary editor.

- Emulates 64 Kbit to 8 Mbit 85/40ns EPROMs.
- Accepts Binary, Ext. Intel & Motorola formats.
- Fast download from printer port (1 Mbit/Sec).
- Fits into EPROM socket to minimize noises.
- 4 Layer double sided SMT (2.2"x0.7"x1.9").
- Jumperless configuration through software.
- Ni-Cd battery backup. Power-up emulation.
- Cascadable to 128 bits. Generates RESET+/-.
- Comes complete with software and cables.

15 DAY MONEY-BACK GUARANTEE.

| | | |
|---|---|---|
| PROMJet 64K-1M/2M | 85ns | $2451295 |
| PROMJet 64K-4M/8M | 85ns | $4951695 |

WESTEC Research Corp. Tel:(213)664-8909 Fax:243 1

2750 Riverside Dr. #205, LA, CA 90039

#113

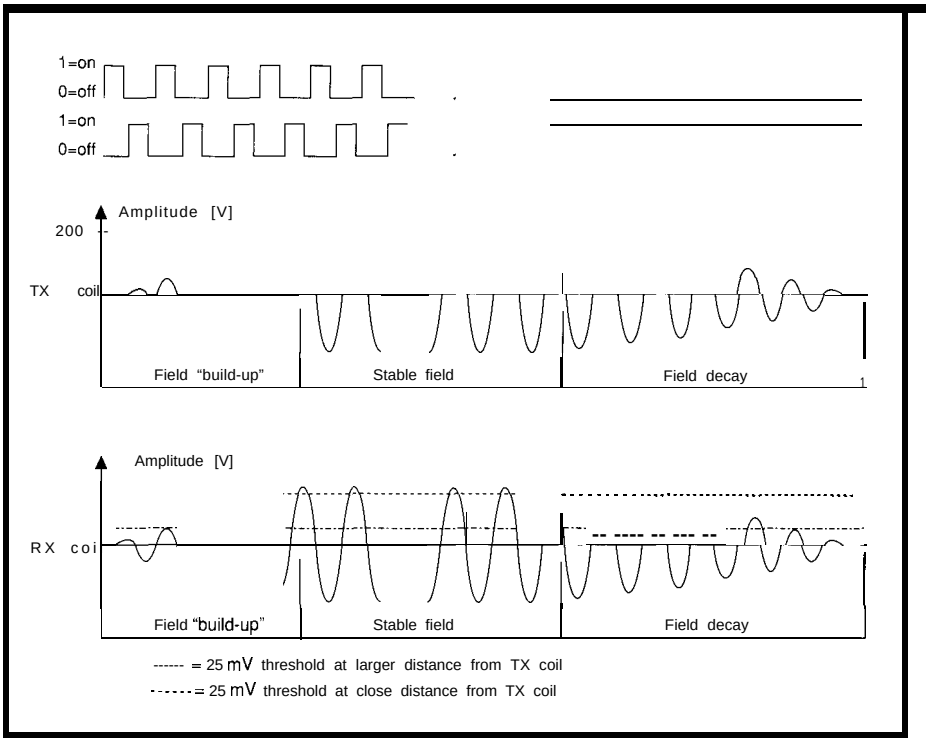



Figure 6—*Leaving a driver on worstens field decay: ideally, if should drop to zero at once, instead.*

Not yet, though! Two extra quirks must be addressed: power consumption and link directionality.

The DS1209-B1 data sheets state a standby current of 2 µA max. I've checked several devices from a batch of 900 and I've seen it drain from 1.5 µA to 5 µA with an average of 3 µA. This is probably not a problem for most applications. However, there is a more important issue. Recall that the device has a wake-up threshold of 25 mV peak-to-peak. This means any magnetic fields up to 250300 kHz strong enough to wake up the device will keep its internal circuits switching at that field's frequency. This will cause the battery drain to go up to 50 µA for any incoming frequency of approximately 300 kHz. Also, the zero-crossing detection circuit, for the duration of this condition, will never be able to recognize any intelligence the base is attempting to send. Common sources of this kind of interference are computer monitors and televisions.

You will probably need to experiment to discover a "safe" distance. During my tests, I have seen the device stay awake when it is 30 cm from VGA screens. I have no experience with what would happen in the proximity of strong LF transmitters, like LORAN base stations, but I suspect there may be trouble.

## DIRECTIONAL, LIKE INFRARED...

At this point, we have a radio device that is direction sensitive. This is because the magnetic coupling between the transmit and receive coils varies with the areas that they see of each other. This in turn is a function of $\cos(\varphi)$, where $\varphi$ is the relative angle between the coils.

I think the laws of physics guarantee there are only two ways to cope with the problem. You can accept it or you can have extra coils with different orientations at different times linking to the receive coil.

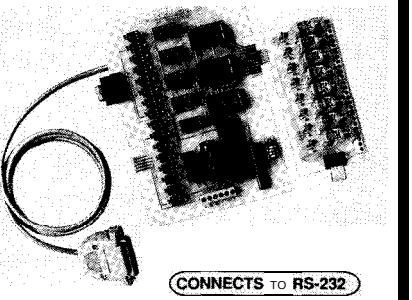
However, if you're planning to write to the device as well as read from it, you must realize that moving it while writing may interrupt the communication, causing data loss. There is a critical angle depending on the system's characteristics and operating distance above which the device will not be able to listen to the transmitting antenna. The amplitude is varied by all the factors that affect the design.

## THE EEPROM INTERFACE

The communication issues up to this point will apply to any DS1209-


RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

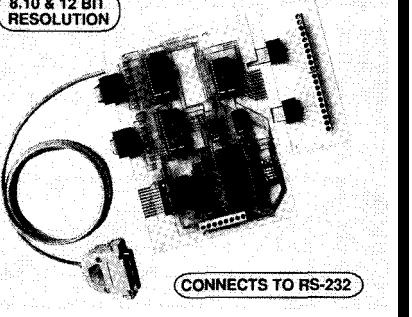


CONNECTS TO RS-232

AR-16 RELAY INTERFACE ..... $ 89.
Two 8 channel relay output ports are provided for control up to 16 relays (expandable to 128 relays using EX-16 expansion cards). Each relay output port connects to a relay card or terminal block. A variety of relay cards and relays are stocked. Call for mom info. RS-422 available (distances to 4,000 feet). PS-4 port selector may be used to control satellite AR-16 interfaces. (up to 16,384 relays)
RD-8 REED RELAY CARD (8 relays, 10 VA) ..... $ 49.
RH-8 RELAY CARD 10 amp SPDT 277 VAC) ..... $ 69.
LX-16 RELAY EXPANSION CARD (16 channel)... $ 59.

ANALOG TO DIGITAL

8,10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 A/D CONVERTER (16 Channel, 8 bit)..$ 99.
Input temperature, voltage amperage. pressure,energy usage, energy demand, light levels, joystick movement a a wide variety of other types of analog signals. Inputs ma be expanded to 32 analog or 128 status inputs using the AD-16 or ST-32 expansion cards. 112 relays may be controlled using EX-16 expansion cards. Analog inputs may be configured for temperature input using the TE-8 temperature input conversion. RS-422 available. PS-4 po selector may be used to connect satellite ADC-16 interfaces (up to 4,096 analog inputs/16,384 status inputs and 14,336 relays). Call for info on 10 & 12 bit converters (terminal Mock end cable sold separately)
ST-32 STATUS EXPANSION CARD ..... $79.
Input on/off statue of relays, switches, HVAC equipment, thermostats, security devices, smoke detectors and other devices including keypads and binary coded outputs. Provides 32 status inputs (opto isolators sold separately).
TE-8 TEMPERATURE INPUT CONVERSION ..... $ 49.
Includes 8 temperature sensors & terminal block. Temperature range is minus 48 to 145 degrees F.
PS-4 PORT SELECTOR (4 channels RS-422) ..... $ 79.
Converts an RS-232 port into 4 selectable RS-422 ports.
TOUCH TONE DECODER and other serial interfacing products available. Call or free information packet.

- FULL TECHNICAL SUPPORT...Provided Over the telephone by our staff. EACH ORDER INCLUDES A FREE DISK WITH PROGRAMMING EXAMPLES IF BASIC, C AND ASSEMBLY LANGUAGE. A detailed technical reference manual is also included
- HIGH RELIABILITY...engineered for continuous 24 hour industrial applications. All ICs socketed.
- Use with IBM and compatibles, Tandy, Apple Mac and most other computers with RS-232 or RS-422 ports. All standard baud rates and protocols may be used (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 8 orders

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

#114

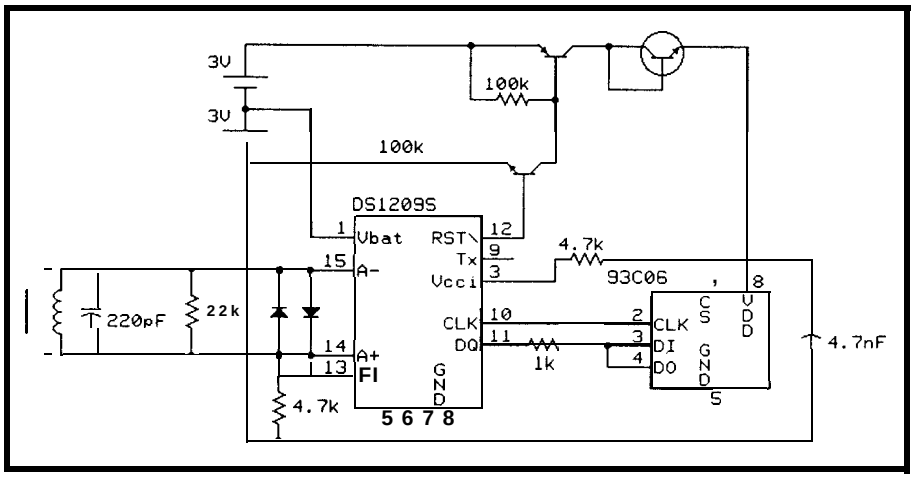



**Figure 7**—*Some tricks in* **the** *power section of the circuit are necessary to mix the DS1209, which can operate on 3 volts, and the 93C06, which requires at least 5 volts.*

based transponder, no matter what device you want to interface with it. Having overcome them, it's now possible to read and write a very cheap device like the 93C06 EEPROM.

Before we do that, we must take into account a very strict timing specification that the DS1209 imposes. The data sheets specify a 500-ns minimum time for the low state of the clock line. The DS 1204 and DS 1205 are designed to interface directly to the 1209, but as I said before, an EEPROM is much less costly and is perfectly satisfactory for most of the applications one may think of. We also want to design a portable device that will last as long as possible without requiring new batteries. At the time I developed the project, no EEPROMs were available that could satisfy all the ideal requirements, those being:

- Fast clock to comply with the 500-ns specification
- Zero power drain when not in use
- 2.5-V operation to meet the lowest allowable $V_{bat}$ of the DS 1209

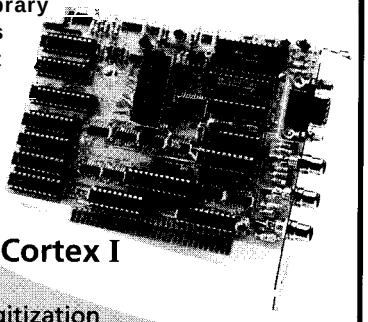
New-generation EEPROMs comply with the first two requirements, but the 93C06 has an $I_{stdby}$ of 200 µA, and its 3-V parts are not fast enough. The choice is to keep the DS1209 in standby at 3 V and the 93C06 off, then use the RST line to switch a second battery in to jump the supply up to 6 V (a diode drop was inserted to limit $V_{dd}$ to not more than 5.5 V) while turning the 93C06 on. As the RST line also drives the 93C06 chip select, an additional RC network must be inserted so the EEPROM sees its Chip Select go high after it wakes up (Figure 7).

The circuit works as follows: When you issue the RSTHIGH command to the DS1209, it will pull its RST line to 3 V. This will switch the auxiliary battery in through the transistors, raising the supply voltage on the DS 1209 to approximately 5.3 V. Now, the RST\ pin and the 93C06 $V_{dd}$ will reach the same level, and after the delay imposed by the RC network, the memory will see its Chip Select line go to logic one as well. This is the first step to establish communication with the remote device. Now we can issue all the command and data words that we need using the DS1209 Write and Read command words.

The simple Command Words shown in Figure 2 can be combined to obtain information from the 93C06. Next, we need to define a method that can be used as a design for writing the real code. As an example, a simple EEPROM readout of the data at a certain EEPROM location will be carried out through the following steps:

*Initialize the DS1209 state machine (100 pulses).

*Wake up the device while masking all the chip select bits (the code for this is 00011000). Alternatively, a DS1209 can be programmed to store a 16-bit value in its standby RAM. A wake-up message can be directed to a specific device using a full 16-bit address or a group of devices using a


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
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
#115

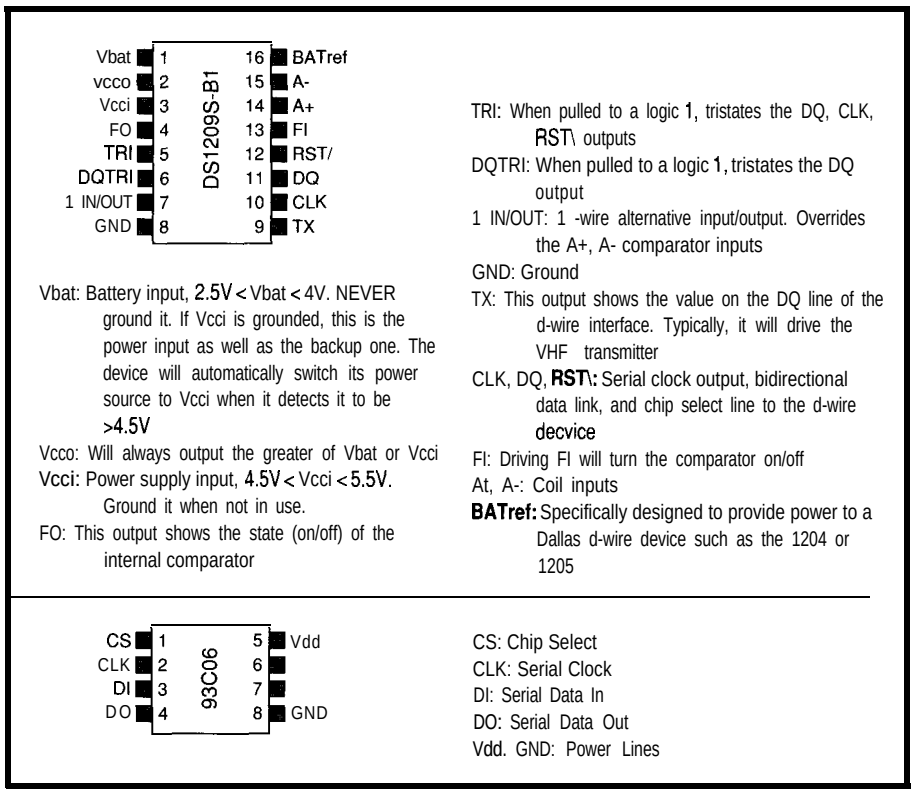



Figure 8—*The DS1209 wireless–to–3-wire chip and the 93C06 serial EEPROM can be combined to make an inexpensive RF transponder with security.*

subset address. Obviously, the DS1209-specific address must be programmed into the device before using this feature. There is a specific procedure in the data sheet for this operation. I suggest you master the basics by writing the code for the standard wake-up mode first. Before long, 16-bit addressing will become a straightforward matter as your experience grows.

*Transmit a dummy address or transmit a real address for selective wake-up.

*Check for the existence of a DS1209 in the proximity using the "beacon" feature. To do this, send 80 pulses, followed by 20, then by 40. If there is any device within range, it will start modulating its transmit pin with a 1.6-kHz square wave.

*Stop the beacon by issuing 60 pulses.

•The EEPROM readout can now begin. The 93C06 protocol needs a trailing 01, so the first two commands should be constructed to provide that sequence.

•Next, you need to transmit the EEPROM address to be read (MSB first). In other words, to specify the address of interest, you need to output a sequence of bits that represents the address. The first bit sent is the MSB of the address followed by the LSB.

● The DS1209 will now reply at every read command by raising its transmit pin if the value read from the 93C06 is a one.

•To complete the read of this address, your code will need to test the receiver in the interval between the read commands to determine if the DS1209 has read a one or a zero. The output timing is described in Figure 9.

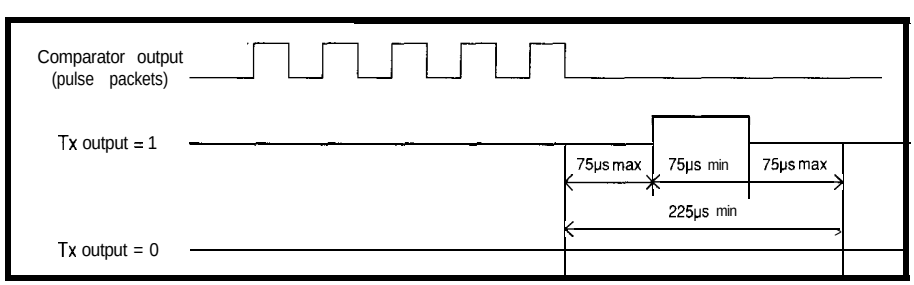



Figure 9—*To complete the read of an address in the 93C06, the code needs to test the receiver in the interval between the read commands to determine if the DS1209 has read a one or a zero.*

## CONCLUSIONS

I have shown you one way to apply the DS1209 in some of your designs through this illustration of a simple transponder circuit. As I have pointed out, this project is a matter of compromise. If you want to use the longest possible range, you must be ready to sacrifice shortest distance recognition, and vice versa.

Power consumption is low enough to guarantee it will work for two years when powered by a 2032 cell as long as you are not in the habit of leaving your device close to televisions or computer monitors. Its internal RAM code can be used to a good advantage to carry identification codes of all types.

Another possible application for this device is to interface it with a microcontroller through a synchronous serial port. This would allow cheap radio transmission of data, albeit at very slow rates, for certain kinds of niche products. Adding a 1204 or 1205 to the 1209 enables you to build a secure transponder for access control and limited data carriage. It's a matter of taste. The device is a bit quirky, but flexible enough to be used in a wide variety of situations. ❑

***Maurizio Ferrari graduated with an M.S. in Electrical Engineering from Bologna University (Alma Mater Studiorum, oldest University in the world) in Italy. He is a software engineer for Magneti Marelli.***

## CONTACT

Dallas Semiconductor Corp.
4401 S. Beltwood Pkwy.
Dallas, TX 75244-3292
(214) 450-0448
Fax: (214) 450-0470

401 Very Useful
402 Moderately Useful
403 Not Useful

# The Covert Chordic Keyboard

FEATURE ARTICLE

Scot Colburn

How long would it take you to type this? And how many fingers do you use to type? One finger or not, you don't have to be a musician to appreciate the benefits of a keyboard that fits in your palm.

The covert chordic keyboard is a small, one-handed, 2-line by 40-character, LCD-based text editor. Why is it covert? The entire machine can be hidden inside a jacket. The idea to have a small, discrete text entry device came to me while using an NEC PC-8201 (the Japanese cousin of the TRS-80 Model 100) to take notes in a FORTRAN class. I was getting stared at, and the "tap tap" sound of the keys was decibels above the soft "scritch scritch" of people writing notes. Also, I took the bus a lot in those days and often wanted to use the time to write, but bus-written notes are barely legible, and they need to be transcribed to the computer later anyway, so what's the point? But a small one-handed terminal could be used to write if I were on the bus, in class, or even while walking home.

The device uses a five-button keyboard that is held in the palm of one hand-each finger poised over its own button. Characters are formed by ***chording,*** which is defined as pushing several buttons at once. Text is displayed on a 40-character by 2-line LCD display. Lines longer than 240 characters can be edited by inserting, overwriting, moving backward or forward, and scrolling the tiny screen horizontally. Only 31 different chords are available, so escape sequences and chord multiples are used to give commands, enter less-common characters, move to different lines, or toggle between upper and lower case.

My original plan was to use some sort of tactile feedback to "read" the text. But when a friend offered me this tiny 80-character LCD, I just knew that was the way to go. This small stick of a display can be attached to the forearm with elastic, mounted to the hand-held portion of the terminal, or just laid on a desktop. With a backlight, this display can be read nearly anywhere, just like a watch.

## THE ALPHABET

The ***chordic alphabet*** is designed to be as ergonomic, efficient, and easy to use as possible. Like Morse code, the most frequent letters are the easiest to chord. The four most common letters, "E," "T," "A," and "0," use one-finger chords (see Figure 1). The space character is more common than any of these, so it uses the fifth one-finger chord. I consider chords using groups of adjacent fingers to be easier to form than separated finger chords, so the letters "N," "H," "S," and "I" (which are members of the group consisting of the next most common letters) are formed using adjacent fingers.

Things get trickier when deciding precisely which chords go with which letters. "E" is most often followed by a space, for example, so the buttons in their chords are placed next to each other. "THE" is one of the most common triplets in English, so the chords for those three letters are designed to naturally and easily follow each other. "RE," "ER," "ON," "TO," and "IS" are other common doublets that are easy to chord. Of course, many other doublets occur that aren't so easily resolved. "IN," "AN," or "AT," for example, aren't adjacent in the current chording scheme. After several days of juggling letter, doublet, and triplet frequencies, I arrived at what you see here. Perhaps a computer could balance all these factors nicely. I'm sure Samuel Morse would have done a better job.

Some letters are so uncommon that two of them share one chord. As any Scrabble player will tell you, Q, J, X, and Z are the highest scoring letters because they are so difficult to use (each one is a lo-point letter). On the chordic keyboard, the letters "Q" and "X" share the same chord. If the chord for "Q" is made twice, then the first

"Q" is erased and an "X" pops up. The same goes for "J." Two "J" chords make one "Z." The only way to chord a "QQ" sequence is to chord one "Q," then any character besides "Q," then backup to erase that letter, and then form the second "Q." I couldn't find "QQ" in Funk & Wagnalls and I believe J.J. Mcabe's has periods in it, so neither "JJ" nor "QQ" should pose a problem. The comma and the period each have their own chord, since they share frequencies with B and W. But all other punctuation is bundled under one punctuation chord. After the comma and the period, the most common punctuation mark is the apostrophe ('). Repeating the punctuation chord erases the last punctuation mark and replaces it with the next most frequent punctuation mark. Fourteen punctuation marks are revealed by successive repeats of the punctuation chord.

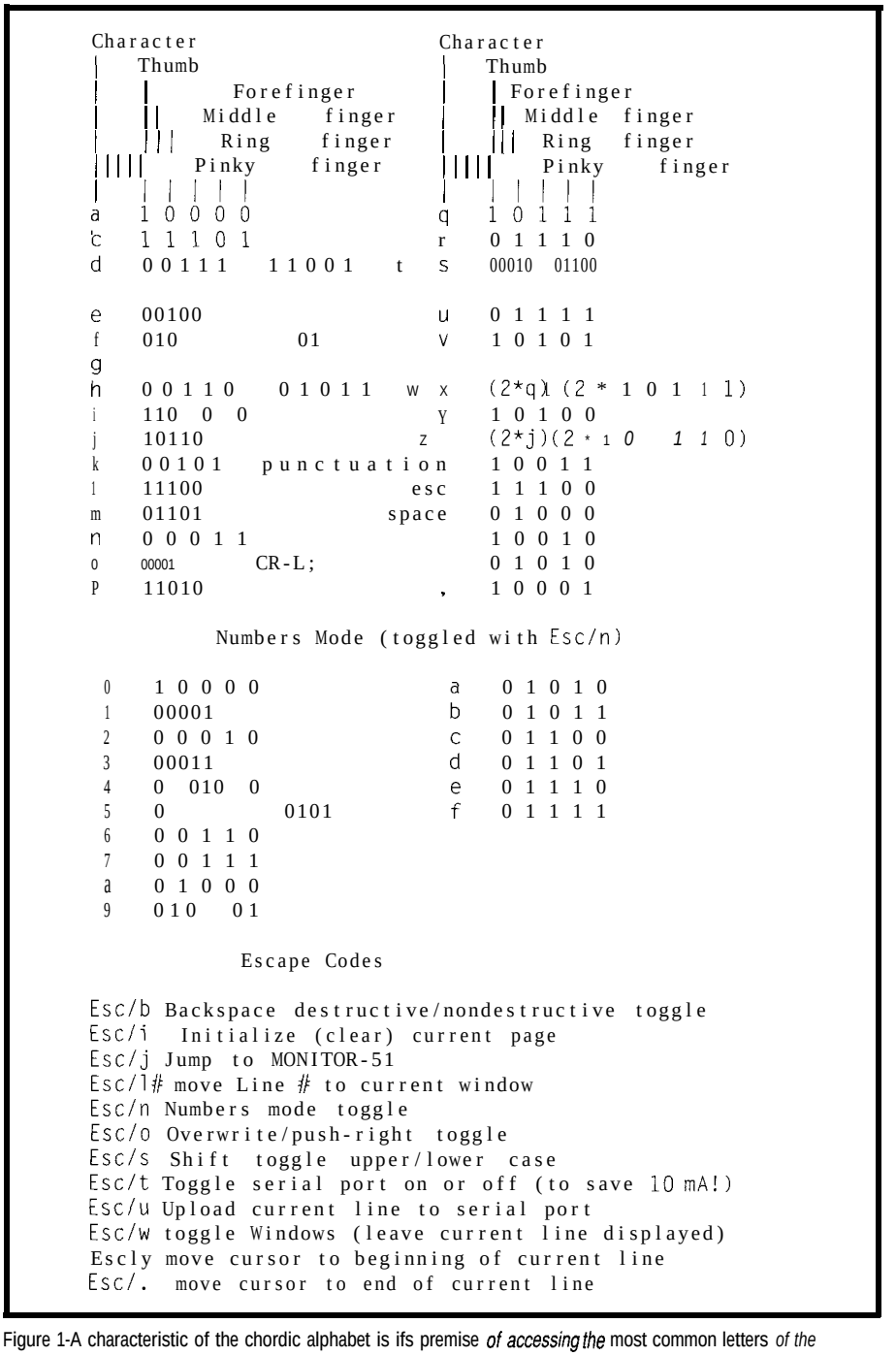



Figure 1-A characteristic of the chordic alphabet is ifs premise *of accessing the* most common letters *of the alphabet. The table above is a summary of the finger chord combinations for the alphabet as well as numbers and frequently used punctuation marks.*

The numbers are easier yet, at least for a computer nerd. Once "Numbers Mode" is toggled with an Esc/N, the numbers themselves are binary, except for zero, which is just a thumb-press. Cursor moves found here are identical to an IBM PC keyboard.

Learning the chordic alphabet is easier than one would think. Once a few sequences are committed to memory, remembering the rest of the alphabet is simple. Learn the words "IS" and "THE," for example, and you're a fifth of the way there. Remembering 3 1 different character chords is much easier than remembering, say, 31 flavors of ice cream.

## MUSIC LESSONS: THE CHORDS

Entering a chord is a bit different than typing. A typewriter is sensitive to key presses, whereas the chordic keyboard is sensitive to key releases. A chord is entered when a button in that chord is released. If the chordic keyboard listened to button pushes, then a "W" might become "TNGW" which is a 4-character string of three decreasingly common characters with a "W" tagged on at the end.

The chordic keyboard enters a character by watching the buttons being pushed and waiting for a button release. When a release occurs, it knows that the user is moving to a different chord, so it records the last combination of pressed keys as the previously impressed chord. Now the keyboard is watched for another key press; subsequent key releases are all ignored until a key press signals that the user is beginning to chord another character. The state diagram in Figure 2 describing this algorithm is perhaps easier to understand.

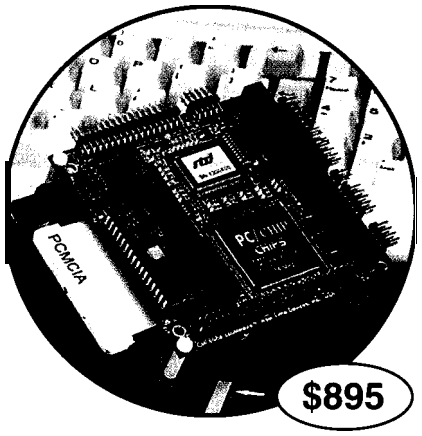
Moving from one chord to another doesn't necessitate releasing all the buttons and then pressing a new chord onto the keyboard. For example, The word "IS" can be spelled by making an "I" chord, releasing only the thumb to enter the "I," then pressing with the middle finger to form an "S," then releasing either finger to enter the "S."

## THE LINE EDITOR

The chordic keyboard can store and edit 31 "lines" of text. Each line

# 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 W-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/l 04 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!***

rtd®

**Real Time Devices, Inc.**
**P.O.** Box ***906***
State College, PA 16804
(814) 234-8087 ■ Fax: (814) 234-5218

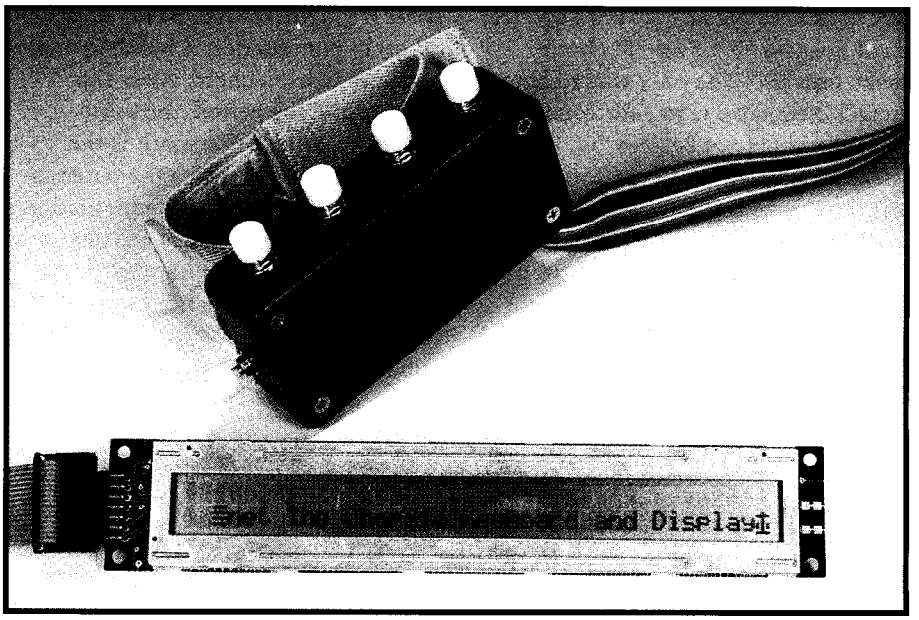



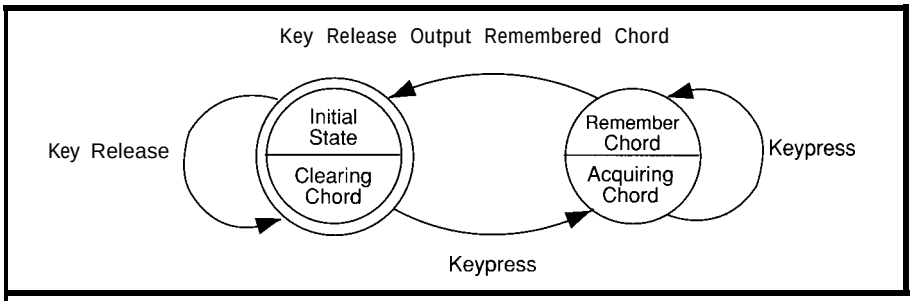
Photo I--The *Chordic* Keyboard *is made up of an off-the-shelf box, five buttons, and a ribbon cab/e that goes to the controller. A Velcro strap holds the keyboard in your palm. A P-line by 40-character LCD display offers a compact window onto already-typed text to allow reviewing and editing.*

can contain up to 240 characters. Lines are displayed on one of the LCD display lines, so two lines may be viewed at once. The Window command (Esc/W) will pop the cursor between the two lines displayed on the LCD. Chorded characters appear at the cursor, just like on a word processor. The Overwrite (Esc/O) toggle changes between an insert-by-pushing-right cursor (an underline cursor) and a write-over cursor (blinking block). If the Insert cursor is active when a line becomes full, the Overwrite cursor automatically appears. The backspace chord (all five buttons held down) can be toggled between destructive and nondestructive modes using the Backspace toggle command (Esc/B). Two escape codes, Esc/Y and Esc/. (escape period), move the cursor to the beginning or the end of the next line, respectively.

The 3 1 lines are instantly accessible with the Line command (Esc/L). Following the line command with any of the 3 1 available chords makes that line instantly pop up on the current window (top or bottom) of the display. These lines cannot be deleted, but they can be erased via the Init command.

## THE MACHINE

The machine is based on Iota Systems' EC-32 80C32 board. Up to 92K of memory can be placed on the board. The upper 32K is mapped to both program and data space, while the lower 32K is mapped in the standard 805 1 Harvard configuration, doubled so that data accesses come from RAM and code accesses from EPROM. All the RAM is battery backed with an on-board lithium battery. I/O devices live in the top 4K, so the top half of the memory is limited to 28K. It should be




**gure 2—***I he primary action of the Chord/c Keyboard can be described with a two-node state diagram. Characters are entered upon the first key release rather than a key press as with traditional keyboards*

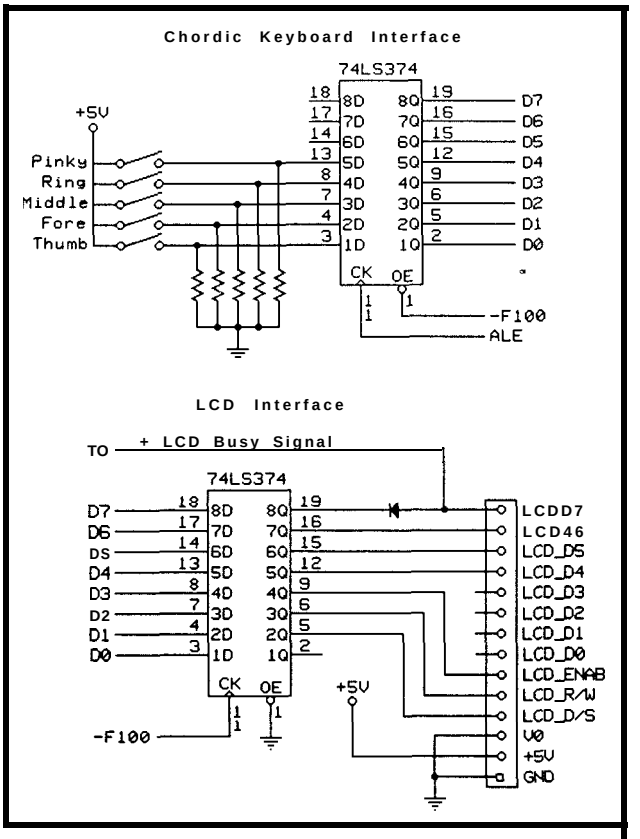



Figure 3—*The keyboard consists of nothing more than some push buttons, pull-down resistors, and a latch. The LCD interface is just another latch.*

noted that this processor platform was my choice for the project. Any other processor platform could be used, however, as long as it has the proper number of I/O lines available and can support the same level of I/O and bit-level operations that I used.

The hardware side of this project is simple (refer to Figure 3). Only two latches and a diode are needed on the board. One latch gets the keyboard data, and the other latch, in conjunction with the diode, handles output to the LCD. The chordic keyboard is housed in one of those ubiquitous blue boxes from Radio Shack, which I chose because it fit well in my hand. Five tactile feedback push buttons are arrayed around the outside of the box. Inside, five 10k resistors pull down the data lines and a ribbon cable carries all the signals to the latch on the board.

The software side is a bit more involved. When I bought this board I bet the same friend who lent me the LCD that an 8051 stacked up easily in processing power to, perhaps, a 6502. He nodded his head wisely and said, "You'll see... " Believe me, I have seen. The 8051 is a bit of a pain to program, but the bit-level operations are a pleasure, and easy access to the internal RAM makes up for the sometimes difficult to access registers.

The low-level interface code for the LCD (as well as the latch-and-diode interface) come nearly directly from Iota Systems' Application Note #1. This code initializes the LCD to the nybble-wide protocol and provides routines for getting data into the LCD.

The chordic and serial input routines are interrupt driven, which means the processor can enter a low-power idle between keyscans and characters. The serial interrupt routines can be ring-buffered in internal RAM, but I haven't found buffering to be necessary; I can't type more than 20 WPM on the chordic keyboard yet. In fact, the monitor program I'm using is incompatible with interrupt-driven serial output, so to make debugging easier I have removed the buffering code until it appears necessary.

Each line of text is stored in a page of memory so accesses to the external RAM can be addressed using the RO or R1 registers rather than DPTR. Each line is separate and no method of transferring text from one line to another is implemented, although it could be. When a line is exited, its cursor position is stored, so text entry can be started again exactly where it left off. If the buffer fills up

Listing 1—*Key scanning is based on a timer interrupt, which also serves to debounce the button presses. The main process simply waits for a semaphore from the interrupt service routine.*

```
;MainLoop sleeps until semaphore from serial
;or chordic systems wakes it.
MainLoop
      jbc   KeyReady,MProcChar        ; Does KeyScan have key ready?
      jbc   RI,MDRecieve              ; Does SBUF have a character?
      setb  IDL                       ; stop clock til next int
      sjmp  MainLoop                  ; Do it again
MProcChar
      acall ProcessChar
      sjmp  MainLoop
MDRecieve
      acall DumbRecieve
      sjmp  MainLoop
;KeyScan gets called on every Timer 0 overflow
;It reads the keyport, debounces 'em, and
;makes the chord-repeat work.
KeyScan
      push  ACC
      push  DPH
      push  DPL
      mov   dptr,#keys
      movx  A,@dptr                   ; get the keycode
      jb    NewKey,KeyTest            ;Keycode being validated?
      xrl   A,KeyHolder               ; Same as the last key?
      jz    KeyDecHold                ; Dec HoldCounter and exit
KeyNew
      setb  NewKey                    ; nope. this is something new!
      mov   HoldCounter,#0            ; reset repeat chord
      movx  A,@dptr                   ; get it again
      mov   CandKeyHolder,A           ; save it away
      mov   KeyVerCounts,#VerLoops    ; reset verify counter
      sjmp  KExit
KeyDecHold
```

*(continued)*

during a Push-Right-And-Insert mode, the mode is automatically changed to Write Over. A funky Japanese character (resident in the funky Japanese LCD) marks the end of a line. Moving between lines isn't as disconcerting as it might be because of the way I preserve the cursor position. Since the cursor position is saved, lines look the same upon entry as they did when they were left.

Text can be entered either from the chordic keyboard or the RS-232 port, though mode-change commands are limited to the chordic keyboard.

## CHORD ACQUISITION

The heart of the Chordic Keyboard is the procedure that implements the simple state diagram shown in Figure 1. The chord acquisition code is complicated by its mostly asleep nature. The Timer 0 interrupt sets off a brief flurry of activity that terminates with the setting of the IDLE bit and a wait for the next interrupt. In between interrupts, the state of the machine is stored in several bit and byte variables.

Listing 1 details the `Main Loop`, `KeyScan`, `KeyTest`, and `KeyChange` procedures. The `Main Loop` procedure dispatches control to the serial or chordic character handlers, then sleeps until the next interrupt. Execution continues after the waking interrupt completes.

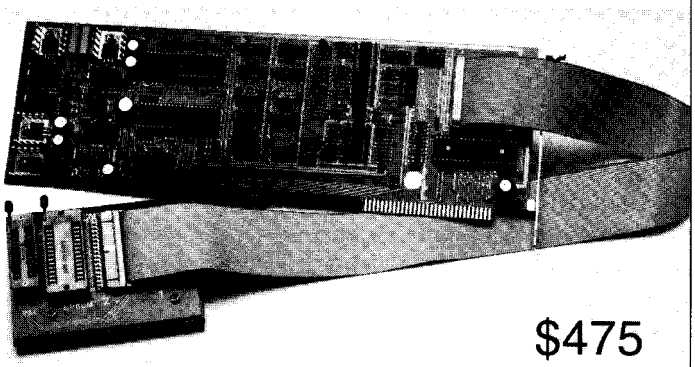
Timer 0 counts up from $EBDO and issues an interrupt when it rolls over to zero. With an 11.06.MHz clock, this means Timer 0 interrupts the 8052 about 180 times a second to run `Key Scan`. Any key change is debounced for nine scan times, or about 50 ms. If a chord is held for 256 scan times, or about 1.4 seconds, it will begin to repeat once every I3 scan times, or every 73 ms.

`Key Scan` peeks at the keyboard every time it runs, watching for bit changes. When a bit does change, `KeyScan` loads `KeyVerCounts` with the value of 9 (`#VerLoops`) and sets the `NewKey` bit so the next nine interrupts will pass to `Key Test`.

`KeyTest` is called while a key is being validated. It debounces the key by verifying the keycode doesn't


UNIVERSAL PROGRAMMER

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

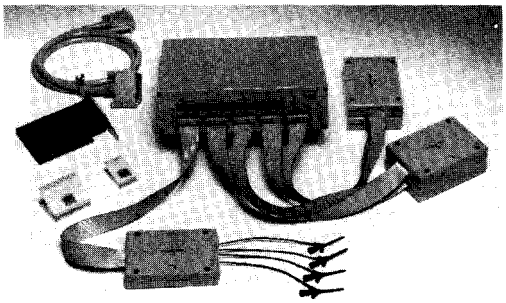


$475

Free software updates on BBS
Powerful menu driven software

400 MHz LOGIC ANALYZER

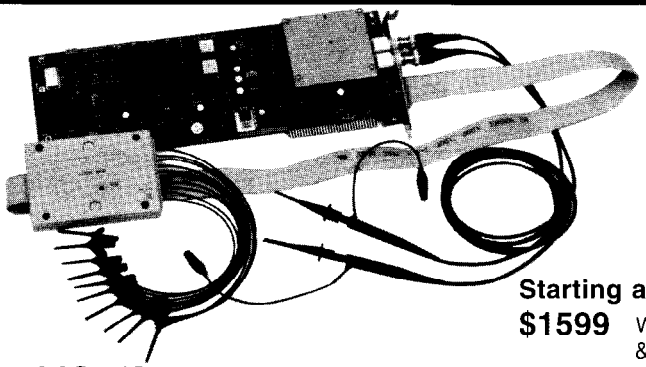
- up to 128 Channels
- up to 400 MHz
- 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



$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
$1599 With Pods
& Software

- 200 MSa/S Sampling Rate
- 2 Analog Channels (2ch. Digital Osciloscope)
- 8 Digital Channels (8ch. Logic Analyzer)
- 125 MHz Single Shot Bandwidth
- 4K Samples/Channel (Analog & Digital)

Call (201) 808-8990

Link Computer Graphics, Inc.
369 Passaic Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786

#118

change while the KeyVerCounts passes through KeyTest. Once the key is debounced, the KeyChange procedure determines whether the event observed was a key press or a key release. When a key is pressed, KeyChange remembers the new candidate chord and waits for another key change. If a key was released and a chord was being acquired, then the candidate chord is moved into CurrentChord and the Release and KeyReady bits are reset. The MainLoop will recognize the Key Ready semaphore as a signal to process the chord held in CurrentChord.

The ProcessChar procedure called by MainLoop translates the acquired chord into an ASCII character and stores it to memory and the LCD screen.

## THREE FINGER BOOT

The Three-Finger-Boot is a means to start the chordic keyboard up in modes not reachable from normal Chordic Keyboard operation. Boot up commands are given by holding down

Listing I-continued

```
      djnz  HoldCounter,KExit          ; has code been held long?
      mov   A,KeyHolder
      jz    KExit                      ; yeap, cuz it's no chord
      mov   CurrentChord,KeyHolder     ; put it in mail box
      setb  KeyReady                   ; not zero? long held chord
      mov   HoldCounter,#RepeatCount
      sjmp  KExit                      ; call it a chord and exit.
KTExit
      pop   PSW                        ; exit point for KeyChange
KExit
      pop   DPL
      pop   DPH
      pop   ACC
      ret                              ; End of interrupt routine

;KeyTest verifies the key's stability
KeyBad
      clr   NewKey                     ; start over after bit wiggle
      sjmp  KExit
KeyTest
      xrl   A,CandKeyHolder            ;is it the same?
      jnz   KeyBad                     ; Noise, start over
      djnz  KeyVerCounts,KExit         ; Keep checking..
      clr   NewKey                     ; Get ready for next one...
      mov   OldKeyHolder,KeyHolder     ; save the last key
      mov   KeyHolder,CandKeyHolder    ; get new
      push  PSW                        ; save carry flag
KeyChange
      mov   A,KeyHolder                ; compare new...
      clr   c
```

(continued)


**AN 8031 FAMILY TRAINING AND DEVELOPMENT PACKAGE FOR**

**EMBEDDED FUZZY-LOGIC CONTROL**

| | |
|---|---|
| HARDWARE: | The **R-535J** -- Rigel's best selling evaluation and training board |
| SOFTWARE: | READS -- Rigel's Embedded Applications Development System, and<br>FLASH -- Rigel's Fuzzy-Logic Applications Software Helper |
| 2 BOOKS: | ***'Programming And Interfacing With Microcontrollers"*** and<br>**"Embedded *Fuzzy-Logic Control*** |
| 3 MANUALS: | The **R-535J** User's Guide, the READS User's Guide, and the FLASH User's Guide |

**R-535J** Evaluation and Training board -- Siemens' 80C535 microcontroller, 32K SRAM, 32K monitor EPROM, watchdog timer, fully duplex serial port, 12 screw-type terminal blocks for digital I/O, 28 total I/O ports, all system signals available on two 32-pin headers, 2-way reset giving access to all interrupt vectors, analog-to-digital convertor. Small size (3.75"x 5") makes it ideal as an embedded controller.

**READS** -- Contains an editor, cross-assembler and host-to-board communications in an integrated environment Debugging with breakpoints and single stepping.

**FLASH** -- is a fuzzy-logic code generator which creates a set of subroutines and tables in the MCS-51 assembly language. No in-depth knowledge of fuzzy-logic control or special programming skills are needed with FLASH. The user writes the input to FLASH in plain English, from READS or any ASCII editor. The output is a block of assembly language code which the user supplements for the specific application. The FLASH Simulator computes and displays on the PC the outputs FLASH will generate from given inputs.

*'Programming And Interfacing With Microcontrollers, Experimenting With The 8031 Family Of Microcontrollers"* -- A comprehensive (250+ pages) hands-on approach is taken to present how the microcontrollers are programmed and interfaced to external circuitry to preform useful automation and control functions. Programming nuggets are given for each instruction and each on chip peripheral.

*Embedded Fuzzy-Logic Control"* -- 50+ pages, Subjects include: background on fuzzy-logic control, developing embedded fuzzy-logic applications, evaluating and fine tuning the fuzzy-logic controller, case studies, and Rigel's support tools.

**MANUALS** -- include source code for user-accessible system calls, complete circuit diagrams, example programs, and tutorials

**SPECIAL PACKAGE PRICE $250**

**RIGEL CORPORATION,** PO Box 90040, Gainesville, FL 32607 (904) 373-4629

Listing I-continued

```
        subb  A,OldKeyHolder            ;     with old
        jb    Release,KeyRelease        ; Releases or presses?
        jnc   KTExit                    ; Wait on another...
        setb  Release                   ; nope. it's a release...
        setb  KeyReady                  ; Valid key, so semaphore it
        mov   CurrentChord,OldKeyHolder: and put it in mail box
        sjmp  KTExit                    ; Process key will grab it
KeyRelease
        jc    KTExit                    ; Release, so keep waiting...
        clr   Release                   ; Keypressed, SO de-flag it
        sjmp  KTExit
```

the thumb, forefinger, or middle finger keys for about ¾ of a second after reset. The board and EPROM are configured to boot up directly into chordic keyboard operation when power is applied and no chordic keys are depressed. The default serial data rate is 2400 bps. Holding down the thumb button during reset will upload all 31 lines of stored text. Holding down the forefinger key will cause the EC-32 to enter a monitor program. Holding down the middle-finger key will erase all 31 lines of text. After uploading or erasure, the keyboard runs normally.

## FUTURE DIRECTIONS

A project like this is never complete. Thank goodness the development of word processors didn't begin and end with EDLIN. The main point of the Chordic Keyboard is to get text and ideas immediately into a digital form. To me, any work-in-progress on a piece of paper appears lifeless, whereas characters on a screen are agile and dynamic. Transferring chordic text to a PC requires only a driver program, and as soon as I get my head out of 8032 assembly mode I'll write some C code to do just that.

Programming the Chordic Keyboard to emulate a PC keyboard might be useful for people with limited keyboarding abilities. And maybe it would sell well at a Science Fiction Writers convention? ❑

***Scot Colburn is a Staff Administrator III at MCI's Systems Engineering facility in Colorado Springs.***

## 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.

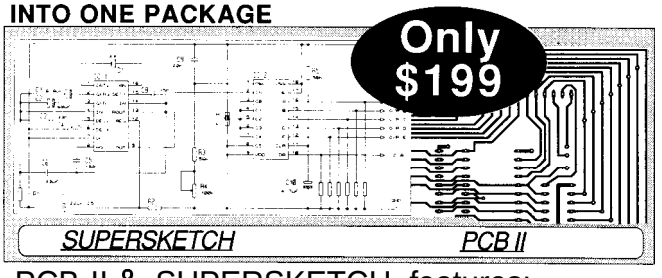
## I R S

404 Very Useful
405 Moderately Useful
406 Not Useful


# CADPAC II

TWO PROGRAMS FOR ONE LOW PRICE!!

**SUPERSKETCH & PCB II; INTEGRATED INTO ONE PACKAGE**




PCB II & SUPERSKETCH features:

* MOUSE DRIVEN *SUPPORTS CGA, EGA, VGA & SVGA,
* OUTPUT TO 9 & 24 PIN PRINTERS, HP LASERJET& HPGL PLOTTERS . OUTPUT TO DTP PACKAGES *
* PCB II ALSO HAS GERBER OUTPUT & VIEWING. •

**THE EASIEST TO USE CAD AVAILABLE**

**R4 SYSTEMS Inc.**
1111 Davis Drive, Suite 30-332
Newmarket, Ontario **L3Y 7V1**
(905) **898-0665**
**fax** (905) 836-0274

*Free DEMO Package Write or Call Today*

ALL PRICES ARE IN US FUNDS. PLEASE INCLUDE $7 S/H

**Download DEMO from BBS at (905) 898-0508 (2400/8/N/1)**

# Cross Assemblers

- **Local Labels and Cross** Reference
- Powerful **Macro Substitution Capability**
- **Machine Cycle Counting**
- **32 Significant Character Labels and Symbols**
- **Unlimited Include File Capability**
- **Selectable Intel Hex or Motorola Hex Object File**

# Simulators

- **Source View Symbolic Debugging**
- **Attach Keyboard, Screen or Data Files to Simulate I/O**
- **Machine Cycle Counting**
- **Ten User-definable Screens**
- **Unlimited Breakpoints, Memory and I/O Mapping**
- **Trace File to Record Simulator Session**

# Disassemblers

- **Automatic Substitution of Defined Label Names for Al Jumps and Branches**
- **Automatic Insertion of Supplied Comments and Expression**

# Application Source Libraries

- **16 and 32 bit Integer Arithmetic and Numeric/Strin Conversion**

# PseudoCorp

**716 Thimble Shoals Blvd.**
**Newport News. VA 23606**

**(804) 873-I 947 FAX:(804) 873-2154**
**BBS: (804) 873-4838**

#121

# Accessing the DS5000T Timekeeper from C

FEATURE ARTICLE

**J. Conrad Hubert**

Is time of the essence in your current project? Accessing the embedded clock/calendar in Dallas's DS5000T proves to be a challenging prospect, but well worth the effort.

The work presented here was part of a consulting project to develop an embedded system for the over-the-road trucking industry. In addition to its intended job, the client wanted the system to display time-of-day during periods of inactivity. Since I had already designed in the DS5000, and since all of the I/O decoding was done by a completely utilized PAL, the DS5000T was a perfect solution.

When Dallas Semiconductor added a time-of-day clock to their DS5000 microcontroller, none of the core 8051's functionality was lost or modified. Rather, they chose to employ a somewhat cumbersome bit-serial routine to gain access to the embedded clock/calendar (ECC). Communicating with the ECC requires selecting a special memory space prior to sending a 64-bit ID pattern followed by reading or writing 64 bits of time data. Fortunately, Dallas supplied efficient assembly language routines for communication with the timekeeper.

The advantages of writing code in a high-level language, with its ease of development, are well known. However, translating the bit-serial communications routines into C would have resulted in grossly inefficient code. That's why I modified Dallas's assembly language routines to conform to the calling conventions of my 8051 C compiler. This allowed me to invoke the bit-serial communications routines as though they were C functions. Later in the article, I'll detail what I did, but here is a generalized summary that you can apply to your own project.

First, you'll have to write a "shell" or "template" program in C to understand how parameters are passed. Next, you compile the shell using the compiler option that generates assembly language output. After that, you'll need to modify the assembly language output to include your own code, and assemble it. Then, you have to write and compile a C program which calls the assembly language subroutines. Finally, you'll need to use the linker to combine the assembly language subroutines with the C functions and make executable code.

Before diving into a description of the code, I should explain that this is

Listing 1-A *shell* C program is used *to find* out *how* your C compiler handles *the* interface *to* assembly *language routines.*

```
void  open(void){
/* not much interesting here */
}
char  rbyte(void){
  /* 5 is a dummy value to see how returns are handled */
  return 5;
}
void wbyte(char timedata) {
/* timedata is a dummy variable to see how
   parameters are passed */
}
void main (void) {
/* when compiled via assembly, main0
   shows how functions are called */

  char timedata;
  open();
  wbyte (10);
  timedata = rbyte();
}
```

the actual code from the client's prototype device (although most of the comments have been removed and integrated into this article). When I showed a draft of this article to a peer for review, he suggested some changes to the code that would have made it more general purpose. Although I agreed with his sentiments, I refrained from making any changes. Since I no longer have the prototype, I didn't want to chance introducing an error by presenting untested code.

## MAKING A SHELL

Listing 1 shows the shell program. I'll call it `shell.c` because I lack sufficient creativity to come up with something better. As you can see, it is composed of `main()` and three other functions: `open()`, `rbyte()`, and `wbyte()`. Although `main()` isn't used in the final assembly language program, I included it because I wanted to show how the functions are called and how parameters are passed. The function `open()` is the simplest; it neither returns a value nor is it passed any parameters. `rbyte()` does not accept a parameter, but it does return a value. The opposite is true of `wbyte()`: it only accepts a parameter and does not return anything.

Listing 2—*Inspecting the* **assembly language generated by the** *compiler for the shell program in Listing 1 reveals how you must write your code.*

```
      NAME shell(18)
      RSEG CODE(0)
      PUBLIC     main
      $DEFFN     main(0,0,1,0,0,0,0,0)
      PUBLIC     open
      $DEFFN     open(0,0,0,0,0,0,0,0),main
      PUBLIC     rbyte
      $DEFFN     rbyte(0,0,0,0,0,0,0,0),main
      PUBLIC     wbyte
      $DEFFN     wbyte(0,0,0,0,0,0,1,0),main
      EXTERN     ?CL8051L_4_10_L17
      RSEG CODE
;1.  void  open(void){
open:
; 2. }
      RET
; 3.
; 4.  char rbyte(void){
rbyte:
; 5.    return 5:
      MOV  R7,#5
; 6. }
      RET
; 7.
;8. void wbyte(char timedata) {
wbyte:
      MOV DPTR,#$LOCBX wbyte
      MOV  A,R7
      MOVX @DPTR,A
; 9. }
      RET
; 10.
; 11.
; 12.
; 13.      void main (void) {
main:
; 14.
; 15.        char timedata;
; 16.
; 17.        open();
      LCALL      $REFFN open
; 18.        wbyte (10);
      MOV  R7,#10
      LCALL      $REFFN wbyte
; 19.        timedata = rbyte();
      LCALL      $REFFN rbyte
      MOV DPTR,#$LOCBX main
      MOV  A,R7
      MOVX @DPTR,A
; 20.      }
      RET
      END
```

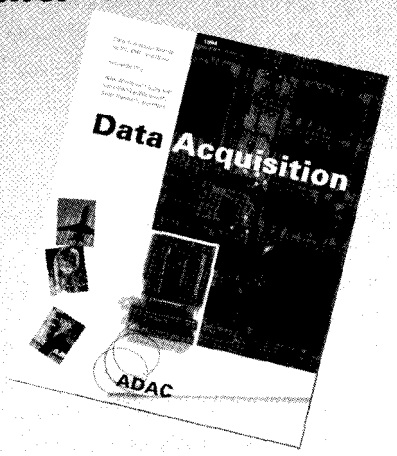
## COMPILING THE SHELL

Much of the first 12 lines of Listing 2 is information specific to Archimedes C-51 and won't be exactly the same for your compiler. The concepts, however, should be directly applicable. At this point, I will mention my fondness for Archimedes C-version 4, that is. Previously, I tried version 3 (which adheres to the ANSI standard for reentrance) and found it was too inefficient for my applications. Version 4 is much better suited to the embedded systems I develop. The only drawback with the new version is that it produces non-reentrant code and therefore will not support recursion. You also must be careful not to call a function from more than one "function tree." This conflict arises if an interrupt routine calls a function or library routine that is used elsewhere in the program. However, on balance, version 4 writes more efficient assembly code than I do and it does so much quicker.

The first line in Listing 2 indicates the name of the module is Shell. The number 18 in parentheses is type-checking information for the linker. According to the Archimedes manual, that entire first line is optional. **RSEG CODE(0)** indicates that the code is to be byte aligned (is there any other way to do it for an 8051?). Declaring a function **PUBLIC** indicates that it is visible to modules other than just itself. The **$DEFFN** directive specifies three pieces of information. First, the name of the function. Second, the numbers inside the parentheses indicate how many bytes of memory are occupied for a given memory type. From left to right, the eight memory types are: local DATA, local IDATA, local XDATA, local BIT, parameter

# 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.

*ADAC*

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

#122


**Listing 3**—*Assembly* **code** *for timekeeper callable from Archimedes C.*

```
          NAME    timekeep(18)
          RSEG    CODE(0)
          PUBLIC  open
          $DEFFN  open(0,0,0,0,0,0,0,0)
          PUBLIC  rbyte
          $DEFFN  rbyte(0,0,0,0,0,0,0,0)
          PUBLIC  wbyte
          $DEFFN  wbyte(0,0,0,0,0,0,1,0)
          EXTERN  ?CL8051L_4_10_L17
          RSEG    CODE

MCON      equ     0c6h
PCON      equ     87h
TA        equ     0C7h

; Subroutine executes the sequence of reads and writes that
;is required to open communication with the timekeeper.
; The subroutine returns with the timekeeper opened for data
;access and with both the accumulator and B register modified.

open:     LCALL   CLOSE       Make sure it is closed
          MOV     B,#4        Set pattern period count
          MOV     A,#0C5H     Load first pattern byte
OPENA:    mov     r7,a        WBYTE EXPECTS DATA IN R7
          LCALL   wbyte       Send out the byte.
          XRL     A,#0FFH     Generate next pattern byte.
          mov     r7,a        WBYTE EXPECTS DATA IN R7
          LCALL   wbyte       Send out the byte.
          SWAP    A           Generate next pattern byte.
          DJNZ    B.OPENA     Repeat until 8 bytes sent.
          RET                 Return from open.

 ; This subroutine ensures the registers of the timekeeper
; are closed by executing 9 reads of date and time
; registers. Subroutine returns with both accumulator
; and the B register modified.
;
CLOSE:    MOV     B,#9        ; Set up to read 9 bytes
CLOSEA:   LCALL   rbyte       ; Read a byte:
          DJNZ    B,CLOSEA    ; Loop for 9 byte reads.
          RET                 ; Return from close.

; Subroutine performs a "context switch" to the CE2 data
; space and then reads one byte from the timekeeping device.
; Then it switches back to the CE1 data space and returns
; the byte read in the accumulator, with all other registers
;  unchanged.

rbyte:    PUSH    MCON        Save MCON register.
          ORL     MCON,#4     Switch to CE2.
          PUSH    B           Save the B register.
          MOV     DPL,#4      Set up for data input.
          MOV     DPH,#0      Set high address byte.
          MOV     B,#8        Set the bit count.
LI:       PUSH    ACC         Save the accumulator.
          MOVX    A,@DPTR     Input the data bit.
          RLC     A           Move it to carry.
          POP     ACC         Get the accumulator.
          RRC     A           Save the data bit.
          DJNZ    B,LI        Loop for a whole byte.
          POP     B           Restore the B register.
          POP     MCON        Restore MCON register.
          mov     r7,acc      ARCHIMEDES CONVENTION
          RET                 Return from rbyte.

; Subroutine performs a "context switch" to the CE2 data
; space and then writes one byte from the accumulator to the
```

(continued)

Listing 3-continued

```
; timekeeping device.  Then it switches back to the CE1 data
; space and returns with al' registers unchanged.

wbyte:  mov     a,r7
        PUSH    MCON            Save MCON register.
        ORL     MCON,#4     ;   Switch to CE2.
        PUSH    B               Save the B register.
        MOV     DPH,    #0  ;   Set high address byte.
        MOV     B,#8            Set the bit count.
LO2:    PUSH    ACC             Save the accumulator.
        ANL     A,#1            Set up bit for output.
        MOV     DPL,A           Set address to write bit.
        MOVX    A,@DPTR     ;   Output the data bit.
        POP     ACC             Restore the accumulator.
        RR      A               Position next bit.
        DJNZ    B,LO2           Loop for a whole byte.
        POP     B               Restore the B register.
        POP     MCON            Restore MCON register.
        RET                     Return from wbyte.

        END                     End of module.
```

DATA, parameter IDATA, parameter XDATA, parameter BIT. Here, "local" refers to variables used inside the function, whereas "parameter" is memory allocated for variables which are passed. The third piece of information names all of the functions which call this particular function. For example, `$DEFFN wbyte (0,0,0,0, 0,0,1, 0), main` is associated with the function `wbyte()`. It is passed one byte of parameter data from the XDATA space. Also, you can see that `wbyte()` is called only by main. The declaration `EXTERN ?CL8051L_4_10_L 17` is optional. It provides type-checking information for the linker. Finally, `RSEG CODE` tells the assembler that the following code shall be placed in the relocatable code segment.

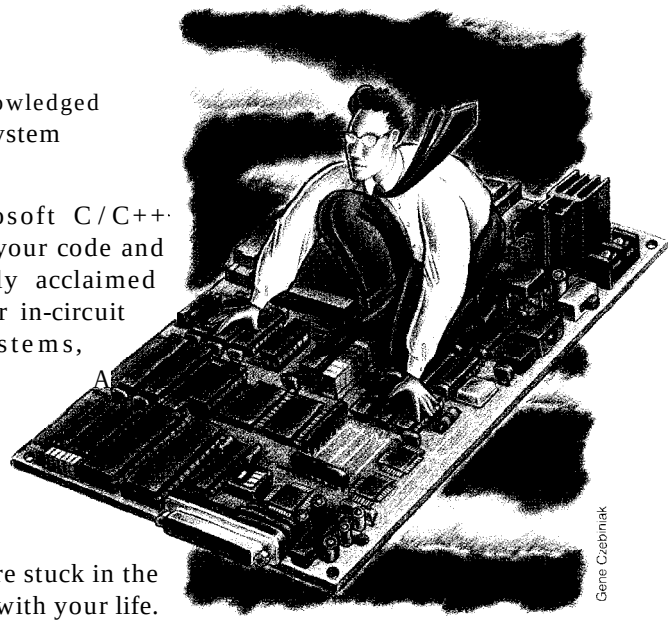
Notice that each line of C code is numbered and included in the assembly listing as a comment. This makes it easy to see how a specific line of C is translated into assembly language. The most important information, however, is that a one-byte value is passed in register R7. This is true whether the value is returned from a function or passed as a parameter to the function. After writing that last sentence, I wondered what happens when *both* a parameter is passed and the function returns a value. So, I compiled the following fragment via assembly to discover the answer.

```
char in_out (char dummy) {
    return (dummy * 2);
}
```


# No bugs on board.

Of course, what else would you expect from the acknowledged leader in Intel 80C186 and NEC V-Series embedded system software development tools.

Start with the complete Borland C++ and Microsoft C/C++ application templates from Paradigm LOCATE. Add your code and check it out in your target system with the highly acclaimed Paradigm DEBUG, either stand-alone or with popular in-circuit emulators from Intel, NEC, Applied Microsystems, CheckMate, Microtek, ZAX, and others.



Gene Czebiniak

If you get stuck, call our toll-free technical support hotline for free technical support. Then rest assured that your embedded application is rock-solid and free from those embedded system nasties.

After all, life is short and you can't play hard when you're stuck in the lab fixing bugs. Get **Paradigm,** get bug-free, and get on with your life.

PARADIGM

*Proven Solutions for Embedded C/C++ Developers*

Supports Borland® Tools

1-800-537-5043

Paradigm Systems ▪ 3301 Country Club Road, Suite 2214 ▫ Endwell, NY 13760 ▫ (607) 748-5966 ▫ FAX: (607) 748-5968

All trademarks are property of their respective holders.

#123

# $149 ICE?!!!

Yes, that's right! HTE has dropped the price of it's popular 8031/32 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 $448 complete!

# Controller 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/O, two PWM outputs, capture/compare registers, 16 digital I/O, one RS232 serial port, four JEOEC 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**

VISA MasterCard

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

(619) 566-I 892

70662.1241 @compuserve.com

#125


What I discovered is that R7 is used both when a value is returned from the function and when a value is passed as a parameter to the function, even when both operations occur in the same function. This makes sense because C is a "call by value" language. That is, it does not modify the parameter passed, rather, it operates on a local copy of the parameter.

Note: I do not mean to imply that it is impossible for a calling routine to modify a passed parameter. You can emulate the behavior of "call by reference" languages like FORTRAN by passing the parameter as a pointer. However, R7 can no longer hold the parameter because pointers require three bytes of storage in Archimedes' implementation of C on the 805 1.

The point here is to illustrate the beauty of compilation via assembly—even though your compiler manual may not explicitly state how a particular case is handled, you can easily write a shell fragment and see for yourself. In fact, compilation via assembly is one of the reasons I switched my high-level development language from Pascal to C.

## STIR IN YOUR OWN CODE

Perhaps the most time-consuming task is writing the assembly language subroutines. But for this project, I simply had to modify the code provided by Dallas Semiconductor in the January 1990 edition of the DS5000 User's Guide. In order to clearly show what was modified, my code appears in lower case with comments in upper case (Listing 3). The original Dallas code appears in upper case with comments in lower case. The following is a summary my changes:

- DPTR is no longer saved and restored
- *On entry, R7 holds the value to write using `wbyte`
- •On exit, R7 holds the value return by `rbyte`
- *Subroutine open was modified to account for changes 2 and 3

When interfacing with assembly language from C, it is crucial that you understand how the compiler is using registers. If you don't, your program may very well crash because you didn't leave the registers the way you found them. In Archimedes C-5 1, PSW, ACC, RO, R1, R2, R3, B, and DPTR hold temporary results and do not need to be preserved. However, R4, R5, R6, R7, C (carry flag), and SP must be preserved.

## ARE YOU TAKING ANY CALLS?

Now comes the fun part-writing the C code. Even if you are not fluent in C, most of the code presented in Listing 4 should be fairly understandable. One notable exception is found in `mainO.Theconstructfor (;;)` looks pretty strange if you've never seen it before. The intent of this empty for statement is to produce an infinite loop. Now I'll discuss the other functions in the order in which they appear in the program.

The only thing tricky about the function `read_timekeeper()` is that all 64 bits of ECC data must be read, since the data is sequential (not random) access. This means eight calls to `rbyte()` must be performed. See

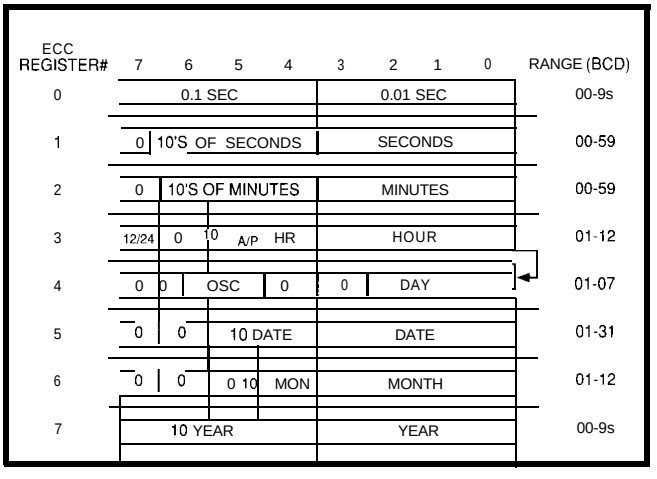



Figure 1—*The DS5000T* accesses *the* clock/calendar information using a 64-M *serial stream in* which *all the time and date* information is sent. *There is no way to access just one piece of* information; it *must all be sent every time.*

Listing 4—*This demonstration program shows how all the pieces fit together.*

```
/* Compiled under Archimedes C-51 Version 4 */

f/include <io51.h>
#include <stdio.h>

#define FALSE 0
#define TRUE 1

struct time_struct{
  char year, month, date, hour, minute
};

struct time_struct rtc;

char time_to_read_clock = TRUE; /* set by ISR */

extern void open(void);
extern char rbyte(void);
extern void wbyte(char);


void read_timekeeper(void){
  EA = FALSE;                   /* don't let an interrupt occur */
  open();
  rbyte();                      /* 0.1 and 0.01 sec */
  rbyte();                      /* ten sec. sec */
  rtc.minute = rbyte();         /* 10 min, min */
  rtc.hour = rbyte();           /* hour */
  rbyte();                      /* day */
  rtc.date = rbyte();           /* date */
  rtc.month = rbyte()           /* month */
  rtc.year = rbyte();           /* year */
  EA = TRUE;
}


interrupt [0x0B] void timer_0_overflow(void){
  static unsigned char trip-count = 0:

  if ( !trip_count++)               /* trip-count equals zero? */
    time_to_read_clock = TRUE;



void show_time(void)
  printf("%2x", rtc.month);

  if (rtc.date < 0x10)
    printf("-0%1x", rtc.date);      /* show leading zero */
  else
    printf("-%2x", rtc.date);

  if (rtc.year < 0x10)
    printf("-0%1x  ", rtc.year);    /* show leading zero */
  else
    printf("-%2x  ", rtc.year);

  printf("%2x:", rtc.hour & 0x1F);  /* don't show leading zero */

  if (rtc.minute < 0x10)
    printf("0%1x ", rtc.minute);    /* show leading zero */
  else
    printf("%2x ", rtc.minute);

  if (rtc.hour & 0x20)              /* is PM bit set? */
    printf("PM");
  else
```

*(continued)*

Figure 1 for a complete description of how ECC register data is defined.

The function timer_0_overflow() iscommonly referred to as an "administrative" function. It is an interrupt service routine (ISR) activated whenever Timer 0 overflows. That is, once every 65,536 µs when using a 12-MHz crystal. Because I used an 11.0592-MHz crystal, overflow actually occurs roughly every 71 ms.

Since the resolution of my time-of-day clock was minutes rather than seconds, it was not necessary to read the ECC very often. By incrementing t. r i p-count once each interrupt, trip_count will wrap around to zero once every 256 passes through the ISR (every 18.2 seconds). Each time t r i p_co u n t equals zero, a global flag is set indicating that it is time to read the ECC. In this way, substantial overhead was avoided by polling the ECC only every 256th occurrence of this interrupt.

An unusual feature of this C code is the use of the hexadecimal format specifier in both the input function scan f() and the output function p r i n t f (). Format specifiers, as used in the scan f () function, allow you to interpret ASCII data in a specific way. Which format specifier is used determines how ASCII input will be treated numerically. For example, the ASCII characters "2" and "7" may result in the numeric value 27 decimal if the specifier is "%d," or 27 hex if the specifier is "%x."

Since the time data are represented as binary-coded decimal (BCD) by the DS5000T, using a BCD format specifier would be ideal. Unfortunately, C doesn't give us this option. Although it may not be obvious at first, the hexadecimal format specifier works correctly for BCD in this instance. In fact, operation of the I/O functions show_timeO and set_ timekeeper0 rely on the hexadecimal format specifier to simplify these routines.

From Figure 1, it is easy to see that two nybbles of time data are packed into each byte of ECC register data. Let's take minutes, which range from 0 to 59, as an example. If we

Listing 4-continued

```
    printf ("AM");
}


void  set_timekeeper(void){
  int temp = 0xFF;              /* this value is never in range */

  while  ((temp > 0x12)||(temp < 1)){
    printf("\25Month: 1..12 ");
    scanf ("%x",&temp);
  }
  rtc.month = (char) temp;

  temp = 0xFF;
  while  ((temp > 0x31)||(temp < 1)){
    printf("\25Date: 1..31 ");
    scanf("%x", &temp);
  }
  rtc.date = (char) temp;

  temp = 0xFF;
  while  (temp > 0x99){
    printf("\25Year: 0..99 ");
    scanf ("%x",&temp);
  }
  rtc.year = (char) temp;

  temp = 0xFF;
  while  ((temp > 0x12)||(temp < 1)){
    printf("\25Hour: 1..12 ");
    scanf("%x", &temp);
  }
  printf("\25Is it Afternoon?");
  if (low_level_get() == 'Y')
    rtc.hour = (char) temp | 0xA0;   /* set 12-hr mode & PM bit */
  else
    rtc.hour = (char) temp | 0x80;    /* set 12-hr mode only */

  temp = 0xFF;
  while  (temp > 0x59){
    printf("\25Minute: 0..59 ");
    scanf("%x", &temp);
  }
  rtc.minute = (char) temp;

  EA = FALSE;                   /* don't let an interrupt occur */
  open();
  wbyte(0x00);                  /* dummy */
  wbyte(0x00);                  /* dummy */
  wbyte(rtc.minute);            /* minute */
  wbyte(rtc.hour);              /* hour */
  wbyte(0x01);                  /* dummy "date of zero invalid" */
  wbyte(rtc.date);              /* date */
  wbyte(rtc.month);             /* month */
  wbyte(rtc.year);              /* year */
  EA = TRUE:
}


void main (void) {
  for(;;) {
    if  (time_to_read_clock){
      time_to_read_clock = FALSE:
      read-timekeeper0;
      show_time();
    }
  }
}
```

encode 27 minutes as two BCD digits, we get 00101110. This binary representation is 27 (hex). Had I done the I/O using decimal representations, additional processing would have been needed to massage the data into BCD. Although the library routines which interpret the hexadecimal format specifier still must do some processing, it is transparent to us. This keeps the code simple. Please note that this trick only works for I/O, and that any calculations based on these hexadecimal values would be incorrect.

Format specifiers are again used in the pri ntf () function. In pri ntf (), they also serve as place holders to permit output of variables in otherwise predetermined ASCII strings. For example, pri ntf ("-%2X ", rtc.year) will print an ASCII hyphen followed by the two-digit hexadecimal value r t c . ye a r, followed by three ASCII blank spaces. Again, using the hexadecimal format specifier saves the bother of processing BCD data as decimal values.

Finally, notice that some of the p r i n t f ( ) statements contain the octal value "\25." This is a special character which cleared the vacuum fluorescent display I was using. Thefunction set-timekeeper0 accepts and qualifies user input of time data. I chose to use the w **h** i l e construct rather than do-wh i l e for the code that gets user input to set the clock. Although the **do w h** i l e statement would have obviated the need to set t emp to an out-of-range value, I found the test conditions for the w **h** i l e loop more understandable. For example, both of these fragments accomplish the same thing:

```
temp = 0xFF;
  while((temp>0x31)||(temp<1)){
    printf("\25Date: 1..31 ");
    scanf("%x", &temp);

rtc.date = (char) temp:
```

```
do {
  printf("\25Date: 1..31 ");
  scanf("%x", &temp);
} while((temp>0x32)&&(temp<0x00));
rtc.date = (char) temp;
```

I find the former easier to grasp because dates less than 1 or great& than 31 are not allowed. However, it is purely a personal preference. You can also see from the function `set_time keeper()` that there are several special-purpose bits interspersed with the time data. Specifically:

- Bit 5 of ECC register 4. Setting this bit to zero enables the 32.768-kHz oscillator, thereby allowing the clock to keep time.
- Bit 7 of ECC register 3. This selects whether the clock operates in 12- or 24-hour mode.
- Bit 5 of ECC register 3. This bit serves two functions. In 12-hour mode, it is the AM/PM indicator. However, in 24-hour mode it becomes the second ten-hour bit. This allows the register to hold a BCD representation of the hours 20 to 23.

## THE LAST PASS

The final step in this process is linking. There is probably not much point in detailing how the Archimedes linker works since little can be generalized about the process. Your linker will invariably be different. Suffice it to say that the linker combines previously assembled relocatable code into absolute executable code. In order to do this, the linker must deal with issues like the starting address for code, how big a stack is needed, where to put different types of variables, as well as other impleinentation-dependent considerations. ❑

*J. Conrad Hubert is a principal in Deus Ex Machina Engineering Inc. where he provides consulting services for the development of hardware and software for embedded systems, data* acquisition, and digital signal *processing. He may be reached at (612) 645-8088.*

**IRS**

*407* Very Useful
408 Moderately Useful
409 Not Useful

**Free Demo Disk! Call 1-800-221-6630**

# What is C_thru_ROM?

**ROM Your Borland or Microsoft *C/C++Code.***

***C_thru_ROM*** **is the complete ROM development software tool kit. It lets you run Microsoft and Borland C and C++ programs on an embedded 80x86 CPU without using DOS or a BIOS.**
***C_thru_ROM*** **saves you money. There are no DOS or BIOS royalties to Pay for your embedded systems.**
***C_thru_ROM*** **is complete! It includes the following and much more:**

- *Supports Borland's Turbo Debugger.
- •Remote Code View style source level debugger.
- ROMable startup code brings CPU up from cold boot.
- ROMable library in source code.
- *Flexible 80x86 Locator.

***COMPLETE PACKAGE ONLY $495. 30-DAY MONEY BACK GUARANTEE.***

**Datalight®**

107 N. OLYMPIC, SUITE 201 • ARLINGTON, WA 98223 • (206) 435-8086 • FAX: (206) 435-0253

#126

## AvCase II 8051 and 680x0 Will Travel

Work in the field using your laptop with AvCase II, Cactus Logic's IDS debugger and Promax parallel port EPROM programmer. AvCase II includes an ANSI C compiler, macro assembler, linker, run-time and floating-point libraries with source code, librarian, plus remote monitor and simulator debuggers.

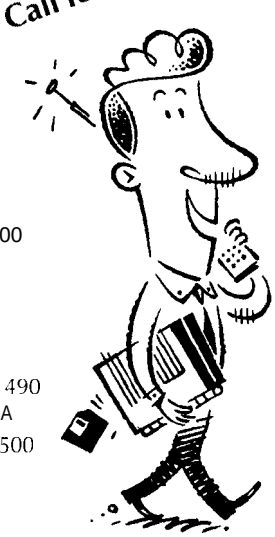
- Integrates all your tools
- Manages the edit/compile/assemble/debug loop
- Compatible with IEEE695 ICEs
- Project management, MAKE, on-line help
- Add your favorite editor, ICE and EPROM

AvCase II 8051
- Supports derivatives from Intel, Philips and Siemens
- New global optimizer produces fast, tight code
- New high-level simulator debugger

AvCase II 680x0 Supports 68000/10/20/30 and CPU300

***Save*** $200 ***on AvCase II until 12/31/93***
***List Price: $1795 – Your cost $1595***

Call for free demo



AVOCET SYSTEMS: INC.
120 Union Street, P.O. Box 490
Rockport, Maine 04856 USA
207-236-9055 ▪ 800-448-8500
Fax 207-236-6713

*Software Development Tools for Embedded Systems*

# Designing with FPGAs

## Part 2: An Example

**FEATURE ARTICLE**

**Del Hatch**

**Here's where the design rubber meets the FPGA road. First, a look at design structure and its compilation. On your mark, get set...now we're in the driver's seat and ready to stuff a real project into an FPGA...GO!**

Last month we learned about a type of logic chip that has become increasingly popular with engineers and experimenters alike-the field programmable gate array (FPGA). These chips can easily accommodate a design that uses hundreds of conventional gates. Despite the obvious appeal of a component of this nature, they are avoided by some people who may have incorrectly concluded that these devices are too expensive or difficult to use.

This month, I'll work further toward dispelling this myth by explaining the relatively inexpensive and painless process that can be used to implement a design using an FPGA.

The clock project I will use as an example uses an FPGA to hold a design that is the equivalent of approximately 39 TTL chips or about 437 gates. The design of the logic system in the clock took four days. The first two evenings were dedicated to designing the logic and another two evenings were spent entering the logic and fitting it into an FPGA. The two evenings spent working the design into an FPGA probably saved me dozens of evenings that would have otherwise been consumed with wire-wrapping a large, power-hungry logic board!

As mentioned in last month's article, I am focusing on the Xilinx 3000 family of FPGAs because of their popularity, their reprogrammability, and their ability to self-load the logic design from easily programmed EPROMs. Another reason for this choice is that design software is readily available that is relatively inexpensive.

Computers (either PCs or workstations) are used to take a design concept and convert it to a format that will be put into a running FPGA. The last step of the design process is a file stored in an EPROM that determines how the FPGA's logic will be configured. But before we get to that, let's look at how the process is started.

### DESIGN ENTRY

The logic that will eventually be implemented in the FPGA must be defined first. A common method is to enter a schematic of the design into a schematic capture program. This requires that the gates and connections be individually entered.

Another increasingly popular method of design entry is to use a hardware description language. This method allows the logic to be defined by a textual description of the functions that are required. A less popular method is to enter the waveforms that are required by the system and let the software generate the logic from this information. The last two methods require that the software synthesize the gate-level logic which, in turn, means these programs cost much more than simple schematic entry software.

When the designer is willing to accept the "burden" of developing and entering the gate-level design, the tools required become much more affordable. ViewLogic and OrCad are the two most popular ways of schematically entering a design destined for use with Xilinx FPGAs.

Structure is extremely important when entering a design schematically for an FPGA, and the use of ***hierarchical design methodology*** is a good framework for imposing the required structure in the design. Using this method, a "block" on a higher level represents an entire section of logic that is defined more precisely on a lower level. For example, the highest level would be a block representing the entire FPGA design. The next lower level could show many blocks representing various combinatorial or register-based functions like adders, bus multiplexers, and the like. The next level down in the hierarchy

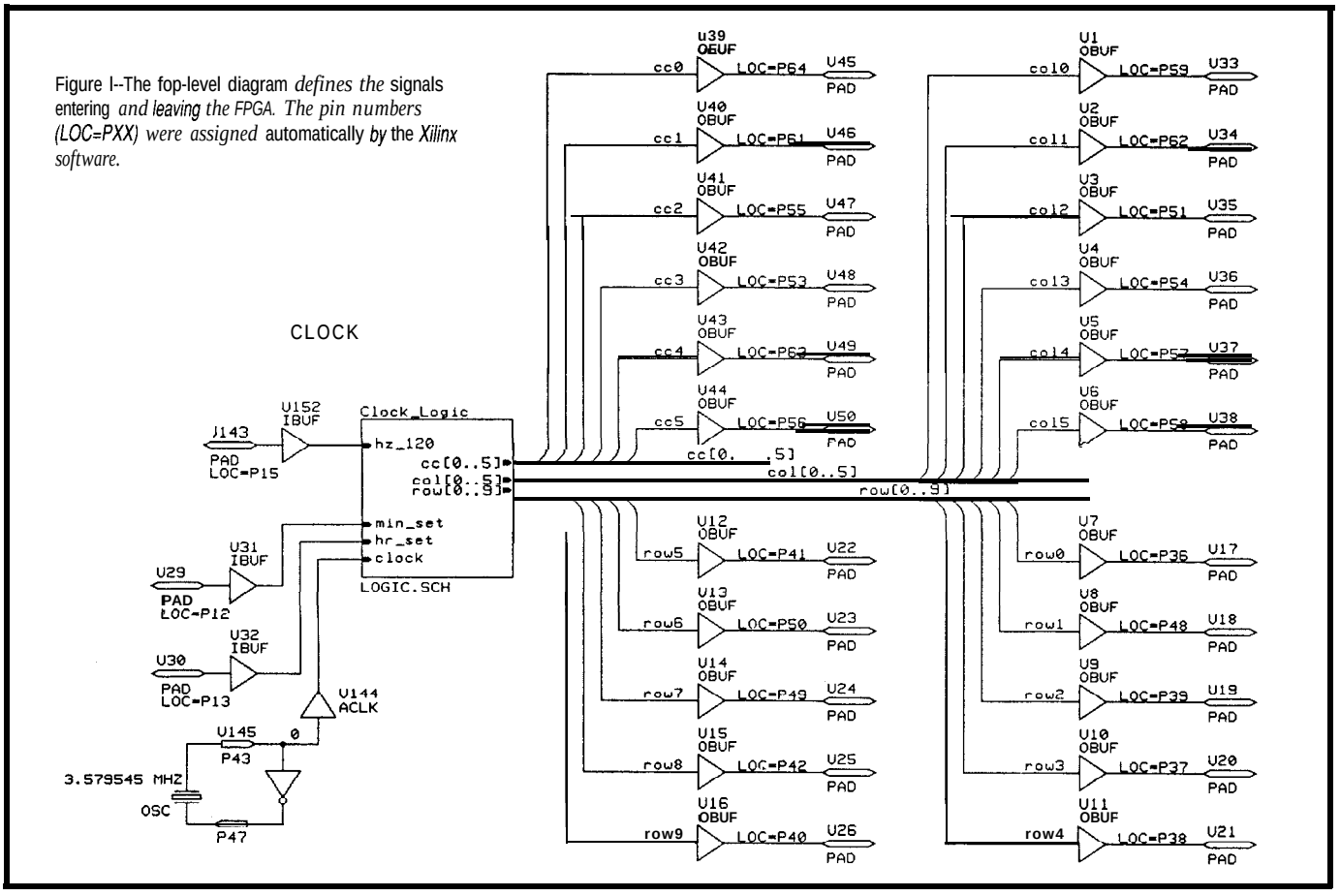



Figure I--The fop-level diagram *defines the* signals entering *and leaving the FPGA. The pin numbers (LOC=PXX) were assigned* automatically *by* the *Xilinx software.*

would then show how the gates are configured to accomplish the functions that were detailed in the upper levels.

Once the schematics are entered, an FPGA-specific step is necessary—defining how the signals get into and out of the chip. This is usually done on the highest-level diagram of the hierarchy. It shows the input and output buffers and the "pads" that represent the actual pins on the FPGA. Figure 1 shows the top-level schematic diagram for the clock design.

The pin numbers that carry a signal into [or out of) the FPGA can be set by the designer. However, doing this may limit the ability of the place-and-route software (which I will explain later) to implement the design inside the FPGA, and so I do not recommend assigning pins to signals during the first attempt at placing and routing the chip. A better method is to let the software assign an optimum placement for the inputs and outputs (it will do this automatically]. If the design uses a considerable amount of the FPGA's logic (more than 70%), or runs at higher frequencies (greater than 10 MHz), you are usually better off if you let the software assign the pinouts for you.

After the pin assignment has been generated, it is advisable to go back to the schematic originals and assign those locations to the I/O pins. This will force the computer to always use that same pinout in the future. This way you can make changes in the internal schematics of the design, and the FPGA pinout will never change. This is obviously important if you don't want to change the printed circuit board or wire-wrap connections every time you change the FPGA's internal logic.

## COMPILING THE DESIGN

The next phase in the design process is to compile the design. This phase actually consists of many small steps which can run automatically in sequence without any input from the user. The first step converts the schematics to a standard format that is independent of the software that was used to enter the design.

The next step converts the logic blocks and flip-flops to Xilinx logic blocks, and then performs circuit optimization. This is necessary and very useful in fitting a design into the FPGA. For example, when a macro is used, some of the internal gates may generate signals that are not used outside of that macro. In that case, those logic gates can be deleted. This step often reduces the gate count significantly.

The short story is, compiling takes the logic you designed and combines and partitions that logic into Xilinx *configurable logic blocks* (CLBs) efficiently. The best part is that all of this is done automatically, and even if you know nothing about the internal workings of the chip, it gets done just as well!

## PLACE AND ROUTE

The next step places the CLBs in the FPGA so the routing phase (the step after placement) is easier to

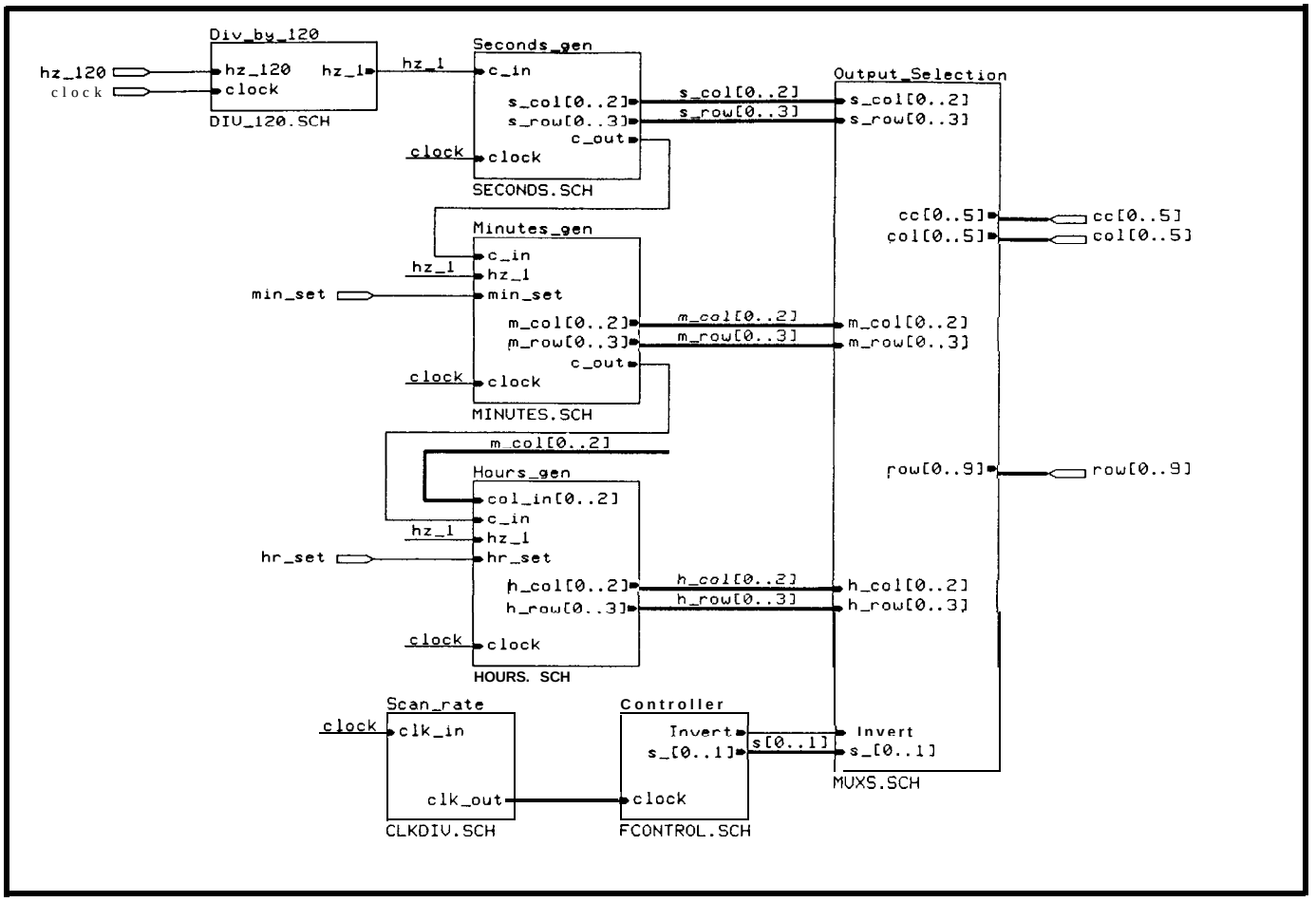



Figure 2—*The second layer in the FPGA schematic* hierarchy *shows the block-level functionality of the entire clock design, plus the multiplexers used when setting the time.*

complete. The placement phase will also assign physical pins to the I/O signals if the user has not done so, as discussed above.

The routing step "wires" all the logic blocks together. There is a finite amount of routing resources with a fairly large number of routing options, so this procedure is very important. In the Xilinx 2000, 3000, and 4000 families, a large percentage of the signal delays in the chip are routing delays, so the effectiveness of this step can play a large factor in how fast the design can run.

The output of this process is a file that completely describes the finished design. The final step is to convert this file to a format that can be programmed into an EPROM. This consists of running two utility programs, one that makes a bit file, and another one that converts it into an EPROM file in Intel hex format. The EPROMs are then programmed with the information in that file and plugged into the circuit.

Xilinx has recently announced a package called the Xilinx Base Development System which includes the schematic (and simulation) libraries and interface for OrCad or ViewLogic. It also includes the software necessary to compile and place and route the internal FPGA logic. The pricing at the time this article was written was just under $1000.

## THE CLOCK

I'll use a unique desk clock as a project that provides a more concrete description of some of the things I typically run into when working with FPGAs.

This clock uses a ring of bicolored light emitting diodes to indicate the position of the "hands" on the clock face. The "hour hand" is represented by a red light, the "minute hand" glows green, and the "second hand" moves around the face with a yellow light. The lights are time-multiplexed, so only one is on at a time. However, the logic turns them each on in sequence rapidly, so it appears that three different colors are on simultaneously.

The basics of the clock are simple: three counters called Seconds_gen, Minutes-gen, and Hours_gen (shown in Figure 2) keep track of the seconds, minutes, and hours. When the second hand reaches the beginning of the minute, it increments the position of the minute hand, and at the top of the hour, the hour counter gets incremented. I wanted the hour hand to move smoothly around the clock face instead of jumping from one hour to the next, so the value of the minutes counter determines where the hour-hand light should appear. The closer it is to the end of an hour, the closer it moves to the next hourly position.

The controller section (labeled Control 1 er) controls which LED is lit and in what color. It controls the multiplexers in the Output_Section to sequence through the hours, minutes, and seconds information.

The control line named Invert is used to select the color: red or green.

The controller section runs at the frequency of the crystal divided by 4096 and is not used for timekeeping purposes. For timekeeping, a 120-Hz signal is generated from the AC line with a full-wave rectifier and a Schmitt-triggered optocoupler. This is brought into the FPGA on pin 11 and then divided internally to provide the 1 -Hz clock rate for the seconds counter.

This l-Hz signal is also used to set the clock. The signals min_set and hour_set route the l-Hz line to the minutes counter or to the hours counter to move the hands around the clock face.

The schematic for the circuitry outside of the FPGA is shown in Figures 3 and 4. Figure 3 shows the external circuitry including the 120-Hz signal conditioner, the power supply, and the EPROM interface. Figure 4 details the matrix of LEDs and their drivers. If the Xilinx chip could put out more current, I would not need these buffers, which account for 67% of the ICs on the board!

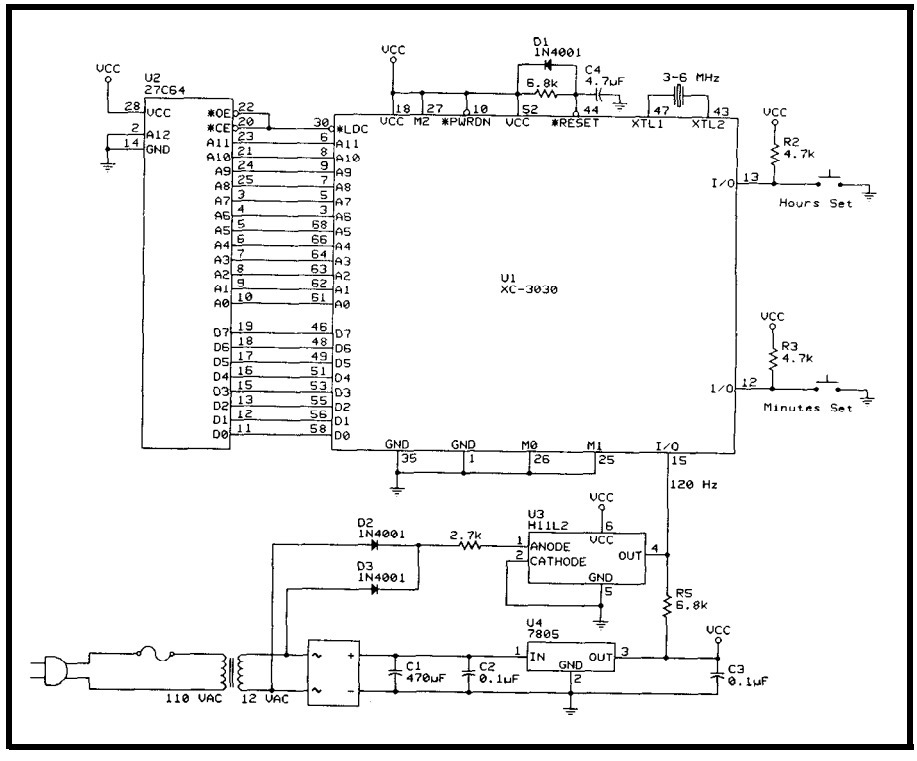



Figure 3-Except for the buffers and LEDs, the schematic for the clock is shown. The FPGA (U1) loads itself from the EPROM (U2) and receives timing information from the opto-coupler (U3).

## THE PROCESS

The journal I kept during the process of developing this FPGA records some typical pitfalls and solutions that may prove useful to first-time FPGA designers.

As I mentioned previously, it took two evenings to fit the design into the FPGA. The first night was spent entering the design in OrCad and then trying to compile it. I looked at the output file and kept seeing that the compiler was deleting very important logic. In the schematic I intended to connect some lines (and split buses) using labels instead of wires, but I had to convince OrCad of that fact. It was usually because I had mislabeled signal or bus lines. The lesson from this is you should always check that the logic you want in the chip is getting into the file that is used to generate the FPGA.

When I started the project, I didn't know how large an FPGA would be required. After the first evening, I had enough information to select one. The compiler told me I would need 62 CLBs to hold the design. The Xilinx XC3020 has 64 CLBs, but it is usually impossible to place-and-route a completely full chip. Also, it would leave no room for later design changes,


# Datrax 52 Remote Terminal & Programmable Controller

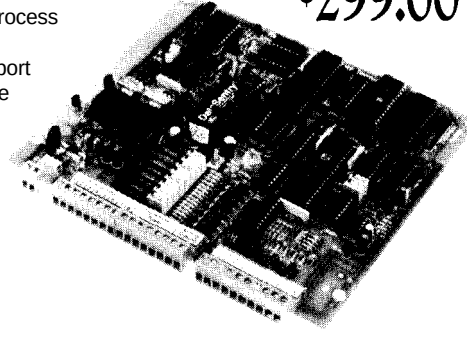
- Intel 8052 CPU
- High Speed Floating Point BASIC
- 32K Static RAM
- 16K EPROM with Programmer
- RS-232 Communication Port
- RS-232 Printer Port
- Eight 4-20 mA Input
- Four Discrete Input
- Four Relay Drivers
- Watch Dog Timer
- Twelve Diagnostic Lamps
- Extensive Surge Protection
- 12/24 Vdc Power Requirements
- Operating Temp -40 to +70 C

***Options:***

- Bell compatible 202 Modem
- Clock/Calendar
- Non-Volatile RAM
- 32 Bit Counter

The Datrax 52 is a computerized terminal aimed primarily for data acquisition, process control, and test instrumentation. Key Telemetering offers full customer support from initial design, application software through on-site assistance and training, to after sales maintenance.

$299.00



Data Track Systems
2829 Lewis Lane
Owensboro, KY 42301
Phone (800) 484-4121 Code 9142
FAX (502) 683-9873

#129

so I selected the XC3030 which has 100 available CLBs. I recommend always leaving room for future changes and improvements. Fixing bugs rarely requires *fewer* gates!

A complete place-and-route procedure takes about 30 minutes on my 386/40 PC using an older version of software (the newer versions of this software are said to be faster). However, the Xilinx "place" and "route" procedure can be interrupted if a completely optimized placement is not required, which is often the case for slow designs or if the FPGAs are not very full. This can save a lot of time when design changes are made often.

During the debugging process I needed to see what was happening with a few internal signals. In order to view them, I added a couple of pins on the top level diagram. Then I routed wires on each successively lower level of the schematic hierarchy and connected them to the signals I needed to view. After this change, it was easy to complete another quick run of the compiler and burn an EPROM. These signals were then available to me outside the device and allowed me to see what was happening. These "internal test points" can be easily changed to see what is going wrong, which is especially valuable if you do not simulate the logic first.

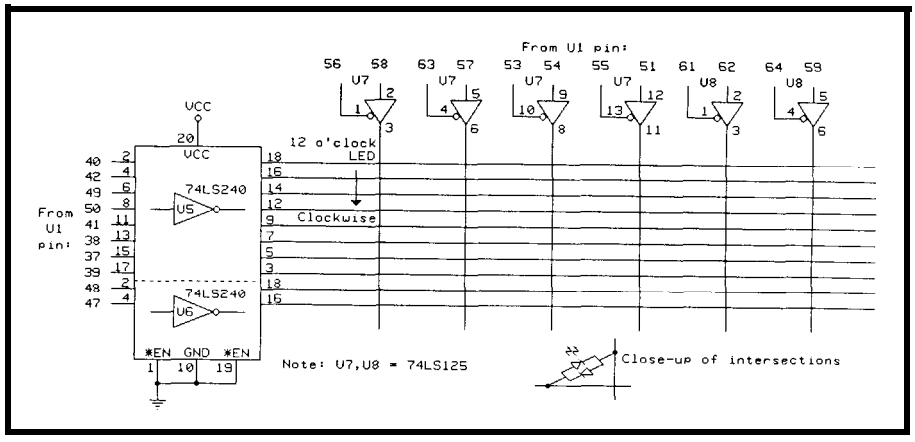



**Figure** *4-The buffers* **and** *bicolor LEDs are wired in a matrix to reduce wiring. The LED voltage polarity is reversed by the logic to illuminate the selected LED red or green.*

The first version of my design clocked all the FPGA flip-flops (using the global clock buffer) at 120 Hz. It later became apparent that this caused too much flicker in the "seconds" LED. To fix this, I decided to clock the FPGA flip-flops at a much higher rate using an external crystal, and to have the counters operate at 120 Hz. This required major changes to the logic, which were readily done by changing the schematics in OrCad.

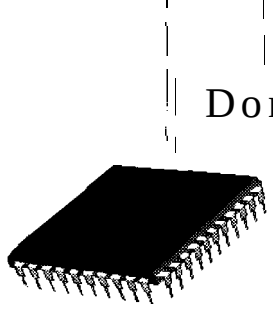
This overhaul of the clock's operation really illustrates the advantages to using FPGAs—the only changes *outside* the FPGA were the addition of a crystal across pins 43 and 47 and the movement of a single wire. If the same changes were necessary to a design implemented in discrete logic, I would still be wire-wrapping today!

Many times I wanted to make changes, such as modifying the appearance of the second-hand LED from an orange to yellow, or to simply correct errors I had made in the design. The process is the same-change the schematics, recompile and route the design, and program an EPROM. With an FPGA, you never have to heat up a soldering iron or pick up a wire-wrap tool to make internal logic changes.

## TIME TO GO

I hope the advantages of using FPGAs have been made more clear with this clock example. Using FPGAs for designs has many advantages, with the only major drawback being the cost of the software tools. If you can convince your boss to buy them for you, or get permission from your spouse, it will allow you to do medium-to-large projects with a minimum of hardware cost and untold savings in time during the entire design and prototyping process. ❑


**Drop 68HC11 Flexibility into Your Application**



Don't worry about which microcontroller to choose when you can rely on the industry leader.

- MODULAR, STACKABLE DESIGN ENSURES SMALL FOOTPRINT AND BIG PERFORMANCE.
- ADD ONLY THE FEATURES YOU NEED - LOWERING OVERHEAD
- FULL FEATURE 'C' COMPILER AND MACRO ASSEMBLER AVAILABLE
- PRICES STAR T AT $59 IN SINGLE QUANTITIES

Micro Wonders
320 Brown St Suite 404
West Lafayette, IN 47906
(317) 420 - 1686

#131

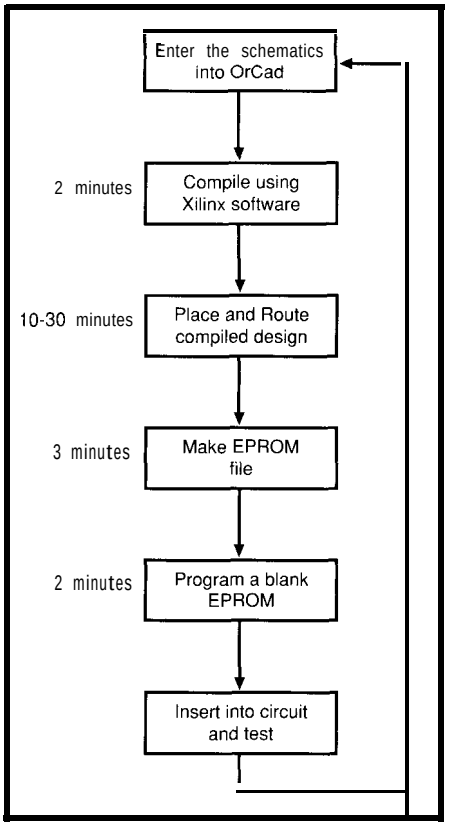



Figure 5—*The process I used for this FPGA design shows the approximate time for each step.*

*Del Hatch is an electrical engineer at Sandia National Laboratories, California, and thinks that any design requiring more than three gates should be put in an FPGA.*

## CONTACT

OrCAD
3 175 N.W. Aloclelc Dr.
Hillsboro, OR 97 124
(503) 690-9881
Fax: (503) 690-9891

ViewLogic Systems, Inc.
293 Boston Post Rd. West
Marlboro, MA 0 1752
(SOS) 480-088 1
Fax: (508) 480-0882

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

## I R S

410 Very Useful
411 Moderately Useful
412 Not Useful


8051 68HC11 68HC05 COP8




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

- Supports 8X51/31, 8X52/32, 8X51FA/FB/FC
- Real-time and Nonintrusive
- a 64K Program Memory 64K Data Memory
- a 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 inter-changeable 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 Freq. | 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 3/92, 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 backguarantee, 10 day trial periods, rental plans and free technical support Call today for FREE demo diskette.

VISA MasterCard

(800) 638-2423

MetaLink® Corporation

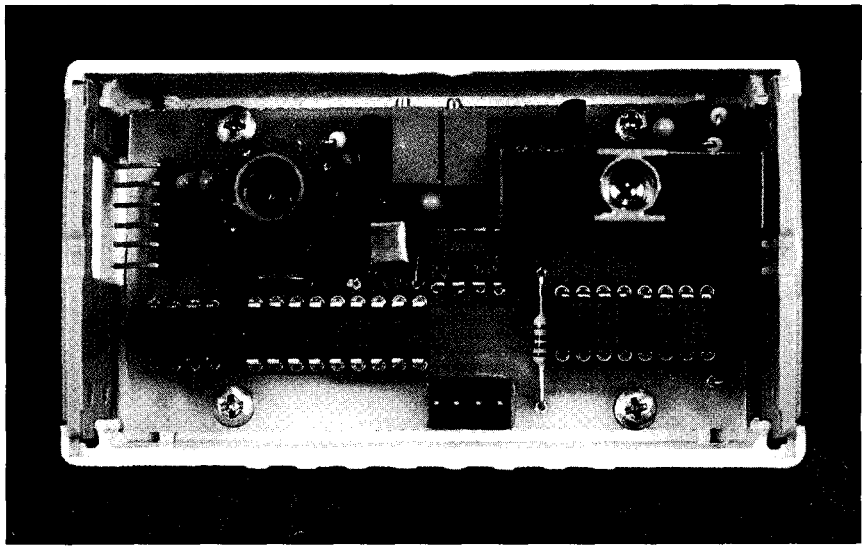
MetaLink Corporation
325 E Elliot Road, Chandler, Az 85225
Phone: (602) 926-0797 Fax (602) 926-1198

MetaLink Europe GmbH
Westring 2. 8011<85614>Kirchseeon-Eglharting
Teleton (08091) 2046. Telefax (08091) 2386

#132

# 5th Annual Design Contest Results

Project descriptions by Rob Rojas



DESIGN CONTEST WINNER

## 1st place Laser Range Finder — Tom Ward

Using an innovative combination of a laser diode and a CCD, Tom was the unanimous winner of this year's contest. The Laser Range Finder can be used in any application that requires accurate distance measurements from inches to yards.

About the size of a pocket pager, the Laser Range Finder is an ideal module for direct connection to robotic vehicles. Operation consists of a Hitachi HL671 1G laser diode producing a spot of light on an object. An image of the spot is then reflected to the surface of a CCD (CCD11A from Loral Fairchild). Distance measurements are calculated based on where the spot falls on the CCD array. As the range finder moves toward and away from the target, the spot moves side-to-side on the CCD.

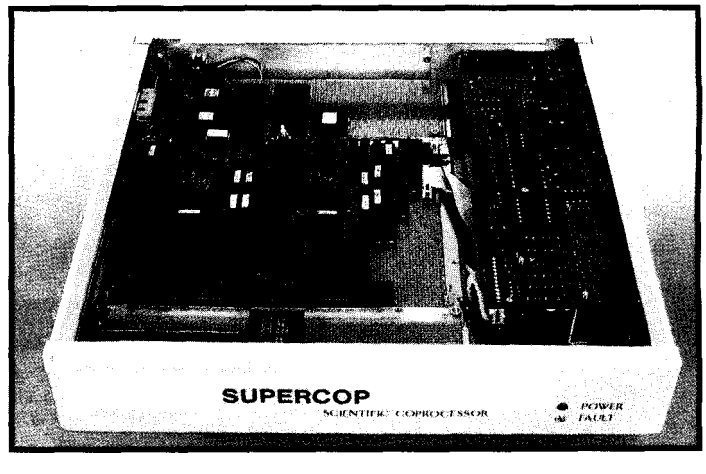
A PIC 16C54 microcontroller converts the data and sends it to a host computer in RS-232 format at 9600 bps. In addition, the PIC allows bidirectional serial communication to allow specifying the sampling rate remotely. Adding ICL776 drivers for the CCD, some voltage regulators, and glue logic complete this low-cost, low-power range finder.

## 2nd place SuperCop — Robert Morrison

Keeping pace with the latest technology, our second-place winner combined the computing power of RISC chips with Field Programmable Gate Arrays and SIMMs to introduce SuperCop, the Super Scientific PC Coprocessor.

This unique and powerful system is capable of double-precision integer or floating-point computations, a shortcoming of present PCs. Expandable to have multiple processors, the SuperCop easily attaches to existing PCs and resides in a separate PC-sized box. As a result of using FPGAs and RISCs in his design, David has kept cost and power dissipation to a minimum.

What makes the SuperCop even more flexible is its potential applications. One example emphasizes the importance of the SIMM memory array and how it is shared by all the processors. As Virtual Reality becomes the newest trend in computer-oriented circles, so too does the technology. Using SuperCop's memory in a graphics application to store 3D information is possible. Processes such as retrieving data and performing intercept checks as well as rendering that same data for 3D perspective transforms can occur simultaneously.

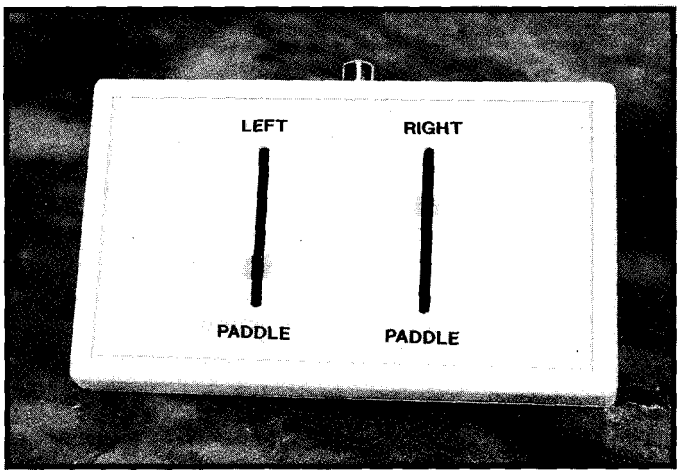


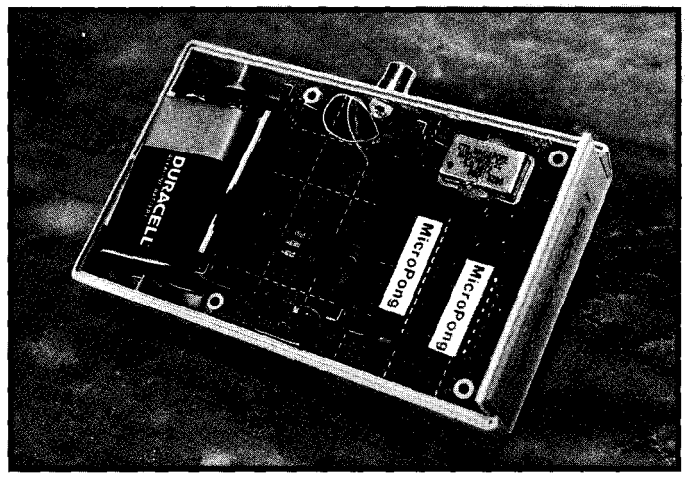
## Micro Pong — Kevin Rouvierre

3rd place

Do you remember your first home video game system? It consisted of a big bulky console with two potentiometers on either end known as "paddles." The name of the game was PONG! and every kid on the block claimed to be the undisputed champion.

A visit to the past in order to relive those old times truly reveals how technology has changed. This time the game is Micro Pong and the objective is the same. However, instead of lugging a large console into your living room and trying to figure out which cables go where, you can stuff




this module into your shirt pocket. One 75Ω cable for connection to an NTSC monitor and you're off and running.

The brains behind Micro Pong is Motorola's MC68HC705, which Kevin received as part of a kit offer last year. Video sync pulses are interrupt driven and software generated by the microcontroller. Video RAM is not needed since the video image is created in a line-by-line, moment-by-moment fashion by the CPU. A PAL22V10 renders the paddles and ball on two separate fields of the interleaved video signal. All this using just 512 bytes of EPROM and 32 bytes of RAM (with only 14 bytes used) on a 16-pin micro leaves one wondering how large this unit's cousin will be in 20 years!

# Honorable Mentions

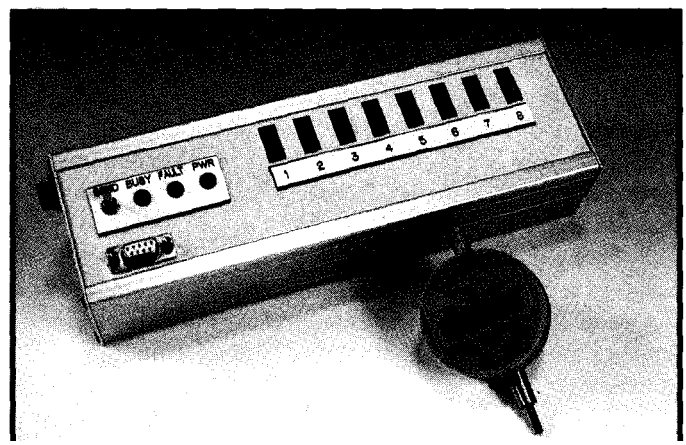
## Indicator Interface — Weston Bye

Most design projects are born out of ideas with a specific application in mind. The Indicator Interface is no exception. The Mitutoyo Indicator is a digital readout interface that uses a miniature linear encoder, MOS circuitry, and a liquid crystal display for precision measurements capable of 0.0005" resolution,

Designed for use in a statistical process control (SPC) environment, this project is Z80-based with 2K of EPROM and 4K of RAM socket space.

An 8255 Programmable Peripheral Interface is used as the bus interface to the Z80 while an LM3900 quad comparator interfaces clock and data lines from the Mitutoyo indicators to the microprocessor.

A channel board is capable of connecting up to eight indicators for data output. Three channel boards connected in parallel with the interface board complete the system.



## Aero-Pix APS — Ken Pergola

Kite flying has been a pastime for many since they were young. Some have even wondered what it would be like to have a bird's eye view of the scene below a kite. For Ken, his dream of aerial photography came true as he realized the advances technology has made on cameras.

The Aero-Pix APS (Aerial Photography System) uses a PIC 16C71 microcontroller for controlling a Canon 35mm camera. It can be lifted into the air using a kite or helium balloon.

Incorporated into the Aero-Pix APS is the programmer board (with a 2-line by 20-character dot-matrix LCD module). It serves as a user interface for easy programming, which involves a Set key and an Enter key. These permit the user to select any number of pictures (1-36) to be taken. Initial time delay before the first picture as well as time delay between pictures range from 1 to 99 minutes.

Once programmed, the main board and programmer board are separated, with the main board being what is



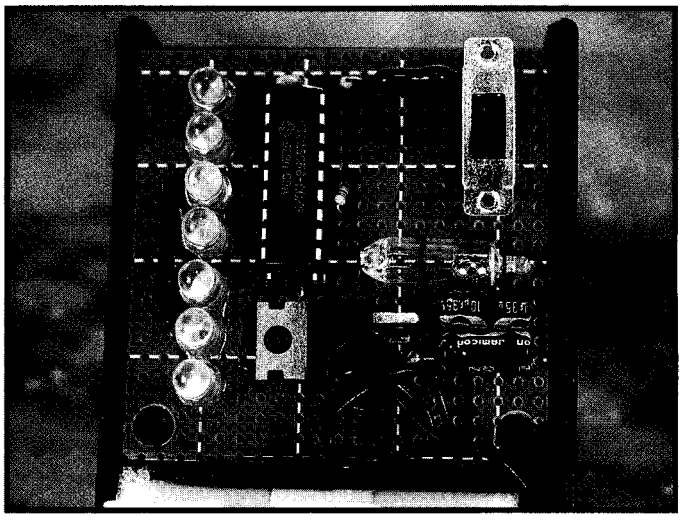

airlifted. The only parts needed in addition to the PIC 16C71 microcontroller are a 4.096-MHz crystal, a 5-volt regulator, a 9-volt battery, a power switch, and the camera-triggering electronics and camera connector.

## Wavy Sign — Jack Torrey

Capping off this year's winners is a small battery-operated toy. The Wavy Sign incorporates a PIC 16C52 microprocessor and a simple accelerometer for sensing movement of the box.



Using your eyes' persistence of vision, the Wavy Sign "paints" a message on your retina as it is quickly moved side to side in front of your face. A vertical line of seven bright LEDs flickers at just the right rate to scan out letters that have been programmed into the PIC. A mercury switch detects which way the unit has been moved so



that the letters appear in the correct orientation all the time. Up to ten letters may be displayed at once.

### Special Mention: Hot Shots

In an effort to encourage young people to study hard and strive to be successful, we would like to acknowledge the submission of one particular entry to our contest. Although she was not a winner, sixth grader Melissa Stovall (with a little help from her dad, Mike) designed a human reaction tester known as Hot Shots.

Using the Motorola 68HC705 microcontroller, this device is capable of detecting the reaction times of up to four people to within one millisecond. It is portable and battery-powered with the capability of displaying the results to its user(s).

A job well done, Melissa. We hope to hear from you in next year's contest!

### For More Information...

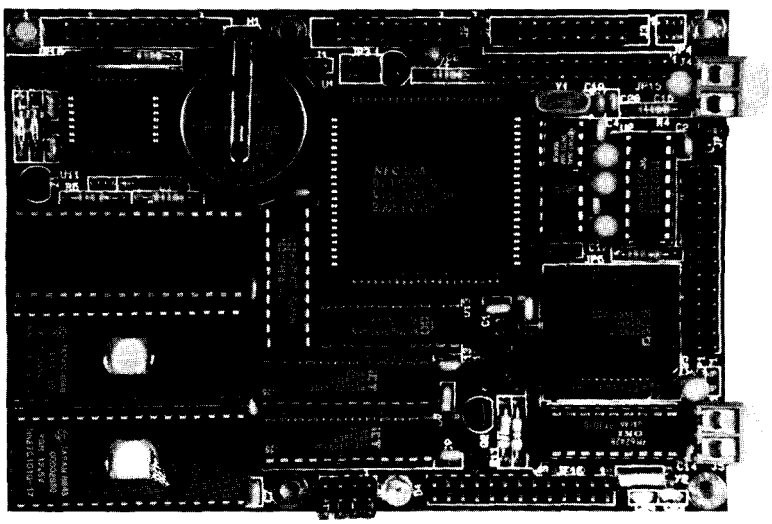
As in years past, we encourage all our Design Contest winners and entrants to write complete articles about their projects. We always highlight our Design Contest articles with the logo you see throughout this spread.

If you would like more information about any of the projects you see here, we're going to have to ask you to be patient and wait for the full article to appear in the coming year. In the interest of preserving privacy, we will not give out addresses or phone numbers for project designers.

If you absolutely must get in touch with one of them and can't wait for an article, you may send a letter to the designer in care of us here at the magazine and we will forward it to the designer. Send letters to Design Contest Winner, The Computer Applications Journal, 4 Park St., Vernon, CT 06066. Thank you for your cooperation.

# V25 POWER COMES TO EMBEDDED CONTROL!

Micromint's RTCV25 is the perfect marriage of an 8088-compatible processor, programming convenience, and control I/O. Forget the need for cross-assemblers and cross-compilers; your favorite high-level language for the PC and a "locate" facility are all you need. The RTCV25 enhances the V25's power with parallel I/O; A/D conversion; RS-232 and RS-485 serial ports; up to 384K of RAM and EPROM; a battery-backed clock/calendar; 128 byte EEPROM, ROM monitor, and the RTC stacking bus. Ease of code development combined with its small size and low power consumption make the RTCV25 ideal for all embedded control applications. And of course the RTCV25 is compatible with Micromint's full line of RTC peripheral boards and products.



**Features:**

- 8 MHz CMOS NEC V25 processor
- Up to 384K of RAM and EPROM
- Two serial ports
- Battery-backed clock/calendar
- 128 byte EEPROM
- 40 parallel I/O lines
- 8-channel, 8- (or 10-) bit ADC
- RTC stacking bus
- Small 3.5" x 5" size
- 5-volt only operation (150mA max.)
- Full featured ROM monitor
- ROMable operating system

100 Qty.
OEM configuration **$270**

MICROMINT, INC.
4 Park St. . Vernon, CT06066
**call 1-800-635-3355**
(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!*

#119

# EXPRESS CIRCUITS

**MANUFACTURERS OF PROTOTYPE PRINTED CIRCUITS FROM YOUR CAD DESIGNS**

**TURN AROUND TIMES AVAILABLE FROM 24 HRS — 2 WEEKS**

**Special Support For:**

- **TANGO. PCB**
- **TANGO SERIES II**
- **TANGO PLUS**
- **PROTEL AUTOTRAX**
- PROTEL EASYTRAX
- smARTWORK
- HiWIRE-Plus
- HiWIRE II
- EE DESIGNER I
- EE DESIGNER III
- ALL GERBER FORMATS

- **FULL TIME MODEM**
- **GERBER PHOTO PLOTTING**

**WE CAN** NOW WORK FROM YOUR EXISTING ARTWORK BY SCANNING. CALL FOR DETAILS!

*Express Circuits*

1150 Foster Street • P.O. Box 58
Industrial Park Road
Wilkesboro, NC 28697

Quotes:
1-800-426-5396
Phone: (919) 667-2100
Fax: (919) 667-0487

#133

## DEPARTMENTS


52 Firmware Furnace

60 From the Bench

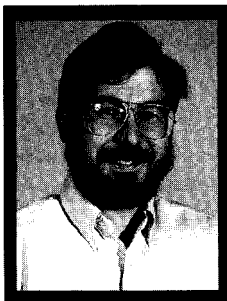
64 Silicon Update

72 Embedded Techniques

77 Patent Talk

84 ConnecTime


# Beyond Small: Mainline C for the '386SX Project



If you've been waiting patiently to use your copy of Borland or Microsoft C with your embedded '386SX, your wait is over. Ed finally takes the plunge and brings on the big guns.

## FIRMWARE FURNACE

**Ed Nisley**

Does your application suffer from Creeping Featuritis? The symptoms include a README file bigger than the program, multiple layers of nested functions, and a tendency to display odd behavior should you dare use the new features. You'll know it when you see it.. ..

One of the (few) nice things about 8031 projects is how the CPU's limited address space puts an upper limit on complexity. Apart from obvious perversions like the two-megabyte kludge I mentioned a while ago, you can only do so much with a single-chip microcontroller. Not so with '386SX systems, I fear, as they are really PCs in controller's clothing. This month's topic may herald the end of innocence.

Thus far I've used Dunfield Development System's Micro-C because it's admirably suited to embedded projects. The startup code is a few lines long, it has simple memory requirements, and, best of all, it works! Version 3 adds structures, unions, and sundry other improvements, so it's even better than before. Dunfield now includes a diskette of video, serial, joystick, and other library routines to get you going right away.

But a recurring question on the Circuit Cellar BBS is, "So, what about our Borland and Microsoft C compilers?" It turns out that things are much more complicated than you'd expect.. .

Unlike Micro-C, the Borland and Microsoft C compilers grew up with the PC. By necessity, they include the assortment of memory models needed to use the 80x86's segmented architecture. Whether these convolutions are a Good Thing is open to debate, but the

fact of the matter is if you want to crawl into the PC's hidden spaces, these compilers help you do it.

There is a catch: mainline PC compilers produce code that *must* run under DOS. Loading their EXE output files isn't easy, their startup code goes on for pages, and DOS functions are scattered throughout the startup and library code. Producing a stand-alone program for a DOS-less machine is not for the faint of heart.

Although several authors have tackled the subject, my opinion is that it's an unrewarding and endless task. Each new version of the compiler ripples changes through your existing applications, so you'll spend quite a bit of time and effort keeping up with the Borlands and Microsofts. Basically, all you want to do is sprinkle magic dust on your EXE file to turn it into the right stuff.

In this column, I'll describe some of the issues involved in producing DOS-less projects with a DOS compiler just so you know what you're missing. I'll also introduce LOCATE, a commercial product from Paradigm Systems, that solves this problem. Paradigm has a low-cost demo version of LOCATE available that will get you started with mainline embedded C.

Paradigm has allowed me to post a customized version of their TDREM program on the Circuit Cellar BBS. If you've built a Firmware Development Board with the battery-backed RAM, you can now debug your Borland C code with Turbo Debugger.. .for free! Of course, they're hoping you'll buy their full version so you can customize the interface for your other projects.

## ALONG THE MAINLINE

A friend is just now starting to write PC code after years of doing large systems software. As he put it, "C is C, but what the heck is all this about NEAR and FAR pointers? When do I need a Large memory model? What's going on here anyway! Who's responsible for this outrage?"

As long as you don't need more than 64K bytes, PC programming is easy because the familiar one-segment COM file format is entirely adequate. As you've seen, a COM file is just a binary image of the program's code and initialized data. Running the program is as simple as copying the disk file into RAM, setting the segment registers, and passing control to the first instruction.

| Offset | Description |
|---|---|
| 0000 | Signature: MZ |
| 0002 | File length modulo 512 (remainder) |
| 0004 | File size in 512-byte units (rounded up) |
| 0006 | Number of relocation table entries |
| 0008 | Header size in paragraphs |
| 000A | Minimum RAM needed in paragraphs |
| oooc | Maximum RAM needed in paragraphs |
| 000E | Location of stack segment in paragraphs |
| 0010 | Initial SP value |
| 0012 | Checksum (not used!) |
| 0014 | Initial IP value |
| 0016 | Location of code in paragraphs |
| 0018 | Offset of relocation table in bytes |
| 001A | Overlay number |
| 001C | Optional reserved space |
| | Relocation table |
| | Optional reserved space |
| | Code and data segments |
| | Stack segment |

Figure I-Programs *stored as EXE files have a header with additional information needed to load and run the program. The exact format depends on the DOS version, but older files will generally work on newer systems. A "paragraph" is 16 bytes.*

But when you need lots of code or data, the Intel Segmented Memory specter rears its ugly head. Because the COM file has no way to specify multiple program segments, the DOS designers came up with the EXE file: not only a new file format, but an entirely different way to load files.

Here's the catch: programs with multiple segments *must* be stored in EXE-format files. Even something as simple as a Small model program with code in one segment and data in another requires the full EXE file structure. Worse yet, because of the additional segment information, EXE files for anything other than Tiny model programs cannot be converted into COM files.

Figure 1 shows the fields in an EXE file header. The "Code and Data Segments" field contains the actual instructions and variables created from your source code, similar to the stuff that's normally in the COM file. The remaining fields are all overhead information needed to load those segments into RAM, adjust them for the actual memory addresses, then set up the CPU registers, and finally start the program.

The compiler and linker produce the EXE file, but they cannot assign the final segment addresses because the load address isn't known until DOS allocates memory just before reading the file from disk. The "Relocation Table" entries point to the values within the code and data that need adjustment before execution. In effect, the DOS program loader performs the final link step needed to put a relocatable program at an absolute address.

For example, the C statement `++Variable` might compile into `INC Variable`. Implicit in that `INC` instruction's operation is the DS register, which must hold `Variable`'s segment address. But the actual DS value depends on precisely where `Variable` wound up when DOS loaded the file. If your program accesses more than 64K bytes of data, DS must match the current `Variable`, so just loading it once at the program's start won't work.

Our embedded programs are definitely *not* relocatable. In fact, most of our code must load and execute at a specific address: if our BIOS extension doesn't start at C800:0000, it simply won't work. The EXE file format's flexibility turns into something of a liability in our situation.

So we need a way to assign absolute segment addresses, then convert the EXE file into an EPROM image or a binary disk file. That's part of what LOCATE does for a living, but there's more to the story.

## JUMPING THE MAINLINE TRACK

By now you should be familiar with the C startup code that gets control before your `main()` function. Although Micro-C's startup code is normally only a few lines long, you can modify it to do whatever you need. With a little tweaking, you can even turn an ordinary C program into a BIOS extension, as I did last month.

The Borland and Microsoft startup code is considerably more elaborate than what we've seen so far. In fact,

the Borland C++ 3.1 startup file spans 746 lines and handles everything from clearing variables and initializing the file system to parsing the DOS environment. Quite a bit of that effort does us no good at all and some of it is actively harmful.

The startup code need do nothing more than the program requires. Because we won't use the DOS file system, all that code can simply Go Away. Similarly, with no DOS environment or command line information, eliminating that part of the startup routine is easy.

Other sections are more difficult to discard. The startup code uses DOS functions to adjust the stack and heap, return excess storage to the operating system, save and install interrupt vectors, and even display error messages. Each of these functions must be replaced with your own code: running a C program without a heap, for example, just won't work.

On top of this, the startup code handles such minutia as defining the proper segment order for the linker, calling C-level startup and exit functions, and so forth. Some of these are tightly tied into the way the compiler generates code, so well-meaning but misinformed changes can wreak subtle havoc on your program. The requirements of C++ add another layer of complexity above what you would expect for a C program, too.

The good news is that LOCATE includes a sample startup code file that you can use unchanged. Comparing the two files tells me I'm glad I didn't have to find the changes on my own!

The last vestige of DOS dependence hides in the runtime library. Many of the routines use DOS calls, but unless you have the library source code, you cannot tell which routines are dangerous. Because the library uses software interrupts to invoke DOS, the linker will not tell you you're using a nonexistent function; your code will simply crash when the interrupt branches into the swamp.

The LOCATE installation includes a program that deletes all the DOS-dependent routines from the normal Borland C runtime library files. If the linker complains about missing functions in your program, well, you'll have to write some additional code, which is considerably better than debugging a mysterious failure.

Now we have all the pieces in place: absolute address assignment, a tailored startup file, and a purified runtime library. The rest is mostly a matter of turning the crank.. .

Listing I-One *of the sample programs* included on the Paradigm LOCATE disk is *the good old SIEVE benchmark. I've modified it to display the iteration number in raw binary on the Firmware Development Board's LED display so you can watch it run. The key* point is *that* this doesn't *look any* different *than any other C code: all the hocus-pocus is handled under the covers by the startup code and LOCATE.*

```
#define TRUE 1
#define FALSE 0
#define SIZE 30000

#include <dos.h>

char flags[ SIZE + 1 1;

void main(void)

   unsigned int i, k, count, loops;

   loops = 0;

/* Run the Sieve forever */
   for (;;){
/* Perform the initialization */
        count = 0 ;
        for (i = 0: i <= SIZE; i++)
           flags[i] = TRUE ;

/* Run the Sieve */
        for (i = 2; i <= SIZE; i++){
           if (flags[i]){
/* Cancel out all multiples of this prime */
                for (k = i + i; k <= SIZE; k += i)
                     flags[k] = FALSE ;
                count++ ;


        outport(0x031e,~loops++);
```

## EMBEDDING A SIEVE

Listing 1 shows Paradigm's version of the venerable Sieve benchmark program. I've added a line displaying the loop counter on the Firmware Development Board's LEDs and increased the sieve test limit to 30,000 integers, but I have not verified the program gets the right answers.

The code looks just like any other C program you've ever seen, because all the magic needed to "embed" it is done elsewhere. The #include <dos.h> line might be worrisome, but it defines the outport macro I used. Fear not, you don't need DOS to run it!

Compiling and linking the file proceeds normally, with two exceptions: you must link with the special startup and runtime library. The result of the link is, as usual, an EXE file. To indicate the special contents, the Paradigm MAKE file forces the file extension to ROM instead of EXE. This prevents DOS from running the program if you should accidentally type it at the command line.

In our case, the outport macro becomes an in-line routine, so Sieve doesn't use any library functions. Incidentally, the LOCATE installation routine does not modify the original Borland libraries, so you can continue to build DOS programs as usual.

The next step is running LOCATE to translate SIEVE.ROM. LOCATE can produce several output formats with an array of options. Among the choices

are Intel hex, extended hex, Tektronix hex, a special "absolute" EXE format, Intel's 0MF86, and raw binary.

For most embedded systems, you would burn a HEX file into an EPROM, but a slightly modified version of the diskette boot loader we've been using so far will handle the new files. The only change involves the'program's initial CS:IP value: the loader must pass control to the program correctly!

Our loader follows the DOS COM file convention of passing control to the program at offset 0100 in the segment indicated by CS. All COM files expect that starting value because there is no way to specify any other register contents. Although EXE file headers include the program's initial IP value (see Figure **1**), the binary file produced by LOCATE no longer has that header information.

The C startup code expects to get control at label `_startup`, which could be anywhere in the code segment. We must ensure that LINK positions the startup file at the beginning of the code segment, then tell LOCATE to put the code segment first in the output file. Because `_startup` marks the first instruction in the startup code, it will become the first byte in the final binary image.

However, the startup code expects II? to be 0000 at the first instruction rather than 0100. Our COM file loader won't work with LOCATE's output!

So the BBS files this month include a slightly modified BOOTSECT that loads the program at 10000 rather than 11000, then jumps to the first byte with CS:IP set to 1000:OOOO. The Paradigm startup code handles all the segment register initializations, but I left the old COM-style setup in place so the registers would at least have a known value.

You might think twiddling the startup code so the offset of the first instruction is 0100 (perhaps by moving other instructions ahead of it) would let you use the old loader. While that might seem OK, remember that the old loader set CS:IP so the *first* byte was addressed at CS:0100. The loader will transfer control to that byte, even though it's not the entry point.

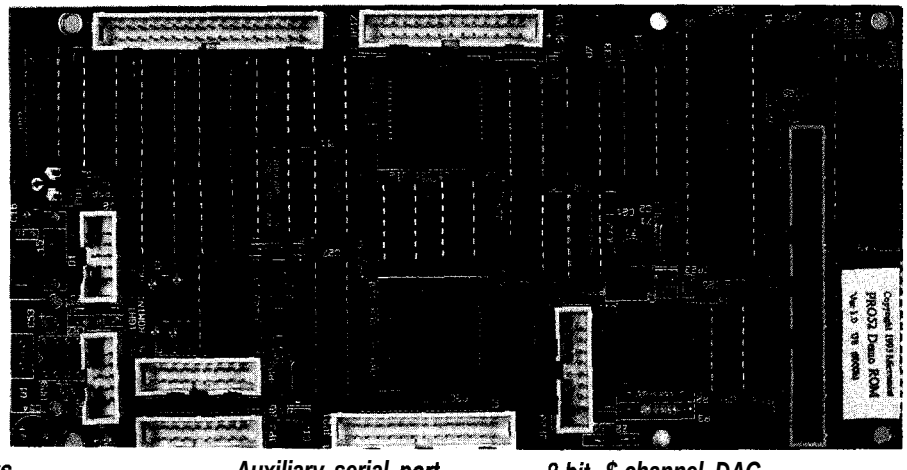
While you could add code to "catch" the boot loader entry and reload CS and II?, my feeling is you'll use either Micro-C's COM files or Borland/Paradigm's binary files. Set the loader up to match your situation and be done with it. Just make sure you mark the diskettes while you're experimenting because the wrong combination just won't work.

With that as background, the addresses I used in the `SIEVE.CFG` file shown in Listing 2 should be comprehensible. LOCATE has far more options than I have room to describe, but you should get a feel for the program's flexibility.

The `HEXFILE` statement specifies an 8K-byte binary output file containing the information starting at address 10000h, which is the load point set by our diskette boot loader. You may use additional `HEXFILE` statements to


**PRO = Positively Rampant Optimization**

*4 × 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

*Power inputs*
*– Single supply or multisupply option*
*Console serial port*
*Auxiliary serial port*
*24-bit parallel TTL I/O*
*Hardware watchdog*
*8-bit, $-channel DAC*
*8- or 10-bit, &channel ADC*
*EXPANSION HEADER*
• *Port 1 processor lines: 8-bit TTL*
• *Six decoded I/O address strobes*
• *Eight address lines*

**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!*

#119

create multiple output files&ring the same pass. LOCATE will warn you if the file is empty, which can catch stupid addressing mistakes.

The MAP statements define how the CPU's address space will be used. LOCATE verifies that all of the segments are appropriate for their addresses: you should not put a code segment into a read/write area, for example, because the code might be overwritten.

The **MAP** option is more useful with the TDREM debugger I'll discuss shortly. Because we are loading the code from disk into RAM, the addresses defined as read-only are really just as volatile as any other RAM. The **MAP** statement cannot write-protect the RAM.. .it just documents your intentions and verifies that no other segments encroach on the code.

The **DU P** statement creates a copy of the data segment in the **ROMDATA** part of the code segment. The C startup code copies **ROMDATA** from the code segment, where it was loaded from diskette, into the data segment. That's all it takes to get initialized variables working!

In last month's column4 wrote a little utility program that massaged Micro-C's HEX output file for exactly the same function: relocate the initialized variables into the code space. One statement in a configuration file is a lot easier, and LOCATE allows you to handle the multiple segments that will crop up in a complex application.

The two C **LASS** statements specify the starting address for the code and data segments. The name CLASS comes about because the compiler and linker can combine several different program segments into a single physical segment in memory; this statement sets the address of the first segment in the resulting class. If your application has multiple classes you need a C **LASS** statement for each one.

The 0 **RD E R** statements specify how to combine individual segments into an overall class that will be nailed down by a CLASS statement. This corresponds (roughly) with the way the linker handles groups of segments, but requires manual intervention to get it right. The LOCATE documentation goes into more detail on this issue. Suffice it to say that the sample files give you a good idea of what to do.

Despite the seeming complexity, after you get a configuration file set up you won't need to tweak it unless you change your program's segment structure or the target system's memory map. In fact, you can probably use the same CFG file for most of your projects with little change.

Actually running LOCATE is just a matter of adding another line to your MAKEFILE; it requires no more thought than your compiler or linker. In fact, I've added the code to copy the resulting BIN file to the boot disk, too, so two keystrokes rebuild the whole application and set up the diskette.

Although SIEVE is compiled with the Small memory model, LOCATE and the modified runtime libraries support all models except Tiny. The diskette boot loader will handle binary files up to 64K bytes, so if you have gigantic programs you must modify the loader. You can surely use my source code to get started.

You can use Borland's Tiny memory model to generate COM files without running LOCATE. Remember to eliminate all DOS functions from the startup code, link with a DOSless library, and boot with the original (IP = 0100) diskette loader.

LOCATE supports Microsoft C as well. I don't have any examples because, for some reasons known only to Microsoft, it doesn't run under OS/2 without some twiddling that I'm not interested in doing. Sigh.

The only question left is, "What if it doesn't work?" Finding errors is always difficult, but there is a better way than burn-and-crash debugging.

## DEBUGGING BY WIRE

Creeping Featuritis afflicts all PC programs. Simple command-line compilers accrete features, sprout program development utilities, and gather debuggers onto their disks. More recently, these messes gelled

**Listing 2**—*The information in this configuration file fells LOCATE how to convert SIEVE.ROM from EXE to binary. The MAP statements define the system address space, while the CLASS statements set the segment starting addresses. The DUP statement creates a copy of the initialized variables in the code segment so the startup routine can copy them to the actual data segment in RAM.*

```
hexfile binary offset=10000h size=8      // boot loader bin-file

listfile segments                        // output listing file

map 0x00000 to 0x005ff as reserved       // Int vectors BIOS data
map 0x00600 to 0x0ffff as reserved       // Unused
map 0x10000 to 0x1ffff as rdonly         // 64K application boot
load area
map 0x20000 to 0x9ffff as rdwr           // Available for app
map 0xA0000 to 0xfffff as reserved       // Video, BIOS ROMs, etc

dup     DATA ROMDATA                     // Copy initialized data

class   CODE = 0x1000                    // Boot loader area
class   DATA = 0x2000                    // Data area

order   DATA                    \        // RAM organization
        BSS BSSEND              \
        STACK

order   CODE                    \        // ROM organization
        FAR-DATA  ENDFAR_DATA   \
        INITDATA EXITDATA       \
        ROMDATA  ENDROMDATA

output  CODE                    \        // Output classes
        FAR-DATA  ENDFAR_DATA   \
        INITDATA EXITDATA       \
        ROMDATA  ENDROMDATA
```

into Integrated Development Environments delivered on a dozen or two high-density diskettes. If you look hard, you can actually find the compiler hidden in a subdirectory and described in a footnote.

Creeping Featuritis means more than never buying another diskette. Along the way, we lost track of the fact that an IDE spawning a debugger to load a nontrivial program compiled with all the debugging options left essentially no room for data. In some cases there isn't enough room for the program! Thus began the arms race to exploit expanded memory, extended memory, any memory but the precious 640K in DOS's territory.

Borland's Turbo Debugger designers, in either a stroke of genius or an admission of defeat, included a "remote debugging" mode that put their famed user interface on an entirely separate machine from your program. A serial cable eliminates the need for exotic RAM mapping, video display sharing, and mouse handoffs.

Whenever Turbo Debugger refers to a memory location, an I/O port, or a CPU register, it sends a message across the cable to the target system. A debugging kernel handles the request and returns the results over the same cable. Because the kernel doesn't interact with the keyboard, video display, or mouse, it can be both small and simple. Most of the conflicts between the program and the debugger simply Go Away when the two programs run on different systems.

The folks at Paradigm (and others, as well) figured out that this was ***the*** solution for embedded 80x86 systems. With a bit of code magic, you can fool Turbo Debugger into thinking there is an entire PC on the other end of the wire, even if the target system lacks disks, DOS, and even a BIOS.

Adapting TDREM to the Firmware Development Board was straightforward, as the Paradigm code can handle most hardware configurations by just changing a few EQUs or #defines. I elected to use the simplest interface: polled communication using COM1.

Because TDREM should get control before a disk boot, I turned it into a BIOS extension using code similar to that shown in Issues 38 and 40. The extension code hooks Int 19 and returns to the BIOS. When the BIOS is finished initializing everything, it invokes Int 19, which starts the TDREM interface. My code updates the Firmware Development Board LEDs to show its progress, concluding with a cheery "TD" just before diving into Paradigm's Turbo Debugger routines.

The BIOS extension loader I've described before copies TDREM from diskette into the FDB's battery-backed RAM and computes the checksum required of a BIOS extension. Just hold down the push button or close the keyboard lock switch while booting the loader to set up the extension.

After that, running the debugger is easy: boot the system! When the LEDs show "TD" you can start Turbo Debugger on your main system. Use the command line:

```
TD -RS3
```

to use a 38.4-kbps data rate. I found that TDREM's default 115 kbps rate doesn't work on my system. Although


**An Unbiased Survey for DOS Developers.**

☐ **No!**

I don't want to find out how I can save a lot of money using ROM-DOS 5 instead of MS-DOS@ in our 80x86 product line. I don't care if ROM-DOS 5 iscompatible with MS-DOS 5 but costs much less. I like spending much more than I have to. It makes me feel like a philanthropist and besides Microsoft@ probably needs the money more than I do anyway.

☐ Yes, I want to know the facts about ROM-DOS 5. Please send me information and a free bootable demo disk to try with my software.

**In the U.S.A. Call Toll Free 1-800-221-6630**

**or** fax this coupon to (206) 435-0253.

*Name* _____

*Company* _____

*Address* _____

*City* _____ *State* _____ *zip* _____

*Phone* _____ *Fax* _____

307 N. OLYMPIC, SUITE 201 • ARLINGTON, WA 98223 USA • (206) 435-8086 • FAX: (206) 435-0253

Datalight®

MS-DOS and Microsoft are registered trademarks of Microsoft Corporation

#134

I don't have the hardware specs for the integrated I/O card, the fact that it doesn't support the higher rate isn't surprising. After all, the Original PC's hardware wasn't rated for that speed.. .so who would use it?

The TDREM kernel always tells the debugger that its copy of the program is out of date (it doesn't have a copy!), so Turbo Debugger will ask you whether you want to send the "newer" version.. .always answer Yes.

My prologue code checks the FDB's button before starting TDREM so you can still boot normally. If the button is down, control returns to the BIOS without capturing Int 19, which means that the standard diskette boot sequence will occur. The LEDs show a pair of dashes to indicate that the extension is not installed.

Now you can choose Micro-C or a mainline C compiler depending on your goals and finances. Either will work for a range of projects, so there's not much holding you back now!

## THE SECOND WORST HACK

'In Issue 38 I described the Worst Hack in PC-dom: using the keyboard controller to reset an 80286 CPU to bail out of protected mode. While doing some research for a future column, I came across another tidbit that probably qualifies as the Second Worst Hack in PC-dom.

Suppose you're writing a protected-mode operating system (can you spell OS/2?) for '286 systems and realize that the Worst Hack's speed (or lack thereof) will clobber overall system performance. What to do! Easy: crash the system!

Starting with the 80286, the CPU enters what's called *shutdown mode* as a result of severe errors. For example, if the stack pointer is 0001, a PUSH will force the '286 into shutdown rather than wrap SP to FFFF and overwrite RAM outside the stack. The CPU sends a specific status output so the rest of the system knows what happened, then waits for a hardware reset or an NMI.

Rather than leave the system stalled forever, the Original IBM PC/AT included a shutdown detection circuit that issues a system reset. The '286 pops out of reset in real mode, the BIOS checks the real-time clock's RAM for the shutdown reason code, and vectors to the appropriate routine. Apart from the fact that the reset comes from a different circuit, it works just as I described in Issue 38.

Because all this happens at hardware rather than software (or even firmware!) speeds, the whole reset sequence takes a fraction of a millisecond. That's enough faster than the Worst Hack to make it worthwhile.. .and probably faster than the hyperthyroid keyboard controllers with the hardware bypass, too.

How do you force the CPU into shutdown? The CPU includes a Double Fault interrupt handler that gets control when two protection violations occur on a single instruction. If that handler *also* causes a violation, the CPU shuts down. Rather than force a stack violation, the OS/2 designers set up a deliberate triple fault to bail out of '286 protected mode.

Although I don't know the exact method they used, you could mark a segment as not present, then invalidate both the Segment Not Present and Double Fault interrupt gates. You disable interrupts and the NMI input, aim a segment register at the missing segment, and fetch a byte. The fetch faults to Int OB, which faults to Int 08, and the missing Double Fault handler slam-dunks the triple fault.

Thud!

## RELEASE NOTES

The SIEVE code and binary files are set up for the modified boot loader. I've included the source and binary files for all four diskette sizes again, so be sure to keep all the versions separate on your hard disk.

Paradigm's TDREM Turbo Debugger remote driver is a BIOS extension, so copy it to a diskette along with the appropriate boot sector extension loader. Boot the diskette with the button down to stuff TDREM into the Firmware Development Board's battery-backed RAM. The next boot with the button up will display "TD" on the LEDs when it's ready for Turbo Debugger.

This version of TDREM is configured for polled operation on COM1 at 38.4 kbps. Remember to start Turbo Debugger on your real system using –RS3 to select remote debugging with the right data rate! All the usual features are available except breaking into a running program; that requires a fully interrupt-driven configuration.

Next month I think I'll be able to start the long-awaited Graphics LCD interface. The trick here is using firmware instead of a fancy LCD controller.. .those cheap surplus LCDs aren't useless any more!

*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.*

## SOURCES

Paradigm Systems offers a $95 demo version of their LOCATE program. Contact them at 3301 Country Club Rd., Ste. 2214, Endwell, NY 13760, (607) 748-5966, Fax: (607) 748-5968.

The demo restricts programs to 16KB of code, which is adequate for small projects. If you need Turbo C, you'll need the full version.

Check the ads for other companies that supply similar products. If your project needs file I/O and other DOS functions, look for "embedded DOS" products that can save you a *lot* of work. There's also support for Microsoft's Codeview if you need it.

Pure Unobtainium has the complete Firmware Development Board schematic, as well as selected parts. Write for a catalog: 13 109 Old Creedmoor Rd., Raleigh, NC 27613, phone or fax: (919) 676-4525.

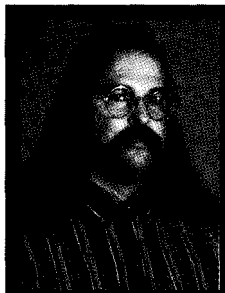
## I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

# Measuring Up an Electronic Caliper Interface

## FROM THE BENCH

Jeff Bachiochi



If you were anything like Jeff was as a child, you probably have your own stories of taking apart everything in sight. Jeff's a few years older now, but that hasn't stopped him. Tag along on his latest excursion.

I started to put together my own tool set by the time I was ten years old. I started with the basic essentials: I had a screwdriver, a jackknife, and a hammer. Hardly a set of tools by today's standards, and they didn't belong to me. They were really "Dad's tools" which I had managed to swipe, and I had them hidden in a secret nook in the dirt floor basement. Why did I hide them? Because of my addiction. Maybe you were hooked at an early age like me. I'm a curiosity junkie.

In those days, Dad's tools were locked up. I had operated on Mom's appliances too many times without the skills necessary to reverse the procedure. If I hadn't broken something in trying to open it, my Dad would try his best to recover from my mishaps. He actually got pretty good at fixing stuff, but I realize now that's *not* the kind of thing a father looks forward to after a tough day at the office.

I would look forward to trash day, because each week a gold mine of things for me to investigate would appear along the neighborhood curb. Radios, motors, clocks, and practically anything else mechanical or electrical was ripe for the plucking. And the best part was no one became angry if it never was reassembled correctly.

With all this hands-on training, it wasn't long until the tables turned. My room slowly mutated into a museum for successful resurrections. Parental communication permuted from "If it ain't broke, don't fix it," to "What are you going to do with all that junk?" That "junk" was how I got my first radio, record player, and even my first tape recorder. The profits to be gained from a disposable society were high, at least for a kid with no spending money.

### STILL CRAZY AFTER ALL THESE YEARS

I still have the same urges. I exact a certain pleasure from investigating mechanical and electrical designs. I can also get a certain delight from the way manufacturing processes have changed to open up new possibilities in packaging.

Inspection is (or at least it should be) "job 1" in the manufacturing process. This point is presented most effectively in visual form in one of TV's automobile promotions. Here we see a ball bearing rolling along perfectly matched body panels. The car (seemingly) is tilted (effortlessly) by a powered gimbal allowing the ball bearing to zip along all seams without ever leaving its prescribed route. This all implies the most rigorous of standards in parts tolerances. So much for special effects.

If we can't keep manufacturing within established tolerances, the finished product will certainly be inferior. It is the inspector's responsibility to assure compliance with the prescribed standards. It is an often boring job to scrutinize each part even though the last and next *(100,* 1000, or 10,000) parts appear (on the surface) to be exactly the same. Tools make this job easier (and in some cases possible). What happens when records must be kept of the inspector's findings, but sophisticated recording instrumenta-

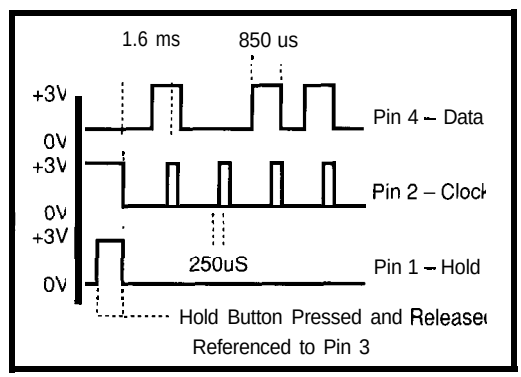



Figure 1—*Pressing and releasing the hold button triggers the clock and data fines.*

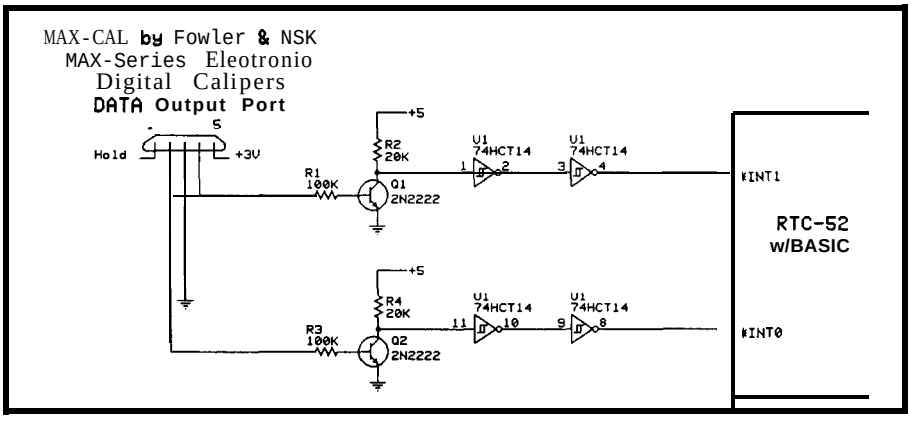



Figure 2-A *pair of common transistors and some Schmitt triggers to clean up noise are all that is required to convert the 3-volt levels from the caliper's clock and data lines to the 5-volt levels of any microcomputer.*

Listing 1—*BASIC code was used to fake fhe raw data and manipulate if to a discernible form.*

```
10  A=4003H                        : REM  INTO vector location
20  READ D
30   IF D=0A5H THEN GOTO 70        : REM 45 used as EOF marker
40  XBY(A)=D
50  A=A+1
60  GOTO 20                        : REM Poke in all data
70  FOR A=4003H TO 4030H           : REM Now check for good data
80    V=V+XBY(A)
90  NEXT A
100  IF (V<>5719) THEN PRINT "No RAM"  : STOP
110 TCON=(TCON.AND.0FDH).OR.1:  REM  INTO edge triggered
120 DBY(18H)=0                     : REM Byte count
130 DBY(19H)=80H                   : REM Bit count
140 FOR Y=0 TO 16                  : REM 17 bytes
150   XBY(4100H+Y)=0               : REM Clear buffer
160 NEXT Y
170 IE=IE.OR.81H                   : REM Enable INTO
180  IF (DBY(18H)>16.) THEN IE=IE.AND 7FH ELSE GOTO 180
190  FOR Y=0 TO 16                 : REM 17 bytes
200    PRINT  CHR(XBY(4100H+Y)),: REM From buffer
210 NEXT Y
220 GOTO 120                       : REM Do it all again
230 DATA 0C0H,0E0H                 : REM       PUSH  ACC
240 DATA 0C0H,083H                 : REM       PUSH  DPH
250 DATA 0C0H,082H                 : REM       PUSH  DPL
260 DATA 090H,041H,000H            : REM       MOV   DPTR,#4100H
270 DATA 030H,0B3H,00FH            : REM       JNB   P3.3,AAA
280 DATA 0E5H,082H                 : REM       MOV   A,DPL
290 DATA 0C3H                      : REM       CLR   C
300 DATA 025H,018H                 : REM       ADD   A,18H
310 DATA 0F5H,082H                 : REM       MOV   DPL,A
320 DATA 050H,002H                 : REM       JNC   BBB
330 DATA 005H,083H                 : REM       INC   DPH
340 DATA 0EOH                      : REM BBB   MOVX  A,@DPTR
350 DATA 045H,019H                 : REM       ORL   A,19H
360 DATA 0F0H                      : REM       MOVX  @DPTR,A
370 DATA 0E5H,019H                 : REM AAA   MOV   A,19H
380 DATA  03H                      : REM       RR    A
390 DATA 0F5H,019H                 : REM       MOV   19H,A
400 DATA 0B4H,080H,002H            : REM       CJNE  A,#80H,CCC
410 DATA 005H,018H                 : REM       INC   18H
420 DATA 0D0H,082H                 : REM ccc   POP   DPL
430 DATA 0D0H,083H                 : REM       POP   DPH
440 DATA 0D0H,0E0H                 : REM       POP   DPL
450 DATA 0D0H,0D0H                 : REM       POP   PSW
460 DATA 032H                      : REM       RETI
470 DATA 0A5H                      : REM End of data marker
```

tion is not available! As they say, Paperwork Happens. But all that paperwork must be handled manually. Part of dealing with that burden is the possibility of transcription errors, which becomes another stumbling block for productivity and reliability.

## MEASURING UP

Recently, I needed to measure the lead size of a new part going into a PC board layout and I could not find my calipers. Rather than dig though the heaps on my desk, I wandered into the production area to borrow theirs. "Wow," I thought, "They have electronic calipers." Interpolating slide rule calipers is not difficult, but it is still a judgment call. Dial calipers increase interpolation accuracy, but with electronic calipers there is no interpolation necessary. Repeatable accuracy (to the 'nth degree) is assured each and every time.

After making the necessary measurements, I noticed a little removable cover on the bottom of the calipers. A quick jab with my fingernail exposed five little pins just below the cover. A smile instantly spread from my mind to my face as small devilish horns appeared from my temples. I needed to know more.

Some level of maturity rose to the surface because the first thing I did was pick up the manual and read it cover to cover. I learned about batteries, button operation, and mode selection. Sure enough, there was a data output mode. However, the particulars were skirted entirely. The only reference to the output format was that the measured value along with the sample number (000-999) would be "printed" upon pressing of the "Ho" button. No mention was made at all of the particulars of the output connector I had found, nor was there any listing of the address or phone number of the manufacturer. "Argh! "

## SERIOUS SPELUNKING

Instead of a hammer, I pulled over my 'scope. The fact that this device is battery operated had an advantage here. I didn't have to worry about where I placed the ground reference.

I examined pins 2-5 with reference to pin 1, then I measured pin 1 and pins 3-5 with reference to pin 2, and so on. Only two of the five pins on the caliper's output port were steady state and independent of any button presses. One of these must be ground. There was a 3-volt difference between the two (ah, the battery voltage). So, I picked pin 3 as ground, which left pin 5 as +3 volts. Now with reference to pin 3, I had to define pins I, 2, and 4.

```
a)                                   b)
11111011111111011011110011...  ...00110000100100000010000 0
11111011111111011101110011...  ...00110001000100000010000 0
11111011111111011001110011...  ...00110001100100000010000 0
11111011111111011110110011...  ...00110010000100000010000 0
11111011111111011010110011...  ...00110010000100000010000 0
11111011111111011100110011...  ...00110011000100000010000 0
11111011111111011000110011...  ...00110011100100000010000 0
11111011111111011110010011...  ...00110100000100000010000 0
11111011111111011011010011...  ...00110100100100000010000 0
11111011101110011111110011...  ...00110000001100010010000 0
11111011101110110111110011...  ...00110001001100010010000 0

11111011101110011111110011...  ...00110000 0011000 1000100000
                                    ASC(30)=0 ASC(31)=1
11111011101110110111110011...  ...00110001 00110001 00010000 0
                                    ASC(31)=1 ASC(31)=1
```

**Figure 3-a)** *Modifying the data by inverting it reveals no identifiable format b) However. reversing the inverted data clearly shows the pattern is 8-bit* ASCII.

Pin **1** is simply a level change dependent on the "Ho" key. The level of this pin rises whenever the button is depressed and falls when the button is released. The remaining pins (2 and 4) also follow the "Ho" buttons release. Timing relationships are shown in Figure **1.**

The data seemed to be in some kind of serial format, but not in the asynchronous format that I am accustomed to. Although it appeared to be synchronous, having both a clock and a data line, the format could have been anything from ASCII or BCD to encrypted. I wagered that it was some kind of recognizable format. But, first I needed a simple interface to go between the 3-volt output of the caliper and TTL levels. Two identical transistor circuits level shifted the signals into their inverted TTL counterparts. A bit of noise on the data line (pin 4) occurred during the clocking pulse's rising and falling edges, so I added Schmitt trigger buffers to eliminate that (see Figure 2).

```
 1     0.0000 I
 2     0.0065 I
 3     0.0505 I
 4     0.0980 I
 5     0.1430 I
 6   - 0.3410 I
 7   - 0.5320 I
 8   - 0.7700 I
 9   - 1.0950 I
10     0.0000 I
11     0.0900 I
12     0.2080 I
13     0.6170 I
14     0.7170 I
15     1.8680 I
16     2.6400 I
17     3.9695 I
```

**Figure** *4-Further* **analysis** *of the* **data** *stream reveals a fixed-format 17-character* **string.**

## CRY PTO-TOOLS

I chose to use the now-infamous RTC52 as an investigation platform. The flexibility of this system is clearly demonstrated here by allowing on-line alterations of a BASIC program to massage and display the collected data in any way imaginable. I'd hoped that by using various ways of presenting what I collected I could ultimately recognize some intelligible patterns.

The clock line is connected to the INTO input on the 8052 and the data line to INT1. Using these inputs allows a falling clock edge to trigger a routine that samples the data line and stores the sample away. The interrupt routine was written in assembler because BASIC's 0 N E X 1 command could not loop as fast as the clock pulses were being sent (1.6 ms). Although this required about two dozen lines of assembler code, it was easily attached to the BASIC program and poked into memory at runtime (Listing 1).

The first BASIC program I wrote simply displayed the captured data in one long stream of ones and zeros. This setup would give me several things: the length of the data transmission, repeatable sampling, and the basis for a correlation between the data and what the caliper's manual suggested as an incrementing sample. Starting from a caliper reset state, I sent several samples (with the caliper jaws closed) to the display platform and noticed that the data appears to be of a consistent length and repeatability. This is a bit of confidence.

Now I turned my attention to the so-called "sample number" that was referred to in the caliper manual. A review of the manual tells of an incrementing sample value from 001 to 999 sent somewhere among the data stream. A short section of the display did indeed change between transmissions, but there was no pattern.

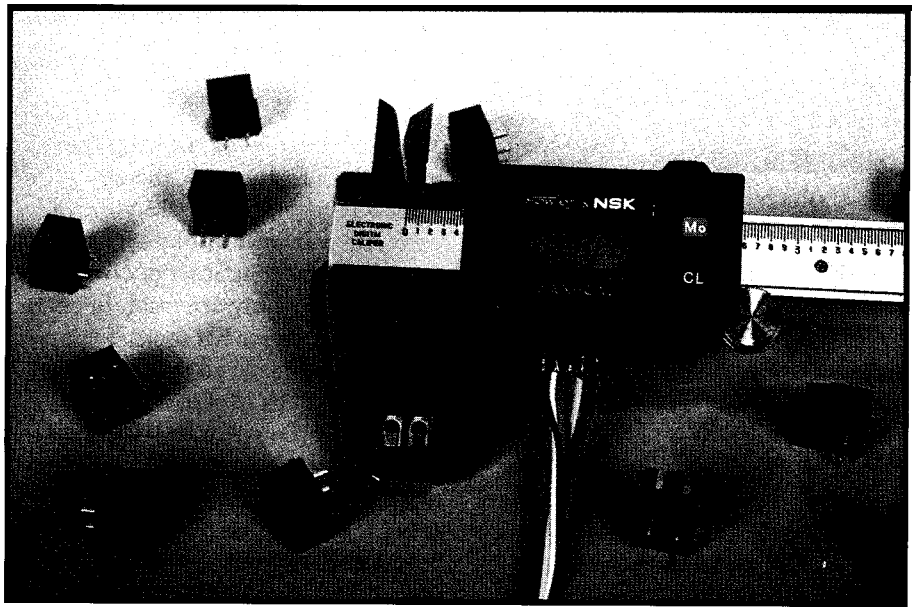
Next, I tried inverting the data by changing just one line of the program. The results are in Figure 3a and still show no pattern. However, changing one more program line to reverse the order proved fruitful (see Figure 3b).

See anything? Notice after nine samples, a new area is altered while the previous area begins to repeat. This looks like BCD except the pattern is not every nybble, instead it changes bits 4, 5, 6, 7, and 8. Could it be simple 8-bit ASCII? Let's see, 00110001 would be the character "0." Bingo!

Assuming the incrementing sample number should change from 9

| Position | Meaning |
|---|---|
| 1 | **Sample number 100s** digit (or **space)** |
| 2 | **Sample number 10s** digit (or **space)** |
| 3 | **Sample number** 1s **digit** |
| 4 | **Space character** |
| 5 | **Spacecharacter** |
| 6 | **Spacecharacter** |
| 7 | **Negative sign (or space if positive)** |
| 8 | **Spacecharacter** |
| 9 | Measurement 1s **digit** |
| 10 | **Decimal point** |
| 11 | Measurement tenths **digit** |
| 12 | **Measurement hundredthsdigit** |
| 13 | Measurement thousandths **digit** |
| 14 | **Measurementtenthousandths digit** |
| 15 | **Space character** |
| 16 | **Mode character** (I=inches, M=**metric)** |
| 17 | **Carriage return** |

**Figure 5**—*The fixed 17-character string format makes decoding on the receiving end very predictable.*




to 10, we can guess (from Figure 3b) that the data stream comes in a format where each digit is sent least-significant bit to most-significant bit, and the character order is right-most character to left-most character. Additionally, there must be some 17 bytes of S-bit data. Manipulating the data stream with these rules and displaying a 17-character string reveals Figure 4. It would seem no further analysis is necessary.

## FRAME OF REFERENCE

By clearing (zeroing) the calipers while the jaws are open, negative measurements are displayed as the jaws are moved toward the closed position. In this mode, the calipers can be used to give plus/minus tolerances with respect to a nominal setpoint. The output string can be formatted in both English and metric systems and indicates the mode as shown in Figure 4.

See Figure 5 for a synopsis of the output string format. The carriage-return-delimited, fixed-length string is easily imported into any number of spreadsheet programs or simply saved to a file for later import or printing. This gives a permanent record of the inspection procedure without ever having to put down the calipers between samples. And, no more transcription errors.

This tool was purchased through a mail order tool catalog and lists for about $100.00. Granted, manual calipers list for less, but look at the advantage of a direct reading data input system. This project could now proceed in a number of directions. The RTC platform could be eliminated by using your favorite flavor PIC, Z80, or 68HC05K-series micro. A true RS-232 or even parallel interface can be strapped to the caliper for less than $20. Or, you can keep the RTC platform and make a completely portable sample logger which you can return to later to download the samples to your PC.

Keep your eyes open and you may find other ways of improving your productivity without having to buy new equipment. Take your time and investigate the possibilities. It might just pay you back twenty-fold. Now, where is that counting scale? I've got my screwdriver and I'll bet.. ❑

*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.*

**IRS**

416 Very Useful
417 Moderately Useful
418 Not Useful

**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 Ctt, 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 µsecs (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 µsec)
- 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 reentrance 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** (TPIBP 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.

On Time 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

#138

# It's Showtime

# SILICON UPDATE

**Tom Cantrell**



Tom gives us a tour of Silicon Valley's hottest conferences and techie shows. Topics ranging from liquid cooling to PC designs to the broad acceptance of PCMCIA are the talk of the town.

Though Silicon Valley has its share of problems, for an insatiable techno-groupie like me, there's nowhere else to be.

I feel sorry for chipheads scattered elsewhere across the continent. Other than your monthly fix of this magazine, things must get pretty grim. "Gee, I guess I'll head over to the Radio Shack and see if they have any new diodes"-Pitiful!

Here in Silicon Valley, we don't go to shows or conferences-they come to us. My calendar is jam packed with so many interesting gatherings that it's hard to choose which scene to make. When in doubt, those shows offering free lunch or beer usually get the nod.

Never fear. As your faithful correspondent, I'll make the rounds so you can experience vicariously the thrill of technology on the bleeding edge (sorry, you'll have to take care of your own lunch and beer). Herewith, reports on three of this summer's more interesting confabs.

## HOT CHIPS V

Shuffling toward the stage, he looks like a hippie who, though somewhat the worse for 25 years of wear and tear, still thinks every summer is the "Summer of Love." The shuffling draws my attention to his feet where I notice his socks don't match, a fact made all the more apparent by lack of shoes. In a voice raspified by too much bad coffee and general lack of use during late nights alone at the lab, he mumbles something about the "vectorists" (or was it "vecterrorists") versus the "schedulers." This should be interesting.. ..

Ah, the good old days.

The Hot Chips conference, now in its fifth year, has definitely gone respectable-lots of suits and ties (not to mention shoes)-and the topics covered have become much more gentrified. Nevertheless, besides great lunch, there is still enough wacky stuff to keep it interesting.

For instance, by now you're probably aware of the creeping "thermal crisis" that is becoming more problematic as transistor counts and clock rates inexorably increase. Witness the fan+heatsink gadgets everyone is sticking on their '486s.

DEC, having taken the lead in the "watt-wars" with their 30-W Alpha, is clearly committed to "hot" chips. They described their latest research project, a 300-MHz ECL micro (ironically of the MIPS variety, an architecture which, since Alpha, DEC no longer pushes) packing a whopping 486,000 transistors into a 504-pin package. Best of all, power dissipation is an astounding 115 watts! OUCH! With "hot spots" on the die approaching 100°C, the major object of the research is how to keep the thing from suffering terminal heatstroke.

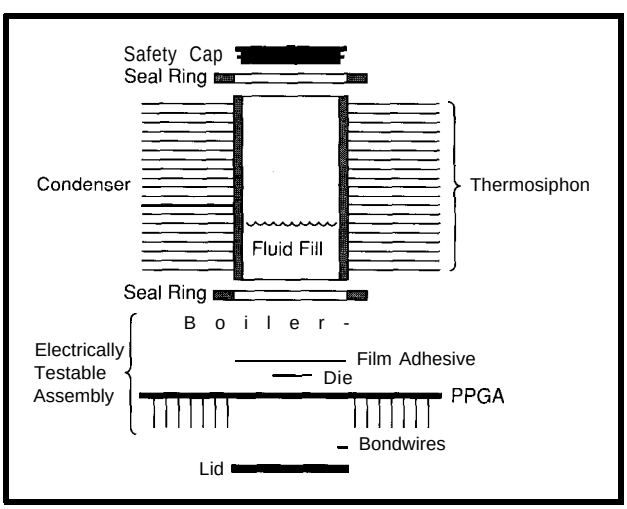



Figure 1—*DEC's thermosiphon* takes *heatsinking* one step further *by using* liquid cooling.

As shown in Figure I, their solution is liquid cooling-shades of mainframes! The "thermosiphon" consists of a boiler (heatsink) glued to the die and attached to a condenser filled with an antifreeze-like coolant. As the coolant boils, the vapors rise to the top of the finned condenser where they cool, return to liquid state, and drip back into the reservoir.

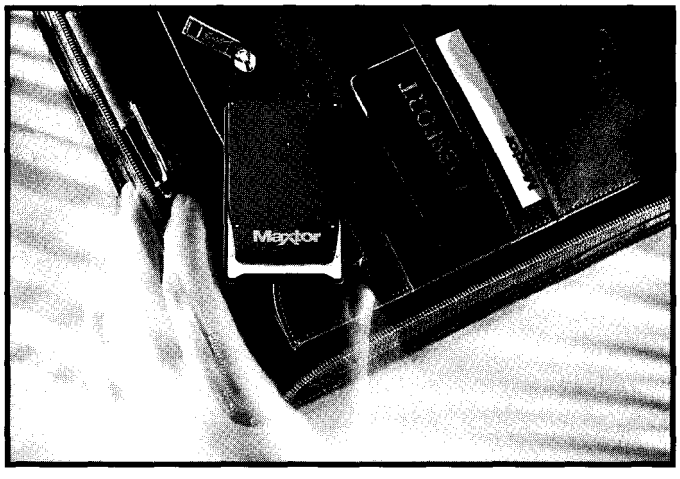



**Photo 1**—*The Maxtor MXL 105-III 105-MB hard disk drive plugs into a type-III PCMCIA card slot.*

The scheme raises a number of interesting questions. First, notice the "safety cap" which, like a radiator cap, conjures up rather ominous images of "boilover." Indeed, it might be wise to include a "DIPstick," just so the overly cautious can check and top-off their coolant each spring.

Furthermore, it turns out the efficiency of the gizmo is enhanced by being tilted at an angle so the turbulence of the rising vapors doesn't interfere with coolant flow back down the side. The presenter pointed out that the principle is the same behind the well-known (or learned the hard way) fact that you tilt a baby when you feed it a bottle-or else. While a baby "burpee" is messy, a chip "burpee" is terminal.

I can see it now-everyone will be propping up a corner of their PCs. But don't go too far. In particular, those who try the somewhat dubious poor man's "tower trick" of flipping the PC on its side had better keep a fire extinguisher handy.

Megatransistor chips have another problem: How do you verify the design before committing to silicon?

In the old days, after drawing a schematic, it was possible to actually build a breadboard with discrete transistors and gates. However, it wasn't long before both approaches—schematic and breadboard-were rendered unfeasible by climbing transistor counts.

Recently, the approach has been to describe the chip's behavior using a hardware description language (HDL), which is synthesized into a gate-level definition much as a C program is translated to binary opcodes. Instead of a hardware breadboard, software breadboarding takes place using a simulator running on a computer.

It sounds great and does work, but not without some pain-namely, the increasingly severe "simulation bottleneck." On the one hand, the model can be simulated at the behavioral level which is fairly speedy, but ignores/hides potentially nasty, low-level timing (race, critical path, etc.) issues. Those can only be found by gate-level simulation, which brings even the most SuperDuper workstation to its knees-we're talking a few clocks per second. Obviously this is problematic since for today's CPUs, a second of real time comprises 30-60 million clocks. Booting DOS on a gate-level simulator could take years! Band-aids include "accelerators" that boost simulation speed and "mixed-mode" simulators that allow gate-level simulation of the portion of the design of interest, while the rest is simulated behaviorally. Nevertheless, simulation is still frustratingly slow.

The latest trend, exploited by Intel during the Pentium design, replaces software simulation with hardware emulation. The concept, pioneered by companies such as Quickturn and PiE (recently merged) is based on the up-and-coming reprogrammable FPGAs (Field Programmable Gate Arrays), specifically the SRAM-based variety.

In the Intel Pentium debug setup, they used actual memory chips wherever possible (cache and microcode), but still had to use fourteen (!) Quickturn RPM boxes (i.e., thousands of FPGAs and millions of bucks].

Despite the high cost, the payback is pretty clear. Intel was able to achieve 300-kHz emulation speeds, run OSes and applications, fix bugs and incompatibilities, and even get a head start on development tools-all before the first wafer start. This is definitely the future of chip development.

More, and faster, transistors are the straightforward way to boost performance. However, the increasing costs associated with pushing the limits are leading to research in new, more efficient circuit designs.

One idea presented by a team from Stanford is called "Wave-Pipelining," which differs from regular pipelining as shown in Figure 2. The challenge is

I

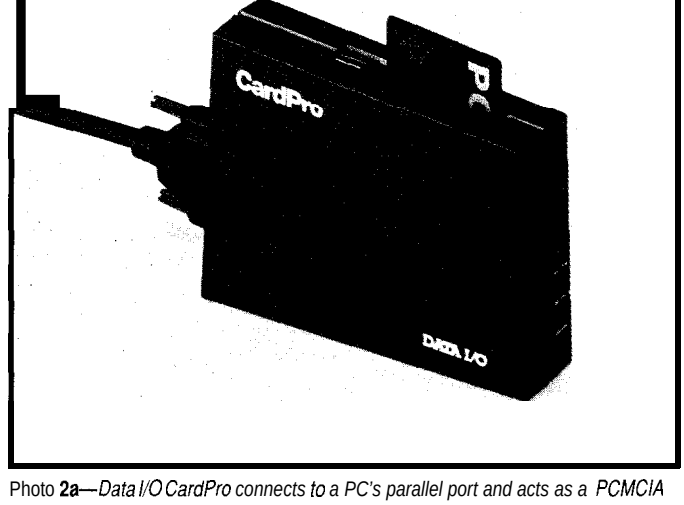



Photo **2a**—*Data I/O CardPro connects to a PC's parallel port and acts as a PCMCIA* card "drive."

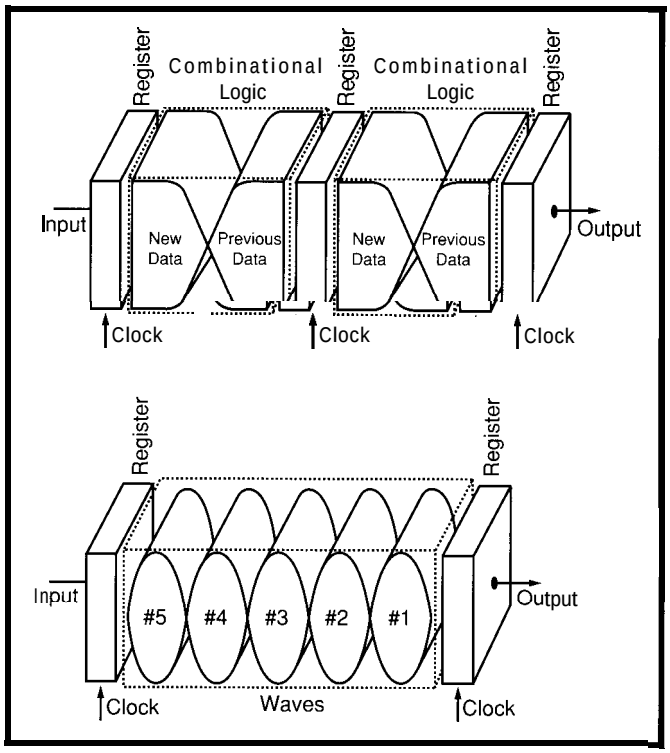



Figure 2-A regular *pipeline (top) versus a wave* pipeline *(bottom). The challenge with the latter is to minimize fhe variation between paths since that determines the speed of a wave pipeline.*

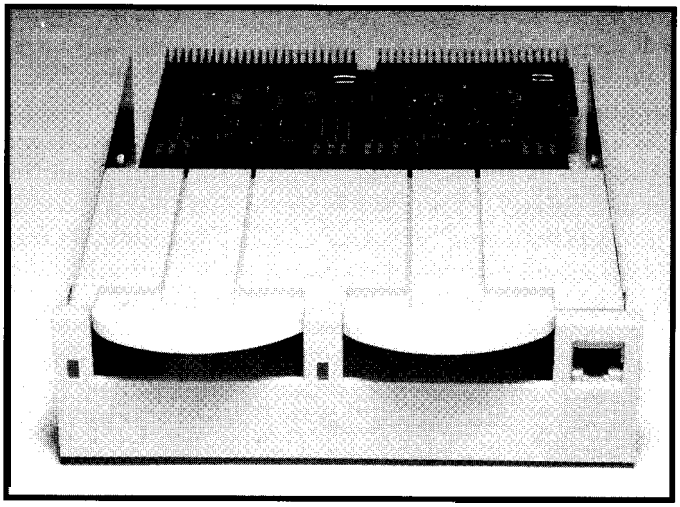


Photo **2b**—*The Greystone CardDock PCMCIA interface installs in a 5.25″ drive bay and comes in one-, two-,* three-, and four-slot *versions.*

to minimize the variation between paths since that determines the speed of a wave pipeline. Unfortunately, the delay of a CMOS gate varies substantially (up to 50%) based on the inputs, not to mention intrinsic path length and capacitive loading differences. However, the delay variation through an entire circuit is less (typically 10–20%) than that of the individual gates since the variance of a sum is less than a sum of variances. The team built a wave-pipelined multiplier test chip that achieved nearly a two-times speed-up over a regular pipeline design, with slightly smaller die area to boot.

Another novel approach, taken by a group from the University of Manchester, abandons today's reliance on synchronous circuits. Besides requiring lots of effort to deal with clock distribution and skew, clocking everything all the time wastes power. They designed an asynchronous version of the ARM (Acorn RISC Machine) in which various parts of the CPU operate independently, being clocked—and synchronizing with each other-only when necessary. The bad news is the async design turned out to have little performance / watt advantage. The good news is the technique is workable and may prove fruitful yet. In any case, they deserve credit for having the guts to show their results without trying to hide behind a hypescreen.

## SVPC

*So you* say you did go to a PC show in the big city once. Bright lights, pretty girls, glistening spreadsheets. Oh, you even came home with some neat souvenirs including a high-capacity pocket protector and a "Stamp Out Bugs" flyswatter.

Yawn...

The Silicon Valley PC Conference isn't for users of PCs, it's for those who design PCs. As such, you won't find any glitzy booths, tacky give-aways, or used car salesmen making a career change.

Instead, you get lots of valuable, technical information including about a zillion pages of proceedings and tutorials covering everything from "GaAs Cache Controllers for Pentium" to "Consumer Friendly Battery Cavity Designs."

Of course, when you're talking PC design you're talking chipsets, and I came home with about ten pounds of datasheets describing a bewildering array of offerings. Gone are the days of a simple AT chipset-now the market is fragmented into every possible mix of CPU ('386SX to Pentium), bus (ISA, EISA, MCA, PCI, VL), bus width (16, 32, 64), cache type (write-through or write-back), voltage (5V, 3V), notebook versus desktop, and on and on.

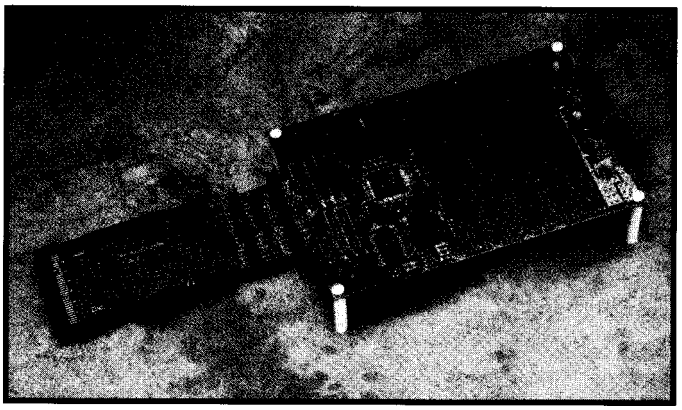


Photo **3**—*The Focus Microsystems CARDTools PCMCIA prototyping card brings everything outside for easy probing.*

One of the newer (of the dozens) of offerings from Opti is the 82C802 '486 PC / AT chip (Figure 3a) which integrates all the messiest glue logic including memory control (DRAM and cache) and an ISA interface. Building a high-performance PC is quickly becoming a rather simple "connect the dots" exercise.

Similarly, Ethernet design is demystified with the Standard Microsystems

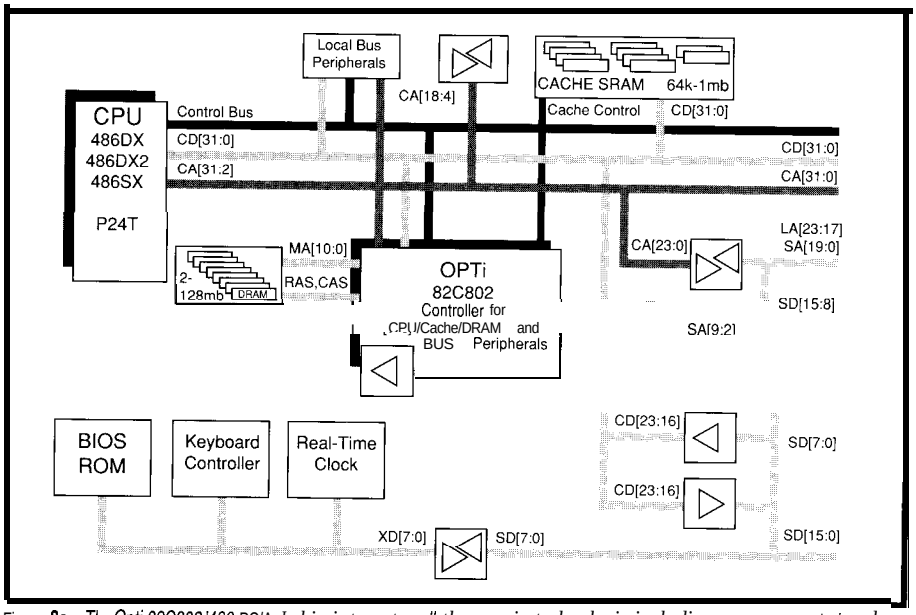



Figure **3a**—*The Opti 82C802 '486* PC/A *J chip integrates all the messiest glue logic including memory control and an ISA interface.*

SMC91C92 (Figure 3b), a single chip that integrates controller, 4K packet memory with buffer manager, and provides a direct interface to 10BaseT twisted pairs (it can also work with external thin [10Base2] and thick [10Base5] coax transceivers]. With chips like this, I imagine it won't be long before Ethernet starts migrating onto motherboards.

Chips & Technologies, suffering from an ill-starred 'x86 clone foray, is fighting back with the 82C735 (Figure 3c) that packs all the "other stuff"—namely serial, parallel, game, floppy, and IDE ports-into a single chip.

Nothing says these chips only work in PCs. Keep in mind that they can also prove to be low-price problem solvers in non-PC applications.

## IC CARD EXPO

I beg all you marketing types out there contemplating the creation of the next consortium, standard, alliance, or whatever-please come up with a name that is easily converted to an acronym.

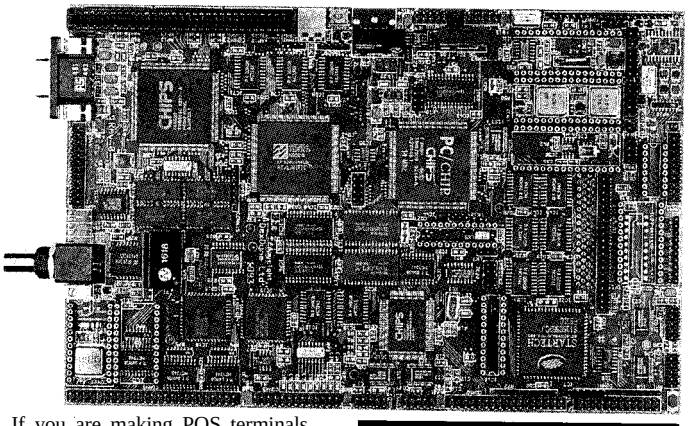
Witness SCSI, which is now affectionately named "scuzzy." I don't


# TotalPc™ - *the* embedded system

High integration compact PC controller.

**STANDARD FEATURES OF *TotalPc*™**

- 14 MHz PC CPU with CGA interface for LCD/CRT display
- Standard PC keyboard I/F and 12 x 8 matrix keypad interface
- Memory space for up to 2M byte EPROM, 4M byte of RAM and 4MBytes of FLASH
- Watchdog timer, Time of Day clock
- Floppy, IDE, Printer and 5 serial RS232 interfaces, two with optional isolated RS485
- Thin Ethernet I/F
- VGA support for LCD and CRT
- PCMCIA-2 I/F with hot insert capability
- 24 Opto 22 compatible buffered industrial parallel I/O lines
- 8 single or 4 differential input 12 bit ADC and 8 output 8 bit DAC
- Fully buffered PC Expansion slot
- Single 5V operation. Low power mode for battery operation



If you are making POS terminals, GPS systems, field portable instruments. factory data terminals, Ethernet based outstations or plain process control systems for industrial environments you will find that ***TotalPc***™ gets to the heart of your problem. Our EPROM based Version 5.0DOS provides 3 known and proven development environment for your application software on the ***TotalPc***™

OEM and customisation enquiries welcomed.

dexdyne

DEXDYNE LIMITED

15 Market Place
Cirencester
Gloucestershire **GL7 2PB**
England

**Tel: 0285 658122**
**Fax: 0285 655644**

#139

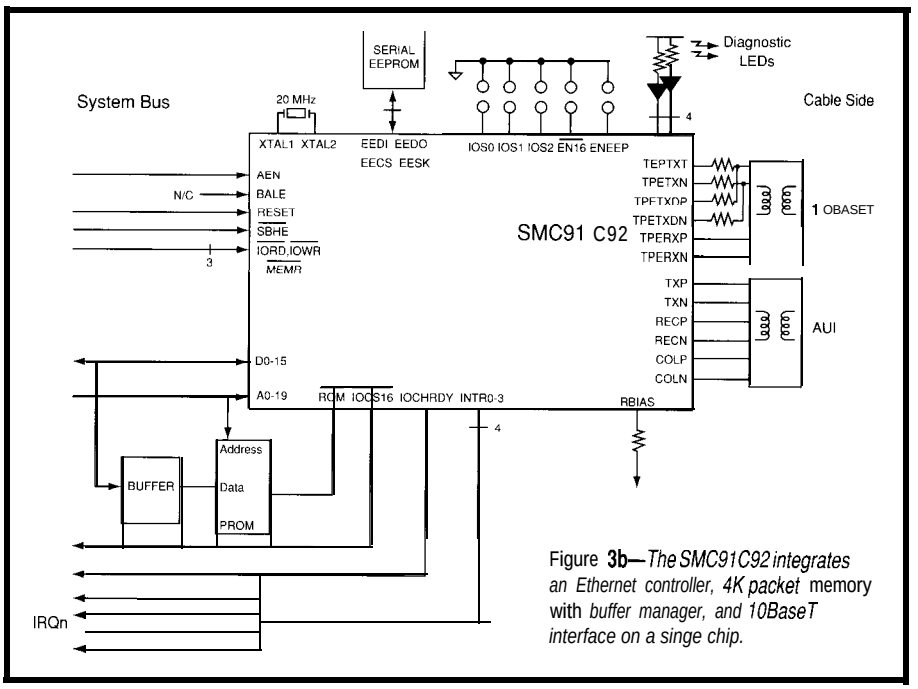



Figure 3b—*The SMC91C92 integrates an Ethernet controller, 4K packet* memory with *buffer manager, and 10BaseT interface on a singe chip.*

know about you, but when I was a kid "scuzzy" referred to antisocial behavior like picking your nose in public. I remember a passionate editorial in the early days of SCSI that pushed for naming it "sexy," which might work for a car, but not a chips or standards (though both cars and chips are guilty of (ab)using the tired SX—gee, what does that mean-moniker). In mixed company (i.e., noncomputer types) I often find myself saying "S" "C" "S" "I," just to avoid the concerned glances and need for long-winded explanations.

Thought "SCSI" would have taught us a lesson, eh?

How does "Puh Chum See Uh" roll off your tongue? Or maybe it's "Pee Cee Mick A?" Perhaps you should just take a deep breath and spit it out as fast as you can—"P" "C" "M" "C" "I" "A." Or better yet, just punt and do what savvy show organizers do-call it "IC card."

Whatever you call PCMCIA, indeed whatever PCMCIA is (you'll see what I mean], it is here to stay. Many of the latest crop of laptops, sublaptops, and PDAs include PCMCIA slots.

Originally, PCMCIA was intended to bring order to the narrowly defined niche of "memory cards." In particular, its fortunes were closely tied to those of Flash memory, whose proponents loudly proclaimed their intention to replace disk drives.

Of course, the disk folks weren't happy so, with the help of 1.8" drives and a lobbying effort, PCMCIA was expanded to support disk drives, most notably by allowing for thicker cards.

Somewhere along the line, everyone realized that what was once a "memory card" slot could just as well handle other I/O functions. Talk about a slippery slope-now the PCMCIA bus supports all kinds of interfaces and comes in three sizes (3.3 mm, 5.0 mm, and 10.5 mm thicknesses] and I understand a fourth (probably around 15 mm) is being discussed.

As far as the original battle between Flash and disk goes, the presumption/hope that the former will generally replace the latter is definitely overblown. Due to a variety of cost, capacity, and reliability issues (Figure 4/Photo 1) they really serve different applications.

While Flash cards top out at 10–20 megabytes, PCMCIA disk drives are pushing past 100 megabytes. Worse, Flash prices remain stubbornly high at $50–$100 per megabyte while even state-of-the-art 1.8" drives are only a few bucks per meg. On the other hand, despite admirable efforts by the disk suppliers, chips are always going to be lower power, more rugged, and quieter than disks.

I contend there's room for both, with disks serving the sub/notebook class "PC equivalent" market, and Flash the emerging demand for PDA

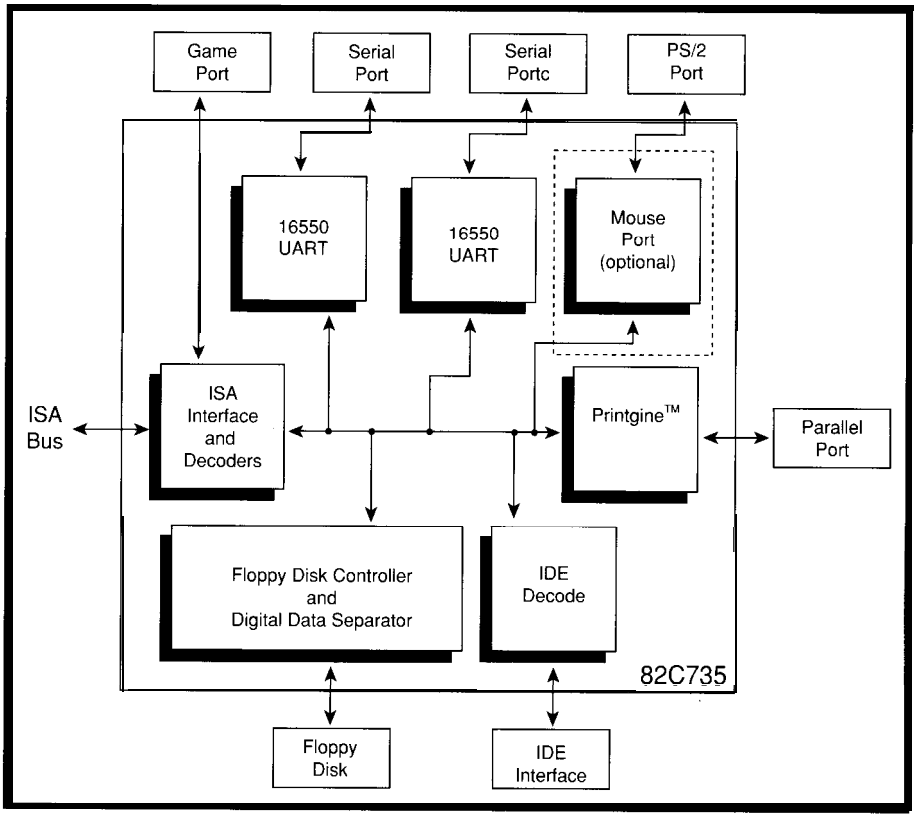



Figure 3c—*The Chips & Technologies 82C735 puts serial, parallel, game, floppy, and IDE interfaces on a single chip.*

| P/N: | Maxtor MXL-105 | Sundisk SDPL5 |
|---|---|---|
| Capacity: | 105 MB | 10 MB |
| Data Transfer Rate: | I-2 MB/s | 1.25 MB/s |
| Avg. Seek Time: | 18 ms | nil |
| Active Power: | 1.95 w | 0.25 W |
| Sleep Power: | 25 mW | 5mW |
| Startup Time | (Sleep->Active): 2.5 s (est.) | nil |
| Thickness: | 10.5 mm (TYPE III) | 5.0 mm (TYPE II) |
| Weight: | 65 grams | 35 grams |
| Op. Temp: | 5–55°C | 0–60°C |
| Op. Shock: | 100 G | 1000 G |
| Noise @ 1 meter: | 34 dBA | nil |
| Price/MB (qty 1): | $5.00 | $100.00 |

**Figure 4**—*In the battle between hard disks and flash memory, disks have a distinct size and price advantage, while flash wins for low power, high speed, and silent operation*

and other pocketable electro-gizmos. In fact, major players from the opposing camps (for example Seagate and Sundisk) are making peace-and deals-with each other.

To connect an existing (i.e., non-PCMCIA) machine, you'll need a "drive." Data I/O offers CardPro (Photo 2a), an external unit that connects to a PC's parallel port. For an internal unit, check out the Greystone CardDock (Photo 2b) that mounts into a 5.25" bay and comes in one-, two-, and four-slot versions.

IBM has an IR LAN interface that allows your PDA (PCMCIA) and office system (MCA, ISA) to talk sans wires at up to **1** megabit/second. Notably, it uses diffuse optics to avoid the line-of-sight restriction of other IR schemes and can cover about 1000–2000 square feet in typical office settings.

If you want to strike it rich by designing your own PCMCIA gadget, check out the Focus Microsystems CARDTools prototyping card (Photo 3). It brings everything outside for easy probing and the 4" x 6" breadboard area accommodates through-hole devices for wirewrapping. Interestingly, the board is based on a Xilinx reprogrammable FPGA for the utmost in design versatility.

If all this makes your head spin, take a position check with Socket Communications Global Positioning System [a.k.a., GPS). Tracking up to eight satellites at once, it not only reports your position anywhere on the face of the globe, but also gives you access to triple-redundant atomic clocks. I believe the positional accuracy (25 meters) is limited by DOD mandate lest some hacker try to homebrew an ICBM.

The skeleton in the PCMCIA closet revolves around higher-level system/software compatibility issues. Your quiver of gadgets may all have PCMCIA slots, but moving cards from one system to another is a plug-and-pray proposition. Though they're working on it, at this point each card/system permutation needs a driver which is, as far as I'm concerned, worse than DIP switches.


**PC-Based Logic Analyzers**



**Sophistication at Low Cost**

ID160 **(50MHz)** $595

ID161 (100 MHz) $695

***High Speed • 8K Trace Buffer • 16 timing channels expandable to 32 state channels *Multi-Level Triggering *State Pass Counting *Event Timer/Counter *Performance Histograms *Hardcopy Output *Disassembles I-bit micros *Supports VGA /EGA/HGA *Demo diskette available**

30 Day Money Back Guarantee

INNOTEC DESIGN, INC.
**6910 Oslo Circle, Suite 207**
**Buena Park, CA 90621**
**Tel: 714-522-1469 FAX:714-527-1812**

#140

**▶ Project Part5 ◀**

| | |
|---|---|
| **New Stuff** | |
| TDA8444 I²C bus octal 6-bit DAC | 9.30 |
| MT8888 DTMF Transceiver (Intel bus interface) | 10.50 |
| PCF8583 I²C bus clock/calendar/counter w/ 256 byte RAM | 11.70 |
| UDN2998W Dual full-bridge stepper driver: 50 V, 2 A | 13.80 |
| PCF8591 I²C bus quad 8-bit ADC, single 8-bit DAC | 15.10 |
| BQ2003 Fast-charge NiCd/NiMH battery controller | 14.55 |
| MT8808 8x8 bipolar signal analog crosspoint | 16.10 |
| BQ2001 Battery Energy Management Unit | 22.75 |
| HBCS-1100 HP high-resolution bar code sensor | 32.80 |
| CH1840 FCC pre-approved DAA (full function, street legal phone hookup) | 72.50 |
| **Firmware Furnace Embedded '386SX Stuff** | |
| Complete schematics (15 pages or so) | 5.00 |
| DS2400 Silicon Serial Number (2 chips) | 8.40 |
| MAX691 Power supervisor | 11.85 |
| Firmware Development Board ICs & parts (through Issue 36) | 45.70 |
| **Firmware Flyers** (prices are postpaid in US / with any parts order) | |
| 8051 Firmware Debugging Techniques (65 pages w/ 3.5" diskette) | 18 /15.00 |
| Introduction to the 8051 Instruction Set (46 pages) | 10 / 8.00 |
| 8051 Instruction Reference Card | 3 / 2.00 |
| 8255 Cheat Sheet Reference Card | 3 / 2.00 |
| **IR Remote Control** | |
| LD273 dual IR LED (bright, wide beam) | 2.10 |
| IR3C02A laser diode controller (±5, for LT022MC/LN9705P laser) | 2.30 |
| IR3C07 laser diode controller (+5V, for LT022PD/LN9705 laser) | 2.85 |
| PH302 fast IR photodiode | 3.10 |
| GP1U52Y 40 kHz IR receiver (side-looking) | 4.00 |
| IS1U60 38 kHz IR receiver | 4.80 |
| IRSAMPLE parts (PH302, LM311, etc. See MCIR-Link article, INK 29) | 5.70 |
| Excellent IR filter (opaque to visible light, 35 mm slide mount) | 6.75 |
| MC145030 IR encoder/decoder | 6.75 |
| IR I/O: IS1U60, LD273, CD4047, 2N2907, red LED, schematic (IR-Link!) | 10.40 |
| **Stepper Motor & Power Drivers** | |
| ULN3751Z Power op amp (±3V to ±13V supply, 3.5 A output) | 4.60 |
| UDN2993B Dual full H-bridge bipolar stepper motor driver | 5.45 |
| UCN5841A Serial-input, 8 latched 500 mA sink drivers | 6.90 |

**...and much more...**

IPS Ground/2nd day $6.519 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 partssource ◀

13109 Old Creedmoor Road • Raleigh, NC 27613

FAX/voice (919) 676-4525

Anyway, despite growing pains, PCMCIA will muddle through and eventually we'll all be using it.

## ON THE ROAD AGAIN

A glance at the calendar reminds me this circus never ends with at least half a dozen shows (DSP Expo, Embedded System Conference, Microprocessor Forum, etc.) coming up.

Racing from one venue to the other, hiking from booth to booth, lugging bushels of literature, wading through the hype-1 guess that's what they mean by "no free lunch."

*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.*

## IRS

*419* Very Useful
420 Moderately Useful
421 Not Useful

Hot Chips, IEEE
701 Welch Rd., Ste. 2205
Palo Alto, CA 94304
(415) 941-6699

SVPC, Systech Research
1625 The Alameda, Ste. 207
San Jose, CA 95126
(408) 293-8383

IC Card Expo
6300 S. Syracuse Way, Ste. 650
Englewood, CO 80111-9912
(303) 220-0600

PCMCIA
1030 East Duane Ave., Ste. G
Sunnyvale, CA 94086
(408) 720-0107

Opti
2525 Walsh Ave.
Santa Clara, CA 9505 1
(408) 980-8178

Standard Microsystems Corp.
80 Arkay Dr.
Hauppauge, NY 11788
(516) 273-3100

Chips & Technologies, Inc.
3050 Zanker Rd.
San Jose, CA 95 134
(408) 434-0600

Sundisk
3270 Jay S.
Santa Clara, CA 95054
(408) 562-0500

Maxtor
211 River Oaks Pkwy.
San Jose, CA 95134
(408) 432-1700

Data I/O
10525 Willows Rd. NE, P.O. Box 97406
Redmond, WA 98073
(800) 332-8246

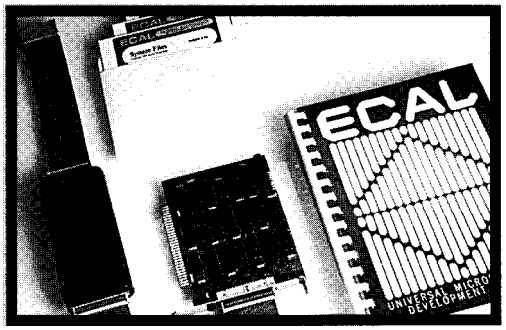
Greystone Peripherals
130-A Knowles Dr.
Los Gatos, CA 95030
(408) 866-4739

IBM
P.O. Box 100
Somers, NY 10589
(800) 426-0181

Focus Microsystems
1735 N. First St., Ste. 307
San Jose, CA 96112
(408) 436-2336

Socket Communications
2501 Technology Dr.
Hayward, CA 94545
(510) 670-0300

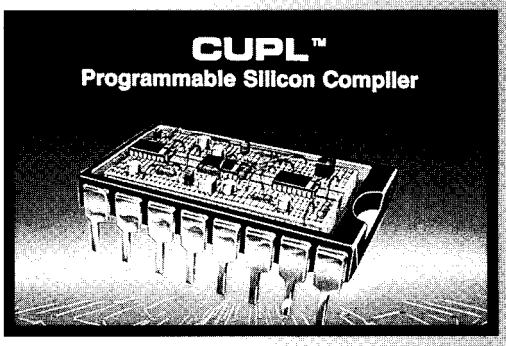
ECAL
Universal Microprocessor Development system




- Memo y Emulator Pod
- 200 Microprocessors Supported
- High Speed Macroassembler
- Source level Debugging
- Integrated Editor
- Complete! $899. 00

digelec

CUPL
PLD/FPGA DESIGN




- Supports PLDs, and FPGAs from most vendors
- Most Fitters included at no extra cost
- Powerful HDL like Syntax & High Level Boolean Equations, and Multiple State Machine entry
- Supported Platforms: DOS and Windows, 3.0 / 3.1 Versions, Macintosh, SUN Spark
- Supports 12 different Schematic Interfaces

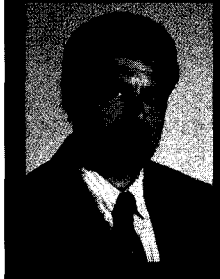
1 800 935-4335

#141

# Conversing with Batteries

## EMBEDDED TECHNIQUES

John Dybowski



Revisiting a topic he only touched upon in a recent column, John pulls out the Benchmarq catalog again and checks out the bq2010 Gas Gauge chip. Off-loading the task onto a dedicated chip makes a lot of sense.

Intelligent microprocessor peripherals exist now that are "smart enough" to independently perform functions that, until recently, would impose considerable overhead on the system processor. In many cases, this resulted in the consumption of significant resources. These resource penalties not only include run-time processing power (which is ultimately the currency of a real-time system), but also trickle back to the initial product development phase, effectively imposing additional constraints on the design team.

The algorithms and supporting circuitry required to handle these ancillary functions in many cases turns out to be somewhat less than trivial. Frequently these functions are of a specialized nature and, lacking expertise in these areas, many engineers find they must expend a considerable amount of time and energy in initial research along with the subsequent trial and error to get it right. The empirical method lives on and arises from the need to progress with insufficient data or knowledge.

When you are presented with this choice, you must decide on the wisest use of your time. And you must temper this decision with the realization that time is also the currency of successfully bringing a product to market during the best window of opportunity. The difficulties of shrinking development cycles and greater product complexity are compounded by the growing sophistication of end users who know what they want and know how they want it. You must, at the very least, keep up with the ever-changing status quo.

In previous columns I showed you ways to unburden your processor by using various complex peripherals to take an increasingly larger role in handling various isolated functions. This off-loading makes a lot of sense for a couple of reasons. First, relegating the design of specialized functions to those who are experts in their particular field naturally results in superior performance. Second, since the expected volume for these devices is presumed to be quite high, you can expect highly integrated and specialized monolithic circuits to perform these individual purposes.

Naturally, the ultimate cost of these function blocks is usually quite reasonable since the considerable development costs inherent in developing new technologies can be amortized over a large unit volume. When faced with the abundance of specialized circuits on the market, the question of whether to build or to buy becomes ever easier to answer. Unless you have a very large potential volume, and a very large engineering budget, the question is usually moot.

But making a purchase decision demands that you evaluate the current market offerings with great care. To assume that the latest products are thoroughly debugged could be a fatal assumption. Although many leading-edge software products are infamous for their bugs, this malady is not restricted just to bits and bytes. I've encountered my share of buggy silicon.

Although everyone can gain an advantage from this natural progression in integrated peripheral technology, this proves to be especially beneficial to the small developer. This trend places the expertise of these various specialists at the disposal of smaller shops that previously had no other recourse but to make do with what could be accomplished with their limited resources. More and more, the modern engineer must operate as an evaluator in selecting the best packaged solution to a particular problem.

At first glance, this view may seem a bit demeaning. However, the end result is still positive in my

opinion. Designing in proven solutions instead of insisting that everything be invented here does make a lot of energy available that can be used to focus on the bigger picture. This approach frees the engineer to think and act with more capacity as a system architect.

## I'VE GOT THOSE BATTERY BLUES

A particularly difficult pursuit for many has been the issue of battery management. Although fast charging of secondary batteries has been addressed by various manufacturers, the answer to the accurate gauging of a battery's available charge has proven to be somewhat more elusive. The determination of a battery's state of charge is a good example of what I have described thus far. As is often the case, there's more to it than first meets the eye. Whereas many have tackled this problem, read on and see if you agree that this is, in fact, an undertaking of such proportions that it is better left to those who are truly experts in their field.

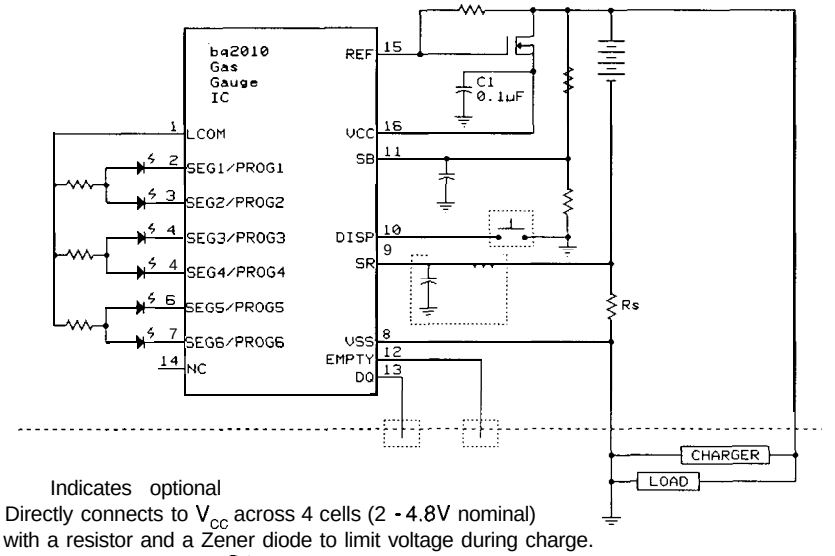
The fundamental idea behind battery charge determination is to accumulate a measurement of charge and discharge currents. This *current metering* can be accomplished using a small-value sense resistor in series with one of the battery terminals. The voltage developed across this sense resistor represents the current that is either flowing into or out of the battery. This voltage is then scaled and integrated over time and is used to drive a counter that indicates the battery's current state of charge numerically. As in most technical undertakings, the simplicity of fundamental concepts evaporates once the details become known. Let's look at those details.

First of all, charge acceptance, which is charge efficiency, is different at different rates of charge; more charge is retained when batteries are charged at high rates. Because of this, most people like to charge batteries at rates as high as the batteries will safely tolerate. Similarly, the rate of discharge determines how much overall energy can be delivered to the load over time. That is, you get more amp-hours at low levels of discharge than when driving heavier loads. Both of these cases illustrate the need for rate compensation and show that it's not quite as simple as just monitoring current-in versus current-out.

Furthermore, a battery will perform differently at different temperatures. For example, if a battery is known to be 60% full at 25°C, the corresponding state of charge for the same battery will be only 40% at 0°C. Similarly, charge acceptance will be different at different temperatures. In addition, for any equipment that remains inactive for periods of time, the battery's self-discharge must also be considered, and here temperature plays a part as well.

At this point, the need for temperature compensation should be obvious. While discussing self-discharge, you should also keep in mind that the particular battery chemistry determines the nominal discharge rate. NiMH batteries self-discharge a higher rate than NiCd types, and the appropriate nominal value should be applied. For these reasons, the charge determination circuitry clearly should be contained in the battery pack itself since the battery and its operating environment must continually be monitored.

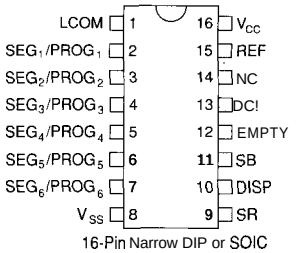
Evidently, supporting the functions I described will require some kind of processor, an analog-to-digital converter, a temperature sensor, and other linear circuits and passive components. It stands to reason that in many cases all but the most indispensable functions get discarded, as cost and complexity are weighed against the perceived value of including such functionality.




Indicates optional
Directly connects to $V_{CC}$ across 4 cells (2 - 4.8V nominal) with a resistor and a Zener diode to limit voltage during charge. Otherwise, RI, CI, and Q1 are needed for regulation of > 4 cells.

Programming resistors (max. = 6) and ESD-protection diodes are not shown.

R-C on $S_R$ to capture pulses < 20ms duration.




16-Pin Narrow DIP or SOIC

| | | | |
|---|---|---|---|
| LCOM | LED common output | REF | Voltage reference output |
| $SEG_1/PROG_1$ | LED segment 1/ program 1 input | NC | No connect |
| $SEG_2/PROG_2$ | LED segment 2/ program 2 input | DQ | Serial communications input/output |
| $SEG_3/PROG_3$ | LED segment 3/ program 3 input | EMPTY | EDV-detected indicator output |
| $SEG_4/PROG_4$ | LED segment 41 program 4 input | SB | Battery sense input |
| | | DISP | Display control input |
| $SEG_5/PROG_5$ | LED segment 5/ program 5 input | SR | Sense resistor input |
| | | VCC | 3.0-6.5V |
| $SEG_6/PROG_6$ | LED segment 6/ program 6 input | $V_{SS}$ | System |

Figure 1—*Only a few additional components are needed for an operational bq2010 circuit. Depending on your needs, an LED display and a single-wire serial link may be used in stand-alone operation.*

There are other considerations as well that limit which functions can practically be employed. For instance, using the system processor to monitor the battery's self-discharge could be constrictive simply because of the power consumption resulting from the monitoring activity itself. In fact, the self-discharge could be entirely swamped by the current required just to keep the processor running! And so it goes. Compromises are made and features discarded. Ultimately you might wind up with a rudimentary battery monitor that just pays lip service to the initial promise.

## BATTERY GAUGING RESCUE

It seems that just as you're about to throw in the towel, a part appears on the market that looks like the answer to your problems. In this case, the answer comes in the form of a 16-pin circuit that is housed in either a 300-mil DIP or an SOIC package. This offering comes from Benchmarq and is called the bq2010 Gas Gauge IC. This IC not only fulfills the somewhat abbreviated shopping list of features I outlined above, but also provides many other features and capabilities. The bq2010 is capable of true stand-alone operation with its built-in drivers for status LEDs and also offers a simple, single-wire communications link to the host processor.

Don't be fooled by the small size. The bq2010 is a well-refined circuit that addresses all the needs of a comprehensive battery gauging system. Before I tell you more about the bq2010, let me touch on some of its more significant features:

- 120-µA standby current
- *Direct drive capability for six indicator LEDs
- *Single-wire serial comm. port
- *Built-in temperature sensor
- *Charge/discharge temperature and rate compensation
- •Self-discharge temperature compensation
- •500: 1 current measurement range
- *On-chip support for voltage regulation using an external FET
- *Battery voltage monitor for end-of-discharge determination

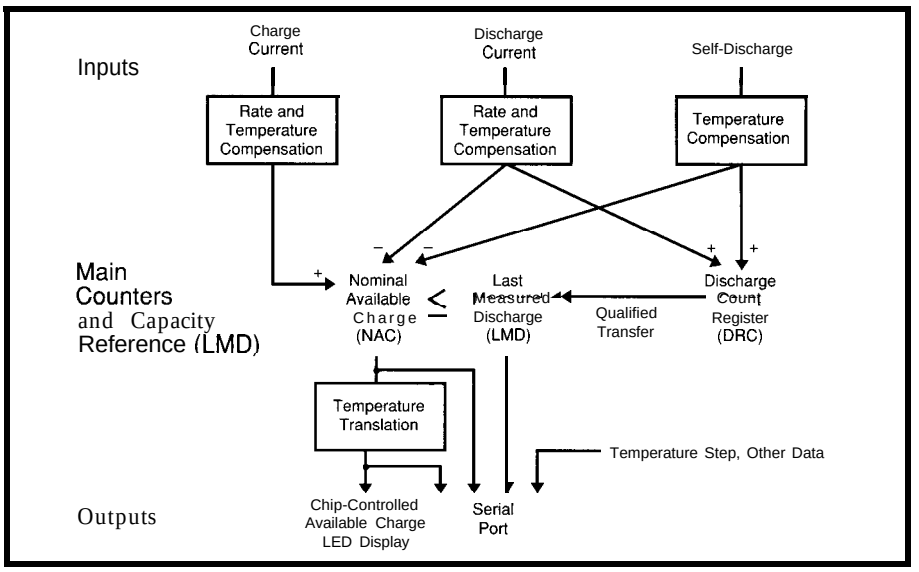



Figure 2—*Most of the glue around the bq2010 is optional and depends on your final application.*

The bq2010 is engineered to satisfy the various levels of integration that may be encountered in the real world. For instance, due to its small size and limited number of external components, existing equipment can be retrofitted without any significant battery pack retooling (the entire circuit card can fit in the space between two AA batteries). And because it can operate without the need for any attention from the system processor, the processor system does not have to be reworked either.

Similarly, the benefits of battery gauging can be incorporated into new designs with a minimum of engineering by also using the Gas Gauge in a stand-alone fashion. With only slightly more effort, the bq2010 can be more tightly coupled to the system where the system processor can exchange information with the battery pack using the single-wire serial link. Here, the LED display can be eliminated if desired and status information can instead be displayed using the system's display with greater resolution than the several LEDs would otherwise allow.

Figure 1 shows what the bq2010 looks like and what is required to get it operational. This circuit provides for both stand-alone operation, using the LED display, and a single-wire serial link to the host processor. Also shown

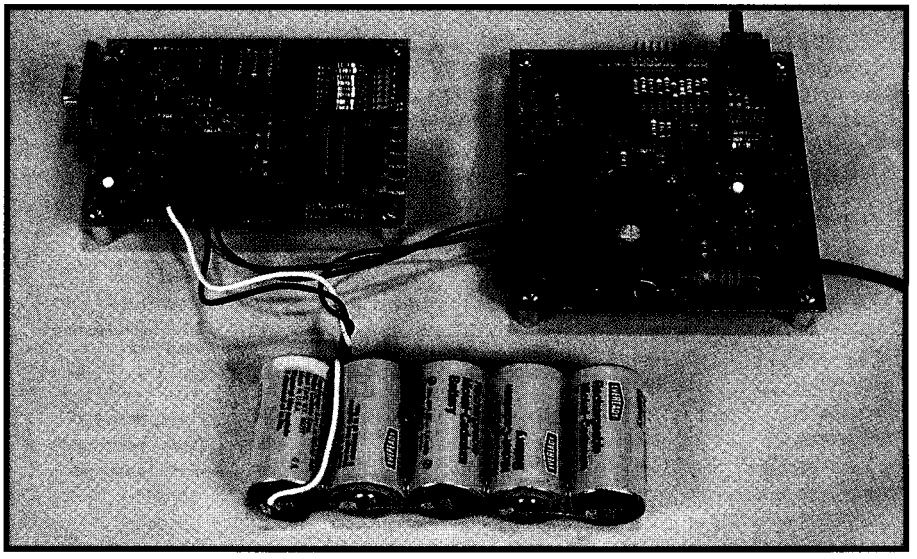


Photo 1--The *EV2010 evaluation package, which includes the bq2010 and PC software, allows the user to control infernal registers as well as directly drive the chip's outputs.*

in this figure is a simple voltage regulator (required if using more than four cells) built around the n-channel FET, R1, and Cl. Note that the capacitor at SR may be required to stabilize the sense voltage if large current fluctuations are expected at the battery. Since the LEDs are normally off to conserve power, the switch connected to \DISP is used to momentarily enable the display.

Not shown in this figure are the programming resistors that connect to the LED drive pins, which are used to program the bq2010 on initial application of power. These programming resistors configure parameters such as the Programmed Full Count and the associated scale multiplier as well as the nominal self-discharge rate.

The display mode can also be programmed to function in either relative or absolute mode. Absolute mode always references the state of charge to the Programmed Full Count, where the relative mode self-adjusts as the battery gradually degrades. The absolute display mode will reflect the loss of capacity that all batteries exhibit as they wear out. Of course, some users might get confused as the battery eventually never returns to the full state after being normally charged. Nonetheless, the implication is that sometimes it pays not to show your customers that the battery is gradually drying out!

In order to conserve pins, the bq2010 shares its LED driver pins with the programming function. Here's how it works: During device power-up or a forced reset, the LED display pins input configuration information that is set by either connecting these pins to Vcc or Vss through a 200k resistor, or by leaving them floating. These three-level inputs serve to keep the number of programming inputs in check since all of the of options could not be conveyed using simple binary levels. Since these pins perform dual functions, the LEDs' power source must be derived from LCOM in order to allow the bq2010 to disable the source so the programming inputs can be read. Look to Figure 2 for a simplified view of what's involved in the bq2010's capacity determination algorithm.

## NOW YOU'RE TALKING

As I mentioned, the bq2010 includes a simple, single-wire interface that is used for bidirectional communications to the system processor. This open-drain pin is pulled up via a resistor to the system's Vcc and along with the return lead forms the serial interface for battery communications. All that's required to tap this capability is an additional contact on the battery pack. By using this interface the system processor uses a command protocol where various battery and system characteristics can be monitored and modified. This is accomplished via direct access to the various internal registers of the bq2010. This communications capability is useful not only for purposes of configuration and monitoring, but also assists during the initial battery pack testing.

Communication is carried out using an asynchronous return-to-one format composed of an 8-bit data stream with a maximum transmission rate of 333 bits per second. You get the bq2010's attention by hitting it with a line break that has a low time of 3 ms. Following the return of the line to a high condition, the bq2010 is ready for communication and bytes of data can be shifted about.

The actual bit frame used by both the host processor and the bq2010 consists of three sections. First the line is pulled low. This is the start hold time and is maintained for 500 µs. Next, the actual bit value is transmitted, where the data level is valid following a setup time of 750 µs. The data bit now must be held in this state for 750 µs, which is the data hold time. It is during this interval that the data bit is sampled by the receiving device. Finally, the bit transmission is terminated by returning the line to a high state. This must occur within 2.25 ms of the negative edge that initiated the transfer. In order to ensure the conclusion of the bit transmission is recognized properly, this level must be held for at least 700 µs.

Unlike some of the asynchronous serial protocols that I've described in recent columns, this one adheres to


**WHY** are our cross compilers so inexpensive???

Because we **give** them away **free!**

We base our cross compilers on the GNU C/C++ compiler from the Free Software Foundation. We provide you with one year 01 support*, and give you a ready-to-run cross compiler with *complete source* for DOS, Windows, OS/2 2.0 or UNIX for $495 per year. Or, get the extended support package for $895, which includes GNU *Emacs* and *make*, the CVS and RCS source code control utilities, and the TEX typesetting system. Targets include i386, i860, i960, Motorola 680x0, 683xx and 88000, MIPS and Sparc.



Hundred Acre Consulting
5301 Longley Lane Suite D-144, Reno NV 89511
(800) 245-2885 +1-702-829-9700

*Compilers are available without support for a media distribution fee.

#142

some fairly relaxed timing constraints. This is appropriate to the task at hand since any dialogue between a processor and its battery would, by nature, be brief and to the point. An added benefit of this simplicity is the fact that it is easy to communicate with a battery using this configuration with a computer. This is exactly the capability Benchmarq provides in their bq2010 evaluation package.

The EV2010, shown in Photo 1, consists of a circuit card containing the bq2010 along with its support circuitry and an RS-232 interface. It also includes some very useful PC software. This evaluation system really assists you when you are trying to gain a better understanding of the capabilities, derived benefits, and features of a comprehensive battery monitoring system. Not only does the software allow you to view and control all of the bq2010's internal registers in real time, but you can also directly drive the bq2010's outputs.

An especially valuable feature of this package is its data logging capability that can be used for testing the effectiveness of your charging scheme and for battery characterization and evaluation. You select a sample period and the name of a file and you get a bunch of practical information that you can feed into almost any spreadsheet or analysis program. The data log consists of ASCII data records separated by tab characters and contains the following elements:

| | |
|---|---|
| *TIME | hh:mm:dd |
| • LMD | Last measure discharge value in mAH |
| •NAC | Nominal available charge in mAH |
| *FLAGS1 | Binary values of FLAGS1 bits |
| *FLAGS2 | Binary values of FLAGS2 bits |

## THAT'S ALL FOLKS

In discussing such a complex and specialized circuit such as the bq2010, it was unavoidable that I skip a lot of details. If you'd like to learn more about the bq2010, then the answers to your questions are contained in Bechmarq's data sheets and application notes, or you can talk to their applications engineers. Or better yet, get your hands on an EV2010 evaluation kit and let your PC do the talking.

*Thanks to Benchmarq's Mike Calise for materials used in this article.*

*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.*

## SOURCE

Benchmarq Microelectronics, Inc.
2611 West Grove Dr., Ste. 109
Carrollton, TX 75006
(214) 407-0011

## I R S

422 Very Useful
423 Moderately Useful
424 Not Useful


**Device Programmers**

Connects to PC parallel printer port

48 PIN ZIF




$449.95 EMP-20

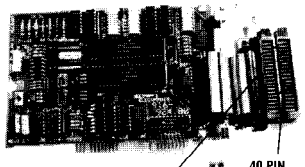
SA-20 8 Gang Eprom

20 X 4 Line LCD Display



$750 Key Keypad

PB-10 Internal Card for PC



$139.95 2 ft. Cable 40 PIN ZIF

- Easy to use software, on-line help, full sceen editor
- **Made in USA**
- 1 & 2 Year Warranty
- Technical Support by phone
- 30 day Money Back Guarantee
- FREE software upgrades available via BBS
- Demo SW via **BBS (EM20DEMO.EXE) (PB10DEMO.EXE)**
- **E(e)proms** 2716 - 8 megabit, 16 bit 27210-27240, 27C400 & 27C800,
- Flash 28F256–28F020, (29C256–29C010 (EMP-20 only))
- Micros 8741A, 42A, 42AH, 48, 49, 48H, 49H, 55, 87C51, 87C51FX, 87C751,752
- GAL, **PLD** from NS, Lattice, AMD-**16V8, 20V8, 22V10** (EMP-20 only)

FOR MORE INFORMATION CALL

**NEEDHAM'S ELECTRONICS, INC.**

4539 Orange Grove Ave.
Sacramento, CA 95841
(Monday-Friday, 8 am-5 pm PST)

MasterCard VISA
C.O.D.

**(916) 924-8037**
**BBS** (916) 972-8042
FAX (916) 972-9960

#14



***8051 EMBEDDED CONTROLLERS***
***With Lots of Extras!***



We offer a full line of low cost 80C32 embedded controllers and software tools which are ideal for developing products, test fixtures and prototypes.

***Features include:***

- Low power CMOS design
- Up to 60K of code space and up to 60K of data space
- 5 to 15 volt operation
- Small form factor (3.5" * 6.5") with prototyping area
- System diskette includes application notes
- Start at $100

***Available Options:***

- Multifunction Board adds A/D, 24 I/O lines and more!
- BASIC-52 or Monitor/Debugger in EPROM,
- C Compiler $100 or BASIC Compiler for $300

**Iota Systems, Inc.**
POB 8987 . Incline Village, NV 89452
PH: 702-831-6302 . FAX: 702 831-4629

#14

# PATENT TALK by Russ Reiss

As engineers, designers, programmers, and entrepreneurs intimately involved with the application of microprocessors, it is gratifying to know that the need for our services continues to mushroom every year! Microprocessors are being applied to new areas at such a pace that it truly does seem at times as though there will soon be nothing left that does not have one embedded within it. This month's special topic theme on "Embedded Control" led me to conduct a broad search for just such applications. The result was a large and diverse assortment of patents which use embedded microprocessors to perform a seemingly endless variety of functions. I will report on seven of them-ranging from camera lenses to car washes-which illustrate how encompassing the applications are becoming.

Sony's "smart lens" system presented in Abstract 1 is a good example of a growing trend. Where, in the past, it was novel to have even one microprocessor in a device such as a camera, nowadays we are seeing "distributed intelligence" with multiple microprocessors, each handling separate functions. As is typical in such applications, the processor in the removable lens mechanism informs the main (camera) processor of that subsystem's capabilities and handles local functions such as controlling focus and zoom drive motors in response to higher-level position commands from the processor within the camera body. Undoubtedly, one reason for this growing trend toward the use of multiple processors is that as integrated circuit costs drop, it is simpler and more cost effective to keep all the signal lines associated with a specific function within that physical subsystem. Only power and communication lines need be joined when the systems are coupled together.

Abstract 2 covers a device familiar to all of us who work with electronics: the soldering iron. A "simple" device, which has evolved from being heated in a fire, to using electric coils, to permitting multiple temperature settings, and finally to providing continuous temperature control within a feedback loop, it now has microprocessor control! Moreover, in Cooper Industries' version, this processor can support a display device and/or communicate with a central computer for data logging, retrieval, and display. One wonders in this case if it is possible for the sensors within the soldering station to detect (and log) the time at which each connection was soldered as well as the temperature at which it was done and the duration of the event. This information could prove useful in quality control as well as in operator training. This patent illustrates another widespread trend; namely that once a processor is present, it presents a natural means for the display of information and communication with other devices. Note, once again, how distributed computing is at work here, using processors within each soldering station to handle local functions while communicating with a central computer for higher-level functions.

1

| | |
|---|---|
| Patent Number | 4,967,281 |
| Issue Date | 1990 1030 |
| Inventor(s) | Takada, Shinji |
| State/Country | JPX |
| Assignee | Sony Corporation |
| US References | 4,471,383 4,518,239 4,609,944 4,728,980 4,831,450 4,833,498 4,837,594 4,851,897 |
| US Class | 3581229 3581225 |
| Int. Class | H04M3/14 |
| Title | Camera with exchangeable lens device removably mounted on a camera body |
| Abstract | A camera has a main camera body containing a main microprocessor, and exchangeable lens devices interchangeably mountable on the camera body and each containing an auxiliary microprocessor storing information identifying the respective exchangeable lens device and characteristics of the latter. Each exchangeable lens device includes an imaging lens assembly having elements controllable by drive motors for adjusting respective functions, and the auxiliary microprocessor controls such drive motors in response to position control signals from the main microprocessor and provides to the latter the stored identifying information and sensed position signals corresponding to the positions of the controllable elements of the respective imaging lens assembly. The main microprocessor provides the position control signals on the basis of at least the sensed position signals and condition sensing signals which correspond to the focused or other condition of an image projected by the imaging lens means onto an imaging element within the camera body. |

# PATENT TALK

Abstract 3 struck my interest as it is the embodiment of an idea that occurred to me also, years ago, as I watched road graders and pavers at work. Here, Spectra Physics has put a microprocessor in the heart of such construction machinery to offer increased intelligence and efficiency of operation. Using acoustic pulses to measure distance, the processor continuously computes the correct position of the blade and drives hydraulic rams to accomplish the positioning. It should come as no surprise by now (since they are easily driven by the microprocessor) that displays are provided to the operator to give him more information about his machinery. One might naturally wonder if perhaps some engineer also may have suggested a link to a data logger. This information might be used to determine long-term wear and tear on the rams or hydraulic pumps or even the condition of the blade itself.

In an earlier column I reported on "intelligent engine transmissions" that sense the particular engine to which they are coupled in order to set the operating point of the transmissions. Abstract 4 by Robert Bosch Company of




| | |
|---|---|
| Patent Number | 5,062,564 |
| Issue Date | 1991 11 05 |
| Inventor(s) | Urban, Paul L. |
| State/Country | SC |
| Assignee | Cooper Industries |
| US References | 2,582,481 2,735,923 2,747,074 3,618,590 3,654,427 4,530,456 4,654,507 4,822,979 4,891,497 |
| Title | Rapid response soldering station |
| Abstract | A soldering tip is provided with a sensor to sense soldering tip temperature during a soldering cycle. The sensor is embedded in the soldering tip and positioned immediately adjacent to the tip's working surface. The sensor location provides rapid response to changing conditions at the tip's working surface. A microprocessor, responsive to the sensor, is provided to process the tip temperature data to control the power delivered to the heater which provides heat to the soldering tip. A visual and/or audio display also can be coupled to the processor. The microprocessor can be coupled to a further processor having long-term memory so that the collected data may be subsequently retrieved and displayed. |




| | |
|---|---|
| Patent Number | 4,914,593 |
| Issue Date | 1990 04 03 |
| Inventor(s) | Middleton, Christopher 0.; Robson, Colin L. |
| State/Country | CA |
| Assignee | Spectra Physics |
| US References | 2,775,748 2,942,258 2,972,143 3,044,195 3,094,693 3,102,983 3,360,794 3,490,539 3,516,051 3,588,795 3,721,978 3,749,504 3,749,997 3,768,097 3,943,491 3,995,267 4,006,394 4,030,832 4,053,018 4,062,634 4,064,945 4,081,033 4,136,508 4,199,246 4,225,226 4,225,950 4,300,638 4,402,368 4,414,792 4,437,295 4,437,619 4,459,689 4,466,076 4,470,299 4,507,910 4,561,064 4,573,124 4,578,997 4,630,226 4,663,712 4,715,003 4,731,762 4,733,355 4,775,940 4,805,086 4,807,131 |
| Title | Method for automatic depth control for earth moving and grading |
| Abstract | A method and apparatus for automatically controlling the depth of earth grading for utilization with a grader or paver is disclosed. The method includes determination of time taken for an acoustic pulse to travel from a transducer to a reference surface and back, with this value being used to calibrate a microprocessor-controlled distance-measuring device. As the grader moves over a surface to be graded, the distance to the reference surface is constantly detected by a repeated emission and detection of such acoustic pulses. The timing of the echoed pulses is converted to addresses in a look-up table which contains control words symbolizing commands to be given to hydraulic rams carried by the grader. By implementing these commands, the depth of the blade relative to the reference surface is constantly updated, compensating for variations with the height of the reference surface. A thermistor provides automatic compensation for temperature variations as the grading takes place. Displays provide the operator of the vehicle to show what type of adjustments are being made to the blade, and whether the height of the reference surface is outside the range of sensitivity of the follower. The follower is automatically calibrated for a given blade depth by repeated incrementation of a delay time variable until a zero adjustment command is generated for the blade control. |

# PATENT TALK

Germany goes the natural next step to cover the microprocessor embedded within the transmission which essentially implements a table look-up procedure based on engine speed and operating load to control modulation pressure. While local display does not immediately suggest itself here, one wonders if data logging could perhaps warn of impending wear or damage before it becomes a problem.

Abstracts 5 and 6 are both related to microprocessors embedded within exercise equipment. The former abstract presents a device which serves to provide control over resistive forces exerted on the user's body based on measured speed and torque data. It uses an electromagnetic or fluid clutch that can be controlled by the microprocessor to modulate the forces delivered to the body from a constant-

| | |
|---|---|
| Patent Number | 5126,940 |
| Issue Date | 1992 06 30 |
| Inventor(s) | Haubner, Georg |
| State/Country | DEX |
| Assignee | Robert Bosch GmbH |
| US References | 4,269,281 4,488,456 4,494,422 4,736,301 4,781,655 4,791,568 4,813,307 4,845,618 4,955,259 4,981,053 4,987,544 |
| US Class | 3641424.1 741866 741867 3641431.04 3641431 .12 |
| Int. Class | G06F 15150 |
| Title | Method of automatically controlling modulation pressure in an automatic transmission including addressing a stored engine data matrix with a combined address formed from both digitized engine speed and load |
| Abstract | The method of controlling the modulation pressure in an automatic transmission of a motor vehicle engine includes determining which of n types of engines is being controlled, converting a digitized measured engine load value to one of eight scaled binary values, converting a digitized measured engine speed value signal to one of 32 scaled binary values, combining the scaled binary values of both engine speed and load into an address word for addressing one of n.times.256 storage locations of a data storage device containing reference control data, addressing the storage location corresponding to the address word and transferring the contents of the storage location so addressed to a microprocessor, which generates a current controlling a modulation pressure valve connected to the transmission according to the predetermined engine speed and load-dependent data. For greater control sensitivity, reference data is retrieved from adjacent storage locations and an interpolation of the reference data in those locations is performed. |

| | |
|---|---|
| Patent Number | 5,015,926 |
| Issue Date | 1991 05 14 |
| Inventor(s) | Casler, John A. |
| State/Country | CA |
| US References | 4,276,887 4,628,910 4,647,039 4,678,184 4,709,917 4,726,582 4,765,315 4,842,274 4,848,152 4,869,497 |
| Title | Electronically controlled force application mechanism for exercise machines |
| Abstract | A force development system for the application of controlled variable speeds and torque forces in exercise machines utilized to strengthen and develop body muscles of an exercising person. The system includes a constant-speed high-torque electric drive motor mechanically coupled to a dynamic clutch device in which the controlled coupling of the rotary force input assembly of the clutch to the rotary force output assembly of the clutch is accomplished via electromagnetic coil activation of metallic powder particles forming coupling particle chains between the input and output assemblies of the clutch. Alternatively, the dynamic clutch of the system may be a fluid clutch containing electrorheological fluid. The force development system of the invention also includes a speed reduction device between the dynamic clutch and the exercise machine to which the system is applied. An electronic sensor, interconnected to a microprocessor, senses the speed, motion, and torque force of the system's output shaft. A control unit (interconnected to the drive motor, the electromagnetic coil of the dynamic clutch, and the microprocessor) is directed by the microprocessor (in relation to the speed and torque force information sensed by the electronic sensor) and in turn controls the coupling torque of the dynamic clutch whereby controlled variable speeds and resistive forces are applied to the resistive force mechanism of the exercise machine. |

# PATENT TALK

| | |
|---|---|
| Patent Number | 5,089,960 |
| Issue Date | 199202 18 |
| Inventor(s) | Sweeney, James S., Jr. |
| State/Country | CA |
| Assignee | Laguna Tectrix, Inc. |
| US References | 4,196,528 4,408,613 4,542,897 4,579,335 4,906,192 4,919,416 4,976,424 |
| Title | Racing system for exercise machines |
| Abstract | A racing system for a group of exercise machines is disclosed. The race is entirely flexible, in that each exercise unit communicates electronically with all of the other potential racing units. Any user may offer a race, accept or reject another user's race offer, or join a race during a limited countdown period. More than one race can be underway. For cost reduction, a daisy-chain hookup is used, in which each unit's microprocessor has an input port receiving message flow from the output port of the preceding unit, and an output port transmitting message flow to the input port of the following unit. The racing function is controlled by the same microprocessor which is embedded in each exercise machine as the controller for that machine, receiving commands from the user and feedback from the machine. |




| | |
|---|---|
| Patent Number | 4,946,513 |
| Issue Date | 19900807 |
| Inventor(s) | Del Prato, Daniel J.; McKenna, David R.; Larson, Sherman L. |
| State/Country | NJ |
| Assignee | Sherman Industries, Inc. |
| US References | 3,495,287 3,626,536 3,793,663 3,795,929 3,822,430 4,462,133 4,719,932 4,726,388 |
| Title | Automated car wash system |
| Abstract | This invention is an automated system and method for washing automobiles. The invention provides a device which sprays liquid onto the vehicle, while closely following the general contour of the vehicle. The system determines that contour by analyzing and recording patterns of broken light beams when the vehicle passes by an array of photoelectric sensors. The system uses the stored information about the contour of the vehicle to control the movement of a spray bar which contains a set of nozzles. As the vehicle is pulled into the washing area by a conveyor, the spray bar initially moves with the vehicle, spraying the front grill while maintaining a constant distance from the vehicle. Then, the spray bar reverses direction, while the vehicle continues to move forward. The spray bar then travels around the vehicle contour, while adjusting the direction of the nozzles so that the liquid flows in the proper direction. After having traced the entire contour, the spray bar reverses direction again. The spray bar now follows the vehicle, spraying liquid towards the rear fender, until the washing cycle is complete. The spray bar oscillates axially while spraying, to ensure that the liquid will reach a large area. The invention also includes a unique shifting mechanism which adjusts the axial position of the spray bar, and thus determines the center of the axial oscillations. The apparatus is preferably controlled by a microprocessor, or its equivalent. |




speed electric drive motor. It seems to suggest application in bicycle-type exercise machines, although it might not be limited to them. With a processor embedded within each exercise machine, extensions to the machine's capabilities such as those suggested by Laguna Tectrix in Abstract 6 are quite natural. By augmenting the operator interface a bit and adding daisy-chained input and output communication ports to the existing embedded processor, one can link several exercise machines together. It is now possible for a group of people to "race" their exercisers in competition.

A seemingly somewhat mundane mechanical contraption is the automatic car wash. Yet, an increase in efficiency of such a machine means faster and better washes as well as lower operating costs. The patent covered by Abstract 7 describes such a "smart car wash" which, under control of a microprocessor, senses the contour of the vehicle from broken light beams and directs the spray bar t follow that contour. It is natural to assume that a close, vehicle-specific, directed spray pattern would be more efficient than a more general, universal one. The car gets cleaner in less time while using less water. One wonders whether or not such a system has sufficient resolution to detect protruding antennas, emblems, and such. With a microprocessor present, natural extensions to the system which we've seen before become possible. Specifically, linking each cleaning bay to a central site might permit control over individual speed, and hence volume of water flow, in order to minimize peak pumping pressure/flow

# PATENT TALK

requirements. Cleaning cycles could be coordinated by controlling conveyor speeds and sprayer flow so that congestion at the exit gate is minimized and operator workload is made more uniform.

As we've seen here, there seems to be no end to where microprocessors may be used to improve the operation, efficiency, or control of devices. You may have been struck, as I was, that many of the applications are basically mechanical in nature. It seems that microprocessors were first used as computing devices. Next they were applied primarily to electronic systems. But nowadays the greatest applicability appears to be in mechanical systems. If this apparent trend is accurate, the microprocessor practitioner of the future should be prepared to work well in interdisciplinary efforts where he is rubbing shoulders with those who specialize in areas quite different from his own bent. ❑

## 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

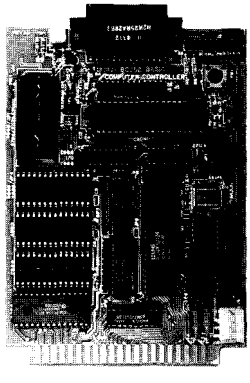
MicroPatent databases include the abstract-only APS version; FullText, which contains the entire patent without drawings; PatentImages, for the complete patent listing including drawings; and other specialized databases for just chemical, computer, or European patents.

*Russ Reiss holds a Ph.D. in EE/CS and has been active in electronics for over 25 years as industry consultant, designer, college professor, entrepeneur, and company president. 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.*

## IRS

425 Very Useful
426 Moderately Useful
427 Not Useful

**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!

**INPUT/OUTPUT**
- Console RS232 – autobaud detect
- Line printer RS-232
- Three 8-bit parallel ports
- EXPANDABLE1
  - Compatible with 12 BCC expansion boards

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

**To Order Call 1-800-635-3355**
**Tel: (203) 871-6170**
**Fax: (203) 872-2204**

| | | |
|---|---|---|
| BCC52 | Controller board with BASIC-52 and 8K RAM | $1 89.00 Single Qty |
| **BCC52C** | Low-power CMOS version of the BCC52 | $199.00 |
| **BCC52I** | 40°C to +85°C industrial temperature version | $294.00 |
| **BCC52CX** | Low-power CMOS, expanded BCC52 w/32K RAM | $259.00 |

CALL FOR OEM PRICING

**MICROMINT, INC.** 4 Park Street, Vernon, CT06066
in Europe. *(44)0285-658122 • in Canada: (514)336-9426 • in Australia. (02)888-6401*
*Distributor Inquiries Welcome!*

#119

TIRED OF WAITING FOR THE PROMPT ?

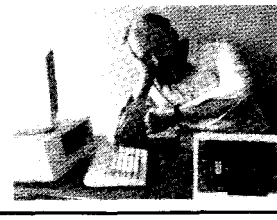
Speed up with a ROM DRIVE! Boots DOS and programs instantly. Also used to replace mechanical drive completely in controllers or diskless workstations. The only perfect protection from viruses. Easy to install half-size card.

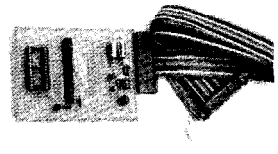
MVDISK1 128k $75
MVDISK2 720k $150
MVDISK3 2.88m $195

**$75**

Quantity discounts!

**DOS IN ROM!**





**$95 EPROM PROGRAMMER**

- Super Fast Programming
- Easier to use than others
- Does 2764/27080 (8 Meg)

WORLDS SMALLEST PC !!!

ROBOTS ALARMS RECORDERS DOS

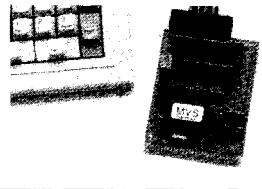
THREE EASY STEPS: **$27** 1K QTY
1. Develop on PC
2. Download to SBC **$95** SGL QTY
3. Burn into EPROM

-2 PARALLEL -LCD INTERFACE
-3 SERIAL -KEYBOARD INPUT
-PC TYPE BUS -REAL TIME CLK
-BIOS OPTION -BATTERY OR 5V

FREE SHIPPING IN U.S.

5 YEAR LIMITED WARRANTY

**8088 SINGLE BOARD COMPUTER**



.MVS Box 850 Merrimack, NH (508) 792 9507

46

# 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**

*In just ifs first month of operation, the Circuit Cellar BBS Internet connection has been fabulously successful. I could fell when everyone received their copy of the last issue because information requests to the BBS jumped markedly. If you haven't made contact with us yet, give if a fry.*

*We start this month with a discussion of AC and phase conversion. This is an area that can get info some pretty massive machinery. Next, we look at calculating typical battery life for a given application. Finally, we get some very useful pointers in the black art of PC board autorouting.*

## AC/phase conversion

**Msg#:17842**
From: DAVE ANDERSON To: ALL USERS

Anyone versed in AC and phase conversion? Got a friend that has some problems setting up a CNC Bridgeport. He has 220-V single phase coming in. Then it runs through a phase converter (looks like just a big motor). But the voltage coming out is:

Phase 1: 98 VAC; Phase 2: 128 VAC; Phase 3: 124 VAC

The spec call for ±5%. He has tried a "Buck reducer transformer," which will also raise a low leg. None of the combos give a good result.

**Msg#:17895**
From: ED **NISLEY** To: DAVE ANDERSON

I'll cheerfully admit I'm no expert on this, but I think the way those rotary converters work gives you a low leg.. but it doesn't matter once you get the load up to speed. Give it a go and see what happens; if the mill doesn't start, then I'm all wet.

A *real * rotary converter is a motor driving a generator, but the simpler ones are just a three-phase motor with the incoming AC driving one leg and the output taken from all three legs. The asymmetry means the voltages don't add up, but you get enough torque to drive the load anyway.

I bet Pellervo can write a dissertation on this that'11 not only bail you out, but demonstrate to everyone's satisfaction that I don't know what I'm talking about. Come in, Pellervo!

**Msg#:18298**
From: PELLERVO KASKINEN To: DAVE ANDERSON

Regardless of what Ed tries to say, I cannot solve your puzzle with the amount of information you present in your message. The missing part is a detailed description of the components in the Bridgeport CNC machine.

Here are a few tidbits, though. An ordinary asynchronous (squirrel gage) motor is extremely robust and would run no problem at all with the voltages you list, unless you try to get out a full 100% of the nameplate power. Actually, your voltage measurements should be taken phase-to-phase rather than phase-to-neutral. Thereby a better indication of the phase shift would become evident.

Anyway, my rule of thumb is that the motor used for the phase converter duty should be rated 3 to 5 times the power of the 3-phase load. Close to the 3-times mark for nice motor loads and up for any less than ideally balanced loads. You do not state whether there is an actual malfunctioning of the CNC machine, but my understanding is that it probably uses a set of servo motors that are fed from switching amplifiers, which again are fed from full-wave-rectified AC, using hefty capacitors. Now, the capacitors may not be quite hefty enough when the line voltages are not stiff and symmetrical. That would cause confusing ripple. The remedy would be adding some more of those monstrous electrolytic capacitors, even though that causes ever worse waveform distortion. The AC current runs only during the peaks of the sine wave and the phase converter does not like that too much.

**Msg#:18396**
From: BRENDAN KEITH To: PELLERVO KASKINEN

On a related topic, what are the best type of motors to use as I-phase generators? Universals or shaded-pole or what?

**Msg#:18574**
From: PELLERVO KASKINEN To: BRENDAN KEITH

Generally speaking, for a generator action you would need a SYNCHRONOUS generator. A universal motor does not produce any distinct frequency if it is rotated for a generator action. This is due to the commutator. It is basically a DC construction with inductances minimized so it would also be able to cope with the AC supply.

A shaded pole motor is not practical for anything beyond a few watts as a motor, and as a generator it would need continuous supply of the reactive (AC) power from the line, so why not pull the real power from there as well?

Of the asynchronous types, only a capacitor start/capacitor RUN type motor can come close to covering a range of powers as a generator, but it still needs the reactive power from the grid. So what would be the benefit?

No, the sad fact remains that you need a synchronous motor to serve a generator duty. Then you have to control the magnetizing for getting out a proper voltage and the mechanical speed to get out the proper frequency.

**Msg#:18638**
From: BRENDAN KEITH To: PELLERVO KASKINEN

I'm not familiar with the term synchronous motor. How about a DC motor? With its permanent magnets, it should induce a good magnetic flux, no? And how would you go about controlling the "magnetizing for getting out a proper voltage," as you mention above?

**Msg#:18794**
From: PELLERVO KASKINEN To: BRENDAN KEITH

The synchronous motor (and generator) could in theory have a permanent magnet and thereby no control over the voltage. Actually, such permanent-magnet motors are indeed used in time-critical systems like time delay relays or wall clocks. The term ***synchronous*** simply means that the rotational position of the shaft is continuously in full synchronization with the power line and actually with the power station generators. Astonishing to think, isn't it? Consequently, there is no way to change the speed of the synchronous motor unless you can control the mechanical speed of the turbine feeding the generator.

Turning back to the synchronous generator. You control the output voltage by changing the intensity of the electromagnet used instead of a permanent magnet. Let's see. The power windings are in the stationary part of the generator, while the excitation magnet is in the rotor. It is fed through slip rings with a DC current so the strength of the magnet can be controlled.

One of the only a handful of simple principles in the power generation and transfer is that your magnetizing level does 'not* affect the power that the generator is putting out. All it does is control the voltage and REACTIVE POWER. To increase the REAL POWER output you have to TRY to run your generator faster than the other generators and loads in the power grid. This, of course, by assuming that the generator is connected to a common grid. Anyway, you do not actually keep on running faster, because your turbine has a governor and a limited power to supply into the common consumption. All that actually happens is that your generator keeps running in a phase-locked fashion with the rest of them, but has just advanced the phase a small amount.

The voltage issue is important because all the asynchronous motors out there NEED reactive power in addition to the real power. Real power does the work, while reactive power makes it possible for it to function. A synchronous motor (in addition to the generator] can GENERATE reactive power for other motors to use WHILE itself consuming real power from the network. Funny, isn't it? But all is just reciprocal. To get the simple, robust asynchronous or squirrel gage motor to function, it needs to "transfer" the magnetism to the rotating member somehow. And that is what uses the reactive power.

Now, DC motors and generators are slightly different in that the commutator is used to automatically handle much of the game with the real and reactive portions of the power. You sort of recirculate things internally, but you lose the clarity that the synchronous motor/generator has. You do not run at the same speed with the other devices you may be sharing the network with. And an increased DC generator speed would cause an increased voltage in addition or just as a side effect of increased power, assuming permanent magnet excitation. However, you could use electromagnets as well and then that also affects the voltage. But the fact remains that you would need expensive inverters to generate an AC from the DC. Only one simplification results from the DC generator: your speed governor would not need to be very accurate. I hope I'm not confusing more than what I clarify with all this babbling.

## Battery life

**Msg#:20074**
From: LEE LEDUC To: ALL USERS

How can I determine how long a battery will last when powering a circuit? For example, I have a circuit that uses a 9-volt alkaline battery and it draws 10 mA of current. I assume this device will stop working when the battery voltage drops to 6 volts. Since the discharge curve of an alkaline cell is relatively flat, would (amp-hour rating)/(current drawn) be a good estimate of battery life?

**Msg#:20168**
From: JON **KIRWAN** To: LEE LEDUC

As long as the current draw is in a reasonable range and you have a good estimate of the amp-hour rating at a current draw near the value you're using, the answer is yes.

For example, the typical 9-V alkaline battery provides about 500 mAh at a draw of 1 mA, 470 mAh at a draw of 10

mA, and 280 mAh at 100 mA (this assumes a "dead" battery occurs when the per-cell voltage hits 1 V). Your suggested draw of 10 mA is quite realistic for 9-V alkaline batteries. The calculated value of (470 mAh / 10 mA) = 47 hours is reasonable.

Your suggestion that alkaline batteries have a "flat" discharge curve is wrong, though. The LeClanche and zinc-chloride batteries do have a somewhat greater voltage droop, but the alkaline voltage droop looks quite similar to these poor performers. Nearly everything else is better. For example, nickel-cadmium, mercury, silver-oxide, lithium-oxyhalide, and even lead-acid batteries all have a lower voltage droop over their discharge life. Pick a chemistry at random and it will likely be better than alkaline, regarding voltage droop.

Operating temperature and the loaded duty cycle (versus the average load) will also affect your results. The numbers you compute are okay for making some comparisons of different chemistries, but you probably should run a test or two anyway to verify your estimates and see how long it really works. (Of course, there are many other factors to compare when you still haven't settled on a chemistry-such as weight, cost, shelf-life, safety, availability, discharge curve, high-current handling, and internal impedance, to name some. Once you've picked your chemistry, it's just a matter of selecting a battery with a practical lifetime, cost, and availability.)

Oh, watch for the newer sulfur anode, aluminum cathode batteries. Initial results are very, very impressive in nearly every regard.

## Autorouting

**Msg#:18225**
From: TOM CARTER To: ALL USERS

Can anyone give me a few tips on helping an autorouter complete a PCB 100%? I have HiWire II and it can't seem to complete a relatively simple layout unless I use 8-mil rules or less.

**Msg#:18230**
From: JEFF BACHIOCHI To: TOM CARTER

What is the density of the boards being routed? Software usually will give density of the board where its area is divided by an equivalent number of 14-pin DIPs (all components are computed into a magic equivalent).

As the density falls below 1, completion times become longer and longer. If it is too dense, the router will never be able to find a path. At some point the router will give up, depending on the rules you have given it to play by.

**Msg#:18640**
From: TOM CARTER To: JEFF BACHIOCHI

The board is 3.5" x 3" and has 375 holes. I am trying to use 12-mil traces and spacing, and about 20- or 30-mil power traces. What power trace do you use! It's a low-power board of 5 V at under 100 mA total. Can I get by with 16-mil power traces? The AR can handle that.

**Msg#:18651**
From: JEFF BACHIOCHI To: TOM CARTER

I like to choose traces that are multiples of the grid I've chosen. This is normally 25 mils for one-between ICs. A 12-space, 13-trace rule works well for almost everything I do. Power I like to have as large as physically possible (naturally power and ground layers are best). If two 13-mil traces are run parallel and the space between them is filled in, you end up with a 38-mil trace, which would be a logical choice (based on the grid). If I'm using 60-mil IC pads, space between two adjacent pins is 40 mils. I can still squeak through a trace even if the traces to those pads are 60 mils, so using large power and ground widths is usually possible.

From a strictly current-carrying standpoint, 16 mils is probably fine. However, it doesn't cost any more to leave more copper on the substrate.

**Msg#:20263**
From: DALE NASSAR To: TOM CARTER

I have just completed a couple of PCBs using Wintek's PCB layout and autorouter software (ver.2.0r3). After many painful hours of experimenting, I think I can give some tips on getting 100% route completion on boards that tech support will tell you are impossible to route. My board is a dense double-sided microcontroller PCB using double trace (two traces between pins) only when necessary. Fortunately, I did not need any.

*First, of course, lay out the PCB footprints to minimize line crossings in a rat's nest run, but don't choose an upside-down placement just to uncross a couple of lines.

*Keep in mind (while manually routing) that one side of the board is for predominantly horizontal traces and the other vertical. It is usually not worth it to place a via for a tiny segment to run at a right angle to another on the same side of the board, for example, on the vertical side just run a horizontal trace if it is very short.

•Be sure the components aren't placed so close together that they bump into each other-this is EXTREMELY easy to overlook with a zoomed-in PCB on the screen. It's a VERY good idea to place the components on one of those $3 Radio Shack type prepunched boards as you place the components on the layout. Also make sure there is enough clearance around connectors for cables. Many times you will need a 50-mil spacing grid, however I have been unable

to find any. What you can do if you have a plotter is plot the solder side pads onto a Cu sheet and then drill holes in each pad. I tried this and it works great. Make the pads BIG (just for the plot), plot with donut centers (a must for a sane drilling session). You must flip the solder side pads before plotting. I bought a kit from General Consulting for making PCBs on a plotter, but had no luck with drawing traces.

•This one learned the hard way: some of the PCB libraries use IC footprints that are 0.4" across-I put it where a 0.3" socket should have gone and didn't know it 'till the socket wouldn't go in the holes on the finished board.

*Most people like to first (before autorouting) run thick power and ground traces connecting the decoupling capacitors to IC power and ground pins. I have found (with this package) that this can drive the autorouter crazy and prevent a good run. I have six ICs on the PCB and found that removing the manually placed decoupling traces from only two of the ICs allowed the autorouter to go from 84% to 99.4% completion-it was running out of (conventional) memory, thus preventing an exhaustive search of all "possible" paths. This results in a "SWITCHING TO SELECTIVE SEARCH STRATEGY" message. After using the extended memory router with 8 MB, I went right back to the conventional-it seems that too much memory allows some really wild twists and turns to be searched out and run. My solution was to use conventional memory and move the decoupling traces around and run the router again. You will eventually get an acceptable route of the decoupling traces. Look for short runs or small loop areas in the cap/IC circuit. On one board, I installed the caps under the socket-be sure you have clearance for the IC when fully seated.

•A bus connector can really mess things up (I used a 50-pin connector with 16 address lines, 8 data, etc.]. You will have to arrange these connections from the PCB-not the schematic. Carefully arranged connections here can dramatically improve routing. I like to try the router without the connector in the beginning.

•I have noticed that small vias seem to make a significant difference in the route completion. You usually don't have to make them bigger than the connection trace.

*Don't let traces run too close to mounting (or other) holes, especially those that will use a washer and screw.

•If you must run two traces between pads, then make only that pad smaller than the rest-no need to reduce them all. You can make it oval in the direction of the trace. Also, the solder side pad and the component side pad can be different sizes-keep as much pad as possible. I have found that the rat's nest crossing counter does not work properly with different pad sizes for the same pin-you just have to use the visual.

*When using "duplicated function" ICs (like a '373 octal latch) you can arrange the connection to the individual latch that minimizes route complexity-use the rat's nest. Keep in mind that the 74LS573 is an octal latch with the pins arranged with inputs and outputs on opposite sides for easy routing-I never use the '373 on PCBs anymore.

*The design rules may state that power and ground must be thick, however some power and ground traces never carry high-current pulses (e.g., enable pins, 8031's EA, etc.). You can save some space by running these connections with thin traces.

● Try to run manual traces at right angles. This package treats a sloped line as the diagonal of a rectangular keepout area-I wish they would fix this!

*The manual says that decreasing the "VIA COST" parameter will let the board go to higher completion by using more vias-I have found the opposite to be true when double-trace rules are used. The way I see it, it assumes the vias are more expensive and searches harder for routes. The phase of the moon has a lot to do with this.

*Don't worry about neat traces in the beginning. Very often, in manual routing, you will have to delete a heaping

WORKS

Does your Big-Company marketing department come up with more ideas than the engineering department can cope with? Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?

Steve Ciarcia and the Ciarcia Design Works staff may have the solution. We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems. Whether you need an on-line solution for a unique problem, a product for a startup venture, or just experienced consulting, the Ciarcia Design Works is ready to work with you. Just fax me your problem and we'll be in touch.

Remember...
a Ciarcia design works!
Fax (203) 871-8986

# CONNECTIME

mess of traces and run another way. During cleanup it is easy to see the "neatness" of a routed board by separately printing a rough 1: 1 scale draft of the component side traces and the solder side traces. You can also print the silkscreen or pad layers to make sure everything fits.

*Use this suggestion with caution: With SRAM you can scramble the address and data lines in any order that minimizes routing. This works because you read and write it every time in the same order. Don't do this on any memory that is read or written off of your board-this includes NVRAM. This tip can dramatically improve routing.

•Finally, back up every 10 minutes or better while manually routing. A 15-minute data loss almost never gets you back where you were before. I like to use a RAM drive for almost instant backup-you don't lose your train of thought. The old HiWire Plus had autobackup, which the new version doesn't. They also seemed to have removed a lot of other necessary functions.

I would like to hear from anyone using any of the higher-priced packages since this is the only one I have worked with.

**Msg#:21464**

From: JAMES MEYER To: DALE NASSAR

I'm using OrCAD's PCB program. Right now it's the "II" version, but the 386 version is on its way.

Your comments and hints are certainly right on the money. Even though my package is different, your suggestions are almost exactly what I use.

I especially liked the one about double checking the footprints that are supplied with the software. Several of OrCAD's footprints were twice the size that they should have been. I guess somebody was making a 2x layout.

OrCAD suggests that bypass caps must be manually routed, but I've found a work-around that lets me let the autorouter do most of the work.

The reason that bypass caps are such a problem is that they are all connected in parallel in the schematic, and the net list doesn't associate a particular cap with a particular IC. The autorouter just blindly follows the net list in a first come, first served fashion, and that can make for some really bizarre routes.

I've made a special footprint to use just for bypass caps. It has three pads instead of two. I've also made a special bypass cap symbol for use in the schematic capture program. Basically, the cap is turned into a three-terminal device. One terminal is connected to one side of the cap. The other side of the cap goes to both of the remaining terminals. The singly connected terminal is connected to ground in the schematic. One of the two terminals on the other side is connected to the IC and the other terminal on that side is connected to the supply voltage. The connection from the IC to its supply voltage then has to go *thorough* the bypass cap. That makes the net list keep the IC and cap together.

The footprint has the double terminal with one pad in the normal position and the other as close as the design rules will allow just outside that. I don't even bother to put a hole in the second pad. The autorouter does its job except there's no connection from the IC to its supply. I go back and manually add a connection between the double pads after the board's routed.

To maintain an uninterrupted flow of copies for your students, PLEASE SEND YOUR UPDATED SPRING SEMESTER CLASS LISTS TO:

Rose Mansella
The Computer Applications Journal
4 Park St. Tel: (203) 875-2199
Vernon, CT 06066 Fax: (203) 872-2204

*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.

## I R S

428 Very Useful 429 Moderately Useful 430 Not Useful

# STEVE'S OWN INK

## Embedded in Everything



The low prices of microcontroller chips has allowed them to be designed into a wide variety of products. Consumers now get products with embedded controllers for not much more than ones that don't.

Unfortunately, the capabilities afforded with micros aren't always being used most effectively. Product marketing managers these days often stuff in a micro just to drive a handful of buttons and blinking lights that invoke esoteric and cryptic modes and functions. The harder it is to use, the more powerful and sophisticated the device must be (read "higher priced").

Products that have a lot of blinking lights and beeping buttons might have a certain degree of entertainment appeal. But are they better products? I guess the point I am trying to make is that not everything that exists has to have a controller attached to it, and that the public has been putting up with too much packaging zeal.

We've been lucky up to now. The consumer has been willing to spend three times as much for a PID house thermostat that reports and controls temperature within a tenth of a degree even though most heating sources have control resolutions no better than ±3 degrees. Oh well, at least their micro is running sophisticated algorithms to direct their home heating needs.

But there is also a potentially grim side to using nonmechanical controllers. If this thermostat takes a lightning hit or power surge and your entire tank of tropical fish perish, will you go back to a $20 mechanical thermostat with 3" resolution (not lightning sensitive) or kid yourself into paying $100 for the "ultimate" thermal control system again?

I think the new battle cry should be "embedded intelligence" instead of "embedded control." I mean, how smart does my toaster have to be? I don't care to know the surface temperature of my toast or the melting factor of butter as it relates to the toast temperature. What I want is a toaster that I don't have to unplug every time a thunderstorm is coming for fear that it will become "toasted."

To me, applying embedded intelligence means seeking authoritative improvement over the mechanical controller rather than just cost-effective replacement and added displays.

Let's admit it, not every product with an embedded controller is better than one without one. The mission of the product designer should be to design products that do indeed improve task management and supervision, but it shouldn't at the same time introduce entirely new complications and sensitivities that can obliterate its own existence. When product marketing starts screaming that they want bells and whistles, go ahead and give it to them, but make sure you glue on a few MOVs, ultrafast diodes, and transient suppressors. Adding embedded intelligence has to mean solving more problems than it creates.

